

# 2

## **NUMBER SYSTEMS, OPERATIONS, AND CODES**



You are familiar with the decimal number system because you use decimal numbers every day. Although decimal numbers are commonplace, their weighted structure is often not understood. In this section, the structure of decimal numbers is reviewed. This review will help you more easily understand the structure of the binary number system, which is important in computers and digital electronics.

After completing this section, you should be able to

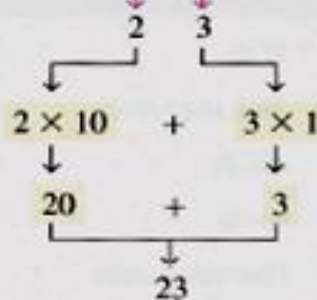
- Explain why the decimal number system is a weighted system
- Explain how powers of ten are used in the decimal system
- Determine the weight of each digit in a decimal number

The decimal number system has ten digits.

In the **decimal** number system each of the ten digits, 0 through 9, represents a certain quantity. As you know, the ten symbols (**digits**) do not limit you to expressing only ten different quantities because you use the various digits in appropriate positions within a number to indicate the magnitude of the quantity. You can express quantities up through nine before running out of digits; if you wish to express a quantity greater than nine, you use two or more digits, and the position of each digit within the number tells you the magnitude it represents. If, for example, you wish to express the quantity twenty-three, you use (by their respective positions in the number) the digit 2 to represent the quantity twenty and the digit 3 to represent the quantity three, as illustrated below.

The digit 2 has a weight of 10 in this position.

The digit 3 has a weight of 1 in this position.





The decimal number system has a base of 10.

The position of each digit in a decimal number indicates the magnitude of the quantity represented and can be assigned a **weight**. The weights for whole numbers are positive powers of ten that increase from right to left, beginning with  $10^0 = 1$ .

$$\dots 10^5 10^4 10^3 10^2 10^1 10^0$$

For fractional numbers, the weights are negative powers of ten that decrease from left to right beginning with  $10^{-1}$ .

The value of a digit is determined by its position in the number.

$$10^2 10^1 10^0 . 10^{-1} 10^{-2} 10^{-3} \dots$$

↑  
Decimal point

The value of a decimal number is the sum of the digits after each digit has been multiplied by its weight, as Examples 2-1 and 2-2 illustrate.

### EXAMPLE 2-2

Express the decimal number 568.23 as a sum of the values of each digit.

**Solution** The whole number digit 5 has a weight of 100, which is  $10^2$ , the digit 6 has a weight of 10, which is  $10^1$ , the digit 8 has a weight of 1, which is  $10^0$ , the fractional digit 2 has a weight of 0.1, which is  $10^{-1}$ , and the fractional digit 3 has a weight of 0.01, which is  $10^{-2}$ .

$$\begin{aligned} 568.23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) \\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0.1) + (3 \times 0.01) \\ &= \mathbf{500} + \mathbf{60} + \mathbf{8} + \mathbf{0.2} + \mathbf{0.03} \end{aligned}$$

**Related Problem** Determine the value of each digit in 67.924.

The binary number system is another way to represent quantities. It is less complicated than the decimal system because it has only two digits. The decimal system with its ten digits is a base-ten system; the binary system with its two digits is a base-two system. The two binary digits (bits) are 1 and 0. The position of a 1 or 0 in a binary number indicates its weight, or value within the number, just as the position of a decimal digit determines the value of that digit. The weights in a binary number are based on powers of two.

After completing this section, you should be able to

- Count in binary
- Determine the largest decimal number that can be represented by a given number of bits
- Convert a binary number to a decimal number

### Counting in Binary

The binary number system has two digits (bits).

To learn to count in the binary system, first look at how you count in the decimal system. You start at zero and count up to nine before you run out of digits. You then start another digit position (to the left) and continue counting 10 through 99. At this point you have exhausted all two-digit combinations, so a third digit position is needed to count from 100 through 999.

A comparable situation occurs when you count in binary, except that you have only two digits, called bits. Begin counting: 0, 1. At this point you have used both digits, so include another digit position and continue: 10, 11. You have now exhausted all combinations of two digits, so a third position is required. With three digit positions you can continue to count: 100, 101, 110, and 111. Now you need a fourth digit position to continue, and so on. A binary count of zero through fifteen is shown in Table 2-1. Notice the patterns with which the 1s and 0s alternate in each column.

The binary number system has a base of 2.



▼ **TABLE 2-2**

Binary weights.

POSITIVE POWERS OF TWO (WHOLE NUMBERS)									NEGATIVE POWERS OF TWO (FRACTIONAL NUMBER)					
$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	$2^{-5}$	$2^{-6}$
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0.5	0.25	0.125	0.0625	0.03125	0.015625

► **TABLE 2-1**

As you have seen in Table 2-1, four bits are required to count from zero to 15. In general, with  $n$  bits you can count up to a number equal to  $2^n - 1$ .

$$\text{Largest decimal number} = 2^n - 1$$

For example, with five bits ( $n = 5$ ) you can count from zero to thirty-one.

$$2^5 - 1 = 32 - 1 = 31$$

With six bits ( $n = 6$ ) you can count from zero to sixty-three.

$$2^6 - 1 = 64 - 1 = 63$$

A table of powers of two is given in Appendix A.

