How does digital data work? Imagine that you want to send a message by flashing a light. Your light switch offers two states: on and off. You can use sequences of ons and offs to represent various letters of the alphabet. To write down the representation for each letter, you can use 0s and 1s. The 0s represent the off state of your light switch; the 1s indicate the on state. For example, the sequence *on on off off* would be written as 1100, and you might decide that sequence represents the letter *A*.

Digital devices are electronic and so you can envision data flowing within these devices as pulses of light. In reality, digital signals are represented by two different voltages, such as +5 volts and +.2 volts. They can also be represented by two different tones as they flow over a phone line. Digital data can also take the form of light and dark spots etched onto the surface of a CD or the positive and negative orientation of magnetic particles on the surface of a hard disk.

The 0s and 1s used to represent digital data are referred to as binary digits. It is from this term that we get the word *bit*—*bi*nary digi*t*. A **bit** is a 0 or 1 used in the digital representation of data.

REPRESENTING NUMBERS, TEXT, IMAGES, AND SOUND

How do digital devices represent numbers? **Numeric data** consists of numbers that can be used in arithmetic operations. For example, your annual income is numeric data, as is your age. Digital devices represent numeric data using the binary number system, also called base 2.

The **binary number system** has only two digits: 0 and 1. No numeral like 2 exists in this system, so the number two is represented in binary as 10 (pronounced "one zero"). You'll understand why if you think about what happens when you're counting from 1 to 10 in the familiar decimal system. After you reach 9, you run out of digits. For ten, you have to use the digits 10—zero is a placeholder and the 1 indicates one group of tens.

In binary, you just run out of digits sooner—right after you count to 1. To get to the next number, you have to use the 0 as a placeholder and the 1 indicates one group of twos. In binary, then, you count 0 (zero), 1 (one), 10 (one zero), instead of counting 0, 1, 2 in decimal. If you need to brush up on binary numbers, refer to Figure 1-25 and to the lab at the end of the chapter.

Decimal (Base 10)	Binary (Base 2)
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
1000	1111101000

FIGURE 1-25

The decimal system uses ten symbols to represent numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. The binary number system uses only two symbols: 0 and 1.

TRY IT!

The table shows the binary equivalent of numbers 1 through 11. What is the binary for the number 12?

❍ 10111

❍ 1100

❍ 10000

❍ 1111

Copyright 2013 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

The important point to understand is that the binary number system allows digital devices to represent virtually any number simply by using 0s and 1s. Digital devices can then perform calculations using these numbers.

◗ How do digital devices represent words and letters?

Character data is composed of letters, symbols, and numerals that are not used in arithmetic operations. Examples of character data include your name, address, and hair color. Just as Morse code uses dashes and dots to represent the letters of the alphabet, a digital computer uses a series of bits to represent letters, characters, and numerals.

Digital devices employ several types of codes to represent character data, including ASCII, EBCDIC, and Unicode. **ASCII** (American Standard Code for Information Interchange, pronounced "ASK ee") requires only seven bits for each character. For example, the ASCII code for an uppercase *A* is 1000001. ASCII provides codes for 128 characters, including uppercase letters, lowercase letters, punctuation symbols, and numerals.

EBCDIC (Extended Binary-Coded Decimal Interchange Code, pronounced "EB seh dick") is an 8-bit code used only by older, mainframe computers.

Extended ASCII is a superset of ASCII that uses eight bits to represent each character. For example, Extended ASCII represents the uppercase letter *A* as 01000001. Using eight bits instead of seven bits allows Extended ASCII to provide codes for 256 characters. The additional Extended ASCII characters include boxes and other graphical symbols. Figure 1-26 lists the Extended ASCII character set.

TRY IT!

Write out **Hi!** in Extended ASCII code. (Hint: Use an uppercase *H*, but a lowercase *i*.)

H ______

i ______

! ______

FIGURE 1-26

The Extended ASCII code uses eight 1s and 0s to represent letters, symbols, and numerals. The first 32 ASCII characters are not shown in the table because they represent special control sequences that cannot be printed. The two blank entries are space characters.