DECIMAL NUMBER	BINARY NUMBER			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

## Binary-to-Decimal Conversion

Add the weights of all 1s in a binary number to get the decimal value.

The decimal value of any binary number can be found by adding the weights of all bits that are 1 and discarding the weights of all bits that are 0.

### EXAMPLE 2-3

Convert the binary whole number 1101101 to decimal.

**Solution** Determine the weight of each bit that is a 1, and then find the sum of the weights to get the decimal number.

Weight:  $2^6$   $2^5$   $2^4$   $2^3$   $2^2$   $2^1$   $2^0$

Binary number: 1 1 0 1 1 0 1

$$\begin{aligned} 1101101 &= 2^6 + 2^5 + 2^3 + 2^2 + 2^0 \\ &= 64 + 32 + 8 + 4 + 1 = 109 \end{aligned}$$

**Related Problem** Convert the binary number 10010001 to decimal.

### EXAMPLE 2-4

Convert the fractional binary number 0.1011 to decimal.

**Solution** Determine the weight of each bit that is a 1, and then sum the weights to get the decimal fraction.

Weight:  $2^{-1}$   $2^{-2}$   $2^{-3}$   $2^{-4}$

Binary number: 0 . 1 0 1 1

$$\begin{aligned} 0.1011 &= 2^{-1} + 2^{-3} + 2^{-4} \\ &= 0.5 + 0.125 + 0.0625 = 0.6875 \end{aligned}$$

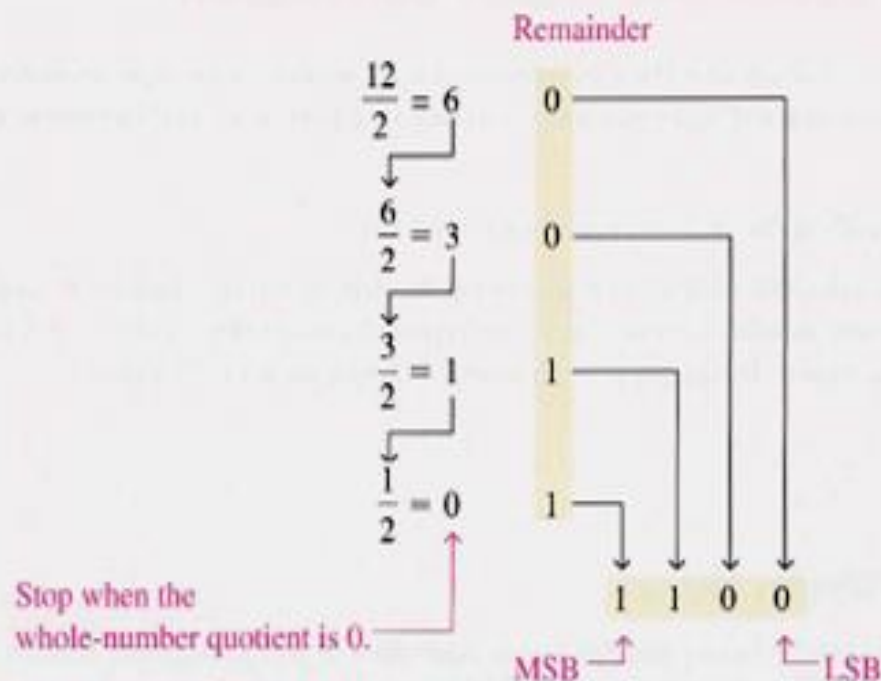
**Related Problem** Convert the binary number 10.111 to decimal.

## DECIMAL-TO-BINARY CONVERSION

### Repeated Division-by-2 Method

To get the binary number for a given decimal number, divide the decimal number by 2 until the quotient is 0. Remainders form the binary number.

A systematic method of converting whole numbers from decimal to binary is the *repeated division-by-2* process. For example, to convert the decimal number 12 to binary, begin by dividing 12 by 2. Then divide each resulting quotient by 2 until there is a 0 whole-number quotient. The **remainders** generated by each division form the binary number. The first remainder to be produced is the LSB (least significant bit) in the binary number, and the last remainder to be produced is the MSB (most significant bit). This procedure is shown in the following steps for converting the decimal number 12 to binary.





### EXAMPLE 2-6

Convert the following decimal numbers to binary:

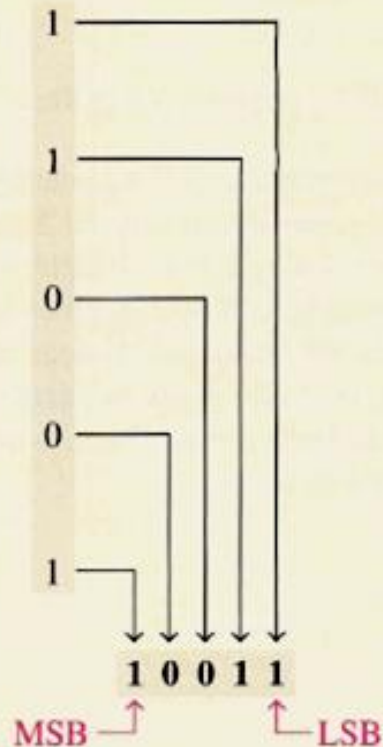
(a) 19    (b) 45

*Solution*

(a)

Remainder

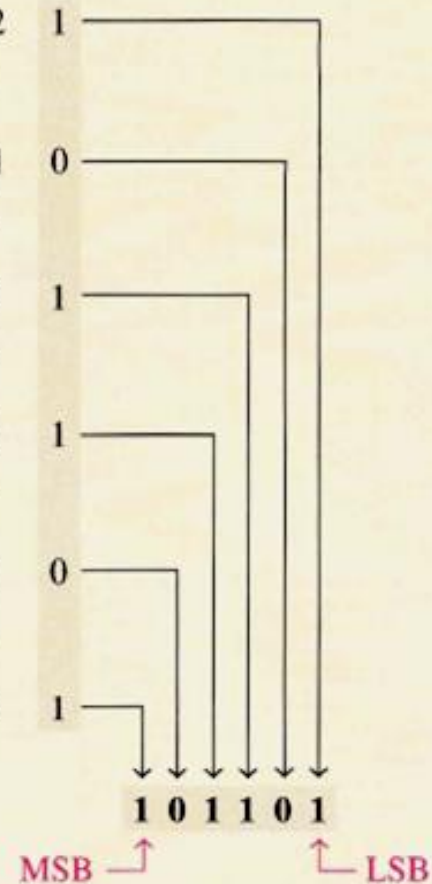
$$\begin{array}{r} 19 \\ \div 2 = 9 \\ \downarrow \\ 9 \\ \div 2 = 4 \\ \downarrow \\ 4 \\ \div 2 = 2 \\ \downarrow \\ 2 \\ \div 2 = 1 \\ \downarrow \\ 1 \\ \div 2 = 0 \end{array}$$



(b)

Remainder

$$\begin{array}{r} 45 \\ \div 2 = 22 \\ \downarrow \\ 22 \\ \div 2 = 11 \\ \downarrow \\ 11 \\ \div 2 = 5 \\ \downarrow \\ 5 \\ \div 2 = 2 \\ \downarrow \\ 2 \\ \div 2 = 1 \\ \downarrow \\ 1 \\ \div 2 = 0 \end{array}$$

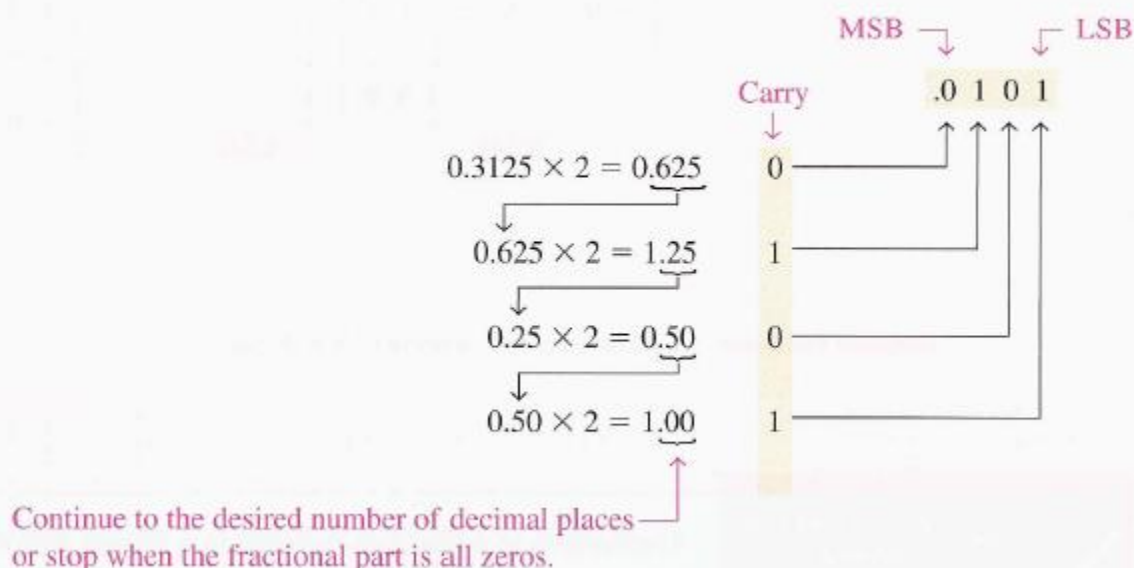


*Related Problem* Convert decimal number 39 to binary.