	00100000	>	00111110	\	01011100	z	01111010	ÿ	10011000	╢	10110110	╘	11010100	≥	11110010
!	00100001	?	00111111	]	01011101	{	01111011	Ö	10011001	╖	10110111	╒	11010101	≤	11110011
"	00100010	@	01000000	^	01011110	|	01111100	Ü	10011010	╕	10111000	╓	11010110	⌠	11110100
#	00100011	A	01000001	_	01011111	}	01111101	¢	10011011	╣	10111001	╫	11010111	⌡	11110101
$	00100100	B	01000010	`	01100000	~	01111110	£	10011100	║	10111010	╪	11011000	÷	11110110
%	00100101	C	01000011	a	01100001	⌂	01111111	¥	10011101	╗	10111011	┘	11011001	≈	11110111
&	00100110	D	01000100	b	01100010	Ç	10000000	₧	10011110	╝	10111100	┌	11011010	°	11111000
'	00100111	E	01000101	c	01100011	ü	10000001	ƒ	10011111	╜	10111101	█	11011011	∙	11111001
(	00101000	F	01000110	d	01100100	é	10000010	á	10100000	╛	10111110	▄	11011100	·	11111010
)	00101001	G	01000111	e	01100101	â	10000011	í	10100001	┐	10111111	▌	11011101	√	11111011
*	00101010	H	01001000	f	01100110	ä	10000100	ó	10100010	└	11000000	▐	11011110	ⁿ	11111100
+	00101011	I	01001001	g	01100111	à	10000101	ú	10100011	┴	11000001	▀	11011111	²	11111101
,	00101100	J	01001010	h	01101000	å	10000110	ñ	10100100	┬	11000010	α	11100000	■	11111110
-	00101101	K	01001011	i	01101001	ç	10000111	Ñ	10100101	├	11000011	ß	11100001		11111111
.	00101110	L	01001100	j	01101010	ê	10001000	ª	10100110	─	11000100	Γ	11100010		
/	00101111	M	01001101	k	01101011	ë	10001001	º	10100111	┼	11000101	π	11100011		
0	00110000	N	01001110	l	01101100	è	10001010	¿	10101000	╞	11000110	Σ	11100100		
1	00110001	O	01001111	m	01101101	ï	10001011	⌐	10101001	╟	11000111	σ	11100101		
2	00110010	P	01010000	n	01101110	î	10001100	¬	10101010	╚	11001000	µ	11100110		
3	00110011	Q	01010001	o	01101111	ì	10001101	½	10101011	╔	11001001	τ	11100111		
4	00110100	R	01010010	p	01110000	Ä	10001110	¼	10101100	╩	11001010	Φ	11101000		
5	00110101	S	01010011	q	01110001	Å	10001111	¡	10101101	╦	11001011	Θ	11101001		
6	00110110	T	01010100	r	01110010	É	10010000	«	10101110	╠	11001100	Ω	11101010		
7	00110111	U	01010101	s	01110011	æ	10010001	»	10101111	═	11001101	δ	11101011		
8	00111000	V	01010110	t	01110100	Æ	10010010	░	10110000	╬	11001110	∞	11101100		
9	00111001	W	01010111	u	01110101	ô	10010011	▒	10110001	╧	11001111	φ	11101101		
:	00111010	X	01011000	v	01110110	ö	10010100	▓	10110010	╨	11010000	ε	11101110		
;	00111011	Y	01011001	w	01110111	ò	10010101	│	10110011	╤	11010001	∩	11101111		
<	00111100	Z	01011010	x	01111000	û	10010110	┤	10110100	╥	11010010	≡	11110000		
=	00111101	[	01011011	y	01111001	ù	10010111	╡	10110101	╙	11010011	±	11110001		

Copyright 2013 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

Unicode (pronounced "YOU ni code") uses sixteen bits and provides codes for 65,000 characters—a real bonus for representing the alphabets of multiple languages. For example, Unicode represents an uppercase *A* in the Russian Cyrillic alphabet as 0000010000010000.