## Converting Decimal Fractions to Binary

**Repeated Multiplication by 2** As you have seen, decimal whole numbers can be converted to binary by repeated division by 2. Decimal fractions can be converted to binary by repeated multiplication by 2. For example, to convert the decimal fraction 0.3125 to binary, begin by multiplying 0.3125 by 2 and then multiplying each resulting fractional part of the product by 2 until the fractional product is zero or until the desired number of decimal places is reached. The carry digits, or **carries**, generated by the multiplications produce the binary number. The first carry produced is the MSB, and the last carry is the LSB. This procedure is illustrated as follows:



### SECTION 2-3 REVIEW

1. Convert each decimal number to binary by using the sum-of-weights method:  
(a) 23      (b) 57      (c) 45.5
2. Convert each decimal number to binary by using the repeated division-by-2 method (repeated multiplication-by-2 for fractions):  
(a) 14      (b) 21      (c) 0.375

# BINARY ARITHMETIC

## Binary Addition

The four basic rules for adding binary digits (bits) are as follows:

$$0 + 0 = 0 \quad \text{Sum of 0 with a carry of 0}$$

$$0 + 1 = 1 \quad \text{Sum of 1 with a carry of 0}$$

$$1 + 0 = 1 \quad \text{Sum of 1 with a carry of 0}$$

$$1 + 1 = 10 \quad \text{Sum of 0 with a carry of 1}$$

Remember, in binary  $1 + 1 = 10$ , not 2.

Notice that the first three rules result in a single bit and in the fourth rule the addition of two 1s yields a binary two (10). When binary numbers are added, the last condition creates a sum of 0 in a given column and a carry of 1 over to the next column to the left, as illustrated in the following addition of  $11 + 1$ :

$$\begin{array}{r} \text{Carry} \quad \text{Carry} \\ 1 \quad 1 \\ 0 \quad 1 \quad 1 \\ + 0 \quad 0 \quad 1 \\ \hline 1 \quad 0 \quad 0 \end{array}$$

## EXAMPLE 2-7

Add the following binary numbers:

- (a)  $11 + 11$       (b)  $100 + 10$       (c)  $111 + 11$       (d)  $110 + 100$

**Solution** The equivalent decimal addition is also shown for reference.

(a)	$\begin{array}{r} 11 \\ +11 \\ \hline 110 \end{array}$	$\begin{array}{r} 3 \\ +3 \\ \hline 6 \end{array}$	(b)	$\begin{array}{r} 100 \\ +10 \\ \hline 110 \end{array}$	$\begin{array}{r} 4 \\ +2 \\ \hline 6 \end{array}$	(c)	$\begin{array}{r} 111 \\ +11 \\ \hline 1010 \end{array}$	$\begin{array}{r} 7 \\ +3 \\ \hline 10 \end{array}$	(d)	$\begin{array}{r} 110 \\ +100 \\ \hline 1010 \end{array}$	$\begin{array}{r} 6 \\ +4 \\ \hline 10 \end{array}$
-----	--	--	-----	---	--	-----	--	---	-----	---	---

**Related Problem** Add 1111 and 1100.

## Binary Subtraction

The four basic rules for subtracting bits are as follows:

$$\begin{array}{l} 0 - 0 = 0 \\ 1 - 1 = 0 \\ 1 - 0 = 1 \\ 10 - 1 = 1 \quad 0 - 1 \text{ with a borrow of } 1 \end{array}$$

Remember in binary  $10 - 1 = 1$ , not 9.

When subtracting numbers, you sometimes have to borrow from the next column to the left. A borrow is required in binary only when you try to subtract a 1 from a 0. In this case, when a 1 is borrowed from the next column to the left, a 10 is created in the column being subtracted, and the last of the four basic rules just listed must be applied. Examples 2–8 and 2–9 illustrate binary subtraction; the equivalent decimal subtractions are also shown.

### EXAMPLE 2–8

Perform the following binary subtractions:

(a)  $11 - 01$       (b)  $11 - 10$

*Solution*

(a)	$\begin{array}{r} 11 \\ - 01 \\ \hline 10 \end{array}$	$\begin{array}{r} 3 \\ - 1 \\ \hline 2 \end{array}$	(b)	$\begin{array}{r} 11 \\ - 10 \\ \hline 01 \end{array}$	$\begin{array}{r} 3 \\ - 2 \\ \hline 1 \end{array}$
-----	--	---	-----	--	---

No borrows were required in this example. The binary number 01 is the same as 1.

*Related Problem* Subtract 100 from 111.



### EXAMPLE 2-9

Subtract 011 from 101.

*Solution*

$$\begin{array}{r} 101 \\ -011 \\ \hline 010 \end{array}$$

Let's examine exactly what was done to subtract the two binary numbers since a borrow is required. Begin with the right column.

Left column:

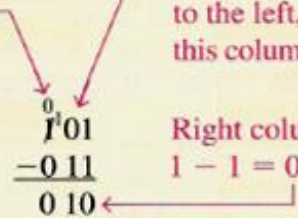
When a 1 is borrowed,  
a 0 is left, so  $0 - 0 = 0$ .

Middle column:

Borrow 1 from next column  
to the left, making a 10 in  
this column, then  $10 - 1 = 1$ .

Right column:

$$1 - 1 = 0$$


$$\begin{array}{r} 0101 \\ -011 \\ \hline 010 \end{array}$$

*Related Problem* Subtract 101 from 110.

## Binary Multiplication

Binary multiplication of two bits is the same as multiplication of the decimal digits 0 and 1.

The four basic rules for multiplying bits are as follows:

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

Multiplication is performed with binary numbers in the same manner as with decimal numbers. It involves forming partial products, shifting each successive partial product left one place, and then adding all the partial products. Example 2–10 illustrates the procedure; the equivalent decimal multiplications are shown for reference.

### EXAMPLE 2–10

Perform the following binary multiplications:

(a)  $11 \times 11$       (b)  $101 \times 111$

*Solution*

(a)

	11	3
	$\times 11$	$\times 3$
Partial products	$\left\{ \begin{array}{r} 11 \\ +11 \\ \hline 1001 \end{array} \right.$	$\left\{ \begin{array}{r} 9 \end{array} \right.$

(b)

	111	7
	$\times 101$	$\times 5$
Partial products	$\left\{ \begin{array}{r} 111 \\ 000 \\ +111 \\ \hline 100011 \end{array} \right.$	$\left\{ \begin{array}{r} 35 \end{array} \right.$

*Related Problem* Multiply  $1101 \times 1010$ .

## Binary Division

Division in binary follows the same procedure as division in decimal, as Example 2-11 illustrates. The equivalent decimal divisions are also given.

A calculator can be used to perform arithmetic operations with binary numbers as long as the capacity of the calculator is not exceeded.

### EXAMPLE 2-11

Perform the following binary divisions:

(a)  $110 \div 11$       (b)  $110 \div 10$

*Solution*

(a)	$\begin{array}{r} 10 \\ 11 \overline{)110} \\ \underline{11} \phantom{0} \\ 000 \end{array}$	$\begin{array}{r} 2 \\ 3 \overline{)6} \\ \underline{6} \\ 0 \end{array}$	(b)	$\begin{array}{r} 11 \\ 10 \overline{)110} \\ \underline{10} \phantom{0} \\ 10 \\ \underline{10} \\ 00 \end{array}$	$\begin{array}{r} 3 \\ 2 \overline{)6} \\ \underline{6} \\ 0 \end{array}$
-----	--	---	-----	---	---

*Related Problem* Divide 1100 by 100.

## SECTION 2-4 REVIEW

1. Perform the following binary additions:  
(a)  $1101 + 1010$       (b)  $10111 + 01101$
2. Perform the following binary subtractions:  
(a)  $1101 - 0100$       (b)  $1001 - 0111$
3. Perform the indicated binary operations:  
(a)  $110 \times 111$       (b)  $1100 \div 011$



## 2-5 1'S AND 2'S COMPLEMENTS OF BINARY NUMBERS

The 1's complement and the 2's complement of a binary number are important because they permit the representation of negative numbers. The method of 2's complement arithmetic is commonly used in computers to handle negative numbers.

### Finding the 1's Complement

Change each bit in a number to get the 1's complement.

The 1's **complement** of a binary number is found by changing all 1s to 0s and all 0s to 1s, as illustrated below:

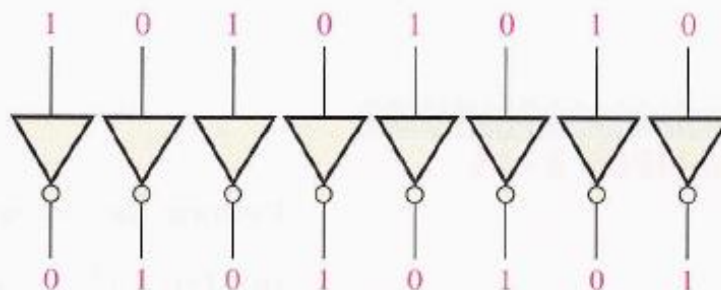
1	0	1	1	0	0	1	0
↓	↓	↓	↓	↓	↓	↓	↓
0	1	0	0	1	1	0	1

Binary number                      1's complement

The simplest way to obtain the 1's complement of a binary number with a digital circuit is to use parallel inverters (NOT circuits), as shown in Figure 2-2 for an 8-bit binary number.

► **FIGURE 2-2**

Example of inverters used to obtain the 1's complement of a binary number.



## Finding the 2's Complement

Add 1 to the 1's complement to get the 2's complement.

The 2's complement of a binary number is found by adding 1 to the LSB of the 1's complement.

$$\text{2's complement} = (\text{1's complement}) + 1$$

### EXAMPLE 2-12

Find the 2's complement of 10110010.

*Solution*

10110010	Binary number
01001101	1's complement
$\begin{array}{r} + \phantom{00000000} \\ \hline \phantom{00000000} 1 \end{array}$	Add 1
<b>01001110</b>	2's complement

*Related Problem* Determine the 2's complement of 11001011.