Why do ASCII and Extended ASCII provide codes for 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9? While glancing at the table of ASCII codes in Figure 1-26, you might have wondered why the table contains codes for 0, 1, 2, 3, and so on. Aren't these numbers represented by the binary number system? A computer uses Extended ASCII character codes for 0, 1, 2, 3 , etc. to represent numerals that are not used for calculations.

TRY IT!

Social Security numbers and the numerals in a street address are considered character data. If your address is 10 B St, how would you complete the Extended ASCII code?

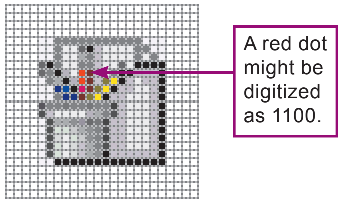
1	0		B		S	T
			01000010	00100000		

How can bits be used to store images? Images, such as photos, pictures, line art, and graphs, are not small, discrete objects like numbers or the letters of the alphabet. Images have to be digitized in order for digital devices to work with them.

Images can be digitized by treating them as a series of colored dots. Each dot is assigned a binary number according to its color. For example, a green dot might be represented by 0010 and a red dot by 1100, as shown in Figure 1-27. A digital image is simply a list of color numbers for all the dots it contains.

FIGURE 1-27

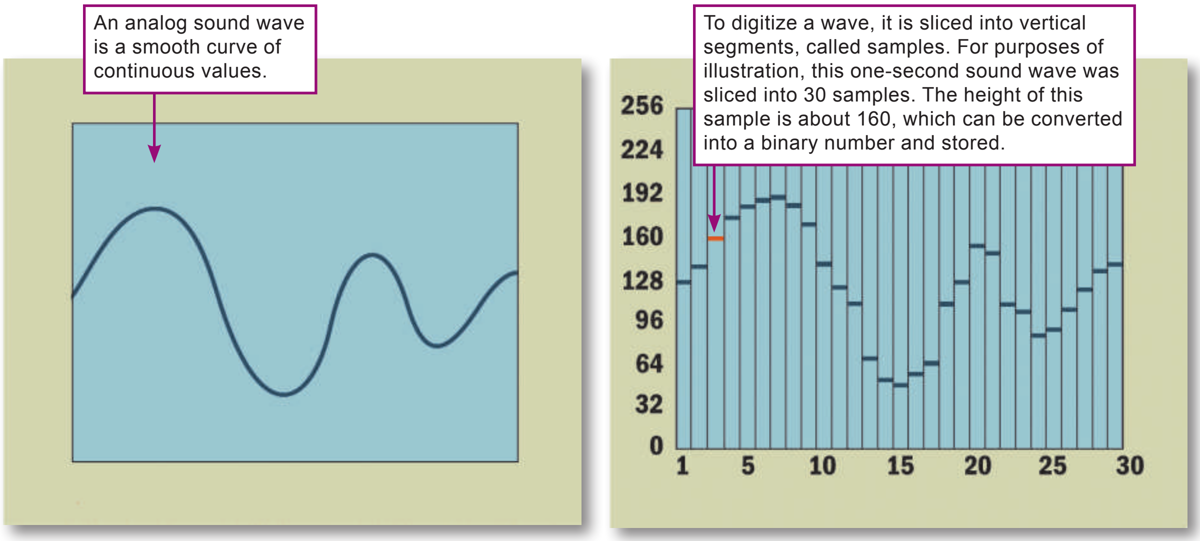
An image can be digitized by assigning a binary number to each dot.



How can bits be used to store sound? Sound, such as music and speech, is characterized by the properties of a sound wave. You can create a comparable wave by etching it onto a vinyl platter—essentially how records were made in the days of jukeboxes and record players. You can also represent that sound wave digitally by sampling it at various points, and then converting those points into digital numbers. The more samples you take, the closer your points come to approximating the full wave pattern. This process of sampling, illustrated in Figure 1-28, is how digital recordings are made.

FIGURE 1-28

A sound wave can be sampled at fraction-of-a-second time intervals. Each sample is recorded as a binary number and stored.



Copyright 2013 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.