## The Sign Bit

The left-most bit in a signed binary number is the **sign bit**, which tells you whether the number is positive or negative.

A 0 sign bit indicates a positive number, and a 1 sign bit indicates a negative number.

### SECTION 2-5 REVIEW

- Determine the 1's complement of each binary number:  
(a) 00011010      (b) 11110111      (c) 10001101
- Determine the 2's complement of each binary number:  
(a) 00010110      (b) 11111100      (c) 10010001

The **hexadecimal** number system has a base of sixteen; that is, it is composed of 16 **numeric** and alphabetic **characters**. Most digital systems process binary data in groups that are multiples of four bits, making the hexadecimal number very convenient because each hexadecimal digit represents a 4-bit binary number (as listed in Table 2-3).

The hexadecimal number system consists of digits 0-9 and letters A-F.

TABLE 2-3

DECIMAL	BINARY	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F



## Counting in Hexadecimal

How do you count in hexadecimal once you get to F? Simply start over with another column and continue as follows:

10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 2A, 2B, 2C, 2D, 2E, 2F, 30, 31, . . .

With two hexadecimal digits, you can count up to  $FF_{16}$ , which is decimal 255. To count beyond this, three hexadecimal digits are needed. For instance,  $100_{16}$  is decimal 256,  $101_{16}$  is decimal 257, and so forth. The maximum 3-digit hexadecimal number is  $FFF_{16}$ , or decimal 4095. The maximum 4-digit hexadecimal number is  $FFFF_{16}$ , which is decimal 65,535.

## Binary-to-Hexadecimal Conversion

Converting a binary number to hexadecimal is a straightforward procedure. Simply break the binary number into 4-bit groups, starting at the right-most bit and replace each 4-bit group with the equivalent hexadecimal symbol.

### EXAMPLE 2-24

Convert the following binary numbers to hexadecimal:

(a) 1100101001010111      (b) 111111000101101001

*Solution*

(a)  $\overbrace{1100} \quad \overbrace{1010} \quad \overbrace{0101} \quad \overbrace{0111}$   
    ↓     ↓     ↓     ↓  
    C    A    5    7    =  $CA57_{16}$

(b)  $\overbrace{0011} \quad \overbrace{1111} \quad \overbrace{0001} \quad \overbrace{0110} \quad \overbrace{1001}$   
    ↓     ↓     ↓     ↓     ↓  
    3    F    1    6    9    =  $3F169_{16}$

Two zeros have been added in part (b) to complete a 4-bit group at the left.

*Related Problem* Convert the binary number 1001111011110011100 to hexadecimal.

## Hexadecimal-to-Binary Conversion

Hexadecimal is a convenient way to represent binary numbers.

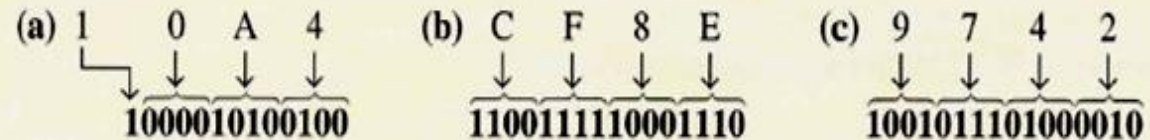
To convert from a hexadecimal number to a binary number, reverse the process and replace each hexadecimal symbol with the appropriate four bits.

### EXAMPLE 2-25

Determine the binary numbers for the following hexadecimal numbers:

- (a)  $10A4_{16}$       (b)  $CF8E_{16}$       (c)  $9742_{16}$

*Solution*



In part (a), the MSB is understood to have three zeros preceding it, thus forming a 4-bit group.

*Related Problem* Convert the hexadecimal number 6BD3 to binary.

It should be clear that it is much easier to deal with a hexadecimal number than with the equivalent binary number. Since conversion is so easy, the hexadecimal system is widely used for representing binary numbers in programming, printouts, and displays.

*Conversion between hexadecimal and binary is direct and easy.*



## Hexadecimal-to-Decimal Conversion

### EXAMPLE 2-26

Convert the following hexadecimal numbers to decimal:

- (a)  $1C_{16}$       (b)  $A85_{16}$

**Solution** Remember, convert the hexadecimal number to binary first, then to decimal.

$$\begin{array}{cc} \text{(a)} & \begin{array}{cc} 1 & C \\ \downarrow & \downarrow \\ 0001 & 1100 \end{array} \\ & \overline{00011100} = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28}_{10} \end{array}$$

$$\begin{array}{ccc} \text{(b)} & \begin{array}{ccc} A & 8 & 5 \\ \downarrow & \downarrow & \downarrow \\ 1010 & 1000 & 0101 \end{array} \\ & \overline{101010000101} = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693}_{10} \end{array}$$

### EXAMPLE 2-27

Convert the following hexadecimal numbers to decimal:

- (a)  $E5_{16}$       (b)  $B2F8_{16}$

$16^3$	$16^2$	$16^1$	$16^0$
4096	256	16	1

**Solution** Recall from Table 2-3 that letters A through F represent decimal numbers 10 through 15, respectively.

$$\text{(a)} \quad E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229}_{10}$$

$$\begin{aligned} \text{(b)} \quad B2F8_{16} &= (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1) \\ &= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1) \\ &= 45,056 + 512 + 240 + 8 = \mathbf{45,816}_{10} \end{aligned}$$



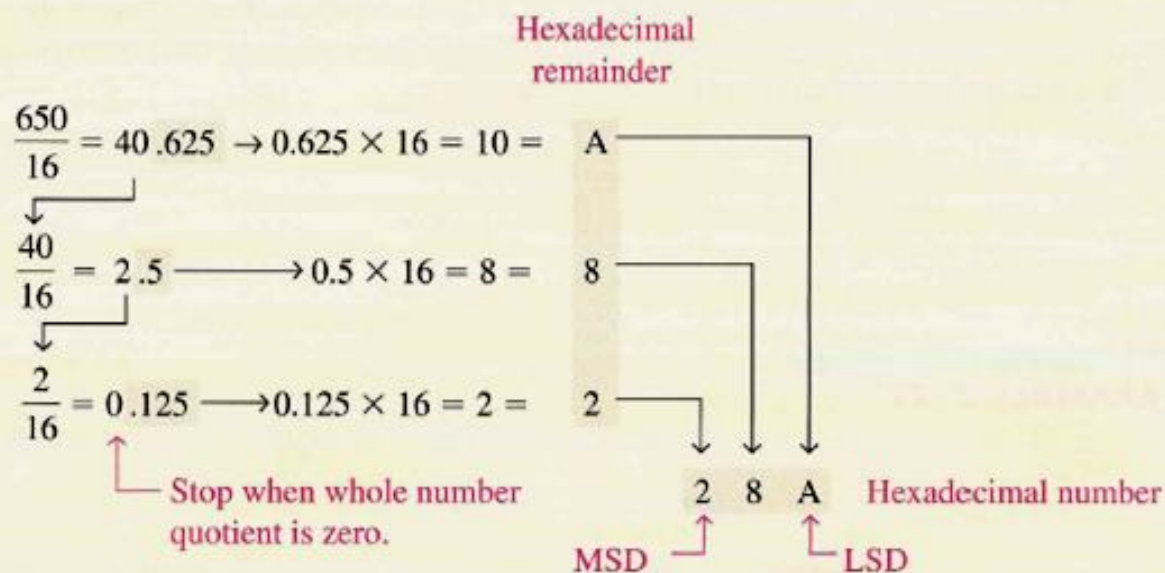
## Decimal-to-Hexadecimal Conversion

Repeated division of a decimal number by 16 will produce the equivalent hexadecimal number, formed by the remainders of the divisions. The first remainder produced is the least significant digit (LSD). Each successive division by 16 yields a remainder that becomes a digit in the equivalent hexadecimal number. This procedure is similar to repeated division by 2 for decimal-to-binary conversion that was covered in Section 2-3. Example 2-28 illustrates the procedure. Note that when a quotient has a fractional part, the fractional part is multiplied by the divisor to get the remainder.

### EXAMPLE 2-28

Convert the decimal number 650 to hexadecimal by repeated division by 16.

*Solution*



**Related Problem** Convert decimal 2591 to hexadecimal.

## 2-9 OCTAL NUMBERS

Like the hexadecimal number system, the octal number system provides a convenient way to express binary numbers and codes. However, it is used less frequently than hexadecimal in conjunction with computers and microprocessors to express binary quantities for input and output purposes.

The **octal** number system is composed of eight digits, which are

0, 1, 2, 3, 4, 5, 6, 7

To count above 7, begin another column and start over:

10, 11, 12, 13, 14, 15, 16, 17, 20, 21, . . .

Counting in octal is similar to counting in decimal, except that the digits 8 and 9 are not used. To distinguish octal numbers from decimal numbers or hexadecimal numbers, we will use the subscript 8 to indicate an octal number. For instance,  $15_8$  in octal is equivalent to  $13_{10}$  in decimal and D in hexadecimal. Sometimes you may see an "o" or a "Q" following an octal number.

### Octal-to-Decimal Conversion

Since the octal number system has a base of eight, each successive digit position is an increasing power of eight, beginning in the right-most column with  $8^0$ . The evaluation of an octal number in terms of its decimal equivalent is accomplished by multiplying each digit by its weight and summing the products, as illustrated here for  $2374_8$ .

Weight:  $8^3$   $8^2$   $8^1$   $8^0$

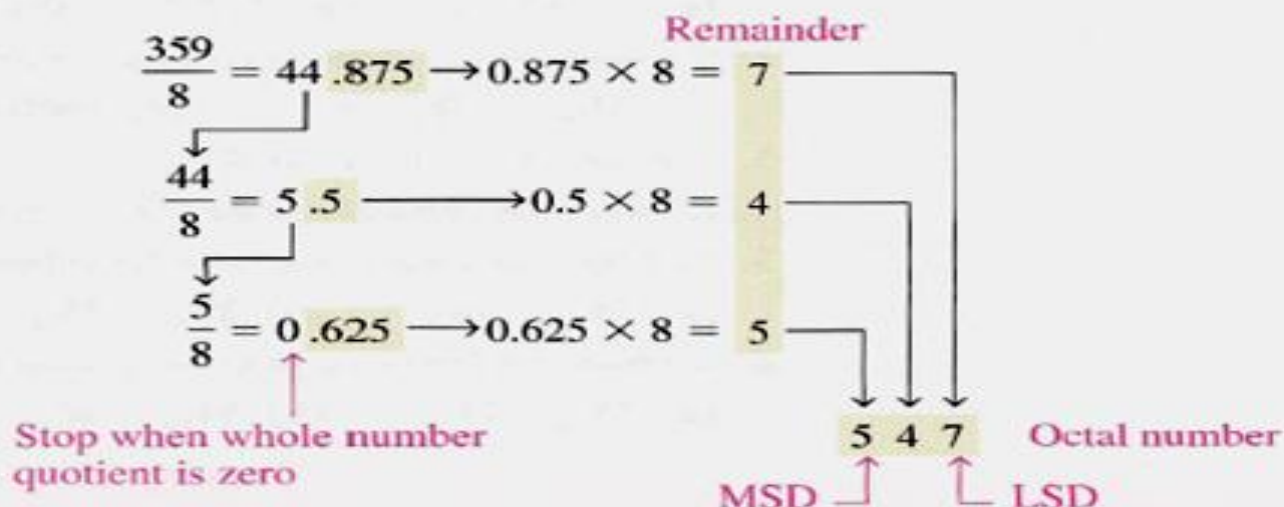
Octal number: 2 3 7 4

$$\begin{aligned} 2374_8 &= (2 \times 8^3) + (3 \times 8^2) + (7 \times 8^1) + (4 \times 8^0) \\ &= (2 \times 512) + (3 \times 64) + (7 \times 8) + (4 \times 1) \\ &= 1024 + 192 + 56 + 4 = 1276_{10} \end{aligned}$$



## Decimal-to-Octal Conversion

A method of converting a decimal number to an octal number is the repeated division-by-8 method, which is similar to the method used in the conversion of decimal numbers to binary or to hexadecimal. To show how it works, let's convert the decimal number 359 to octal. Each successive division by 8 yields a remainder that becomes a digit in the equivalent octal number. The first remainder generated is the least significant digit (LSD).



## Octal-to-Binary Conversion

Because each octal digit can be represented by a 3-bit binary number, it is very easy to convert from octal to binary. Each octal digit is represented by three bits as shown in Table 2-4.

TABLE 2-4

Octal/binary conversion.

OCTAL DIGIT	0	1	2	3	4	5	6	7
BINARY	000	001	010	011	100	101	110	111



To convert an octal number to a binary number, simply replace each octal digit with the appropriate three bits.

### EXAMPLE 2-31

Convert each of the following octal numbers to binary:

- (a)  $13_8$       (b)  $25_8$       (c)  $140_8$       (d)  $7526_8$

*Solution*

(a)	$\begin{array}{cc} 1 & 3 \\ \downarrow & \downarrow \\ \overline{001} & \overline{011} \end{array}$	(b)	$\begin{array}{cc} 2 & 5 \\ \downarrow & \downarrow \\ \overline{010} & \overline{101} \end{array}$	(c)	$\begin{array}{ccc} 1 & 4 & 0 \\ \downarrow & \downarrow & \downarrow \\ \overline{001} & \overline{100} & \overline{000} \end{array}$	(d)	$\begin{array}{cccc} 7 & 5 & 2 & 6 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ \overline{111} & \overline{101} & \overline{010} & \overline{110} \end{array}$
-----	---	-----	---	-----	--	-----	---

### Binary-to-Octal Conversion

#### EXAMPLE 2-32

Convert each of the following binary numbers to octal:

- (a)  $110101$       (b)  $101111001$       (c)  $100110011010$       (d)  $11010000100$

*Solution*

(a)	$\begin{array}{cc} \overline{110} & \overline{101} \\ \downarrow & \downarrow \\ 6 & 5 = 65_8 \end{array}$	(b)	$\begin{array}{ccc} \overline{101} & \overline{111} & \overline{001} \\ \downarrow & \downarrow & \downarrow \\ 5 & 7 & 1 = 571_8 \end{array}$
(c)	$\begin{array}{cccc} \overline{100} & \overline{110} & \overline{011} & \overline{010} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 4 & 6 & 3 & 2 = 4632_8 \end{array}$	(d)	$\begin{array}{cccc} \overline{011} & \overline{010} & \overline{000} & \overline{0100} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 3 & 2 & 0 & 4 = 3204_8 \end{array}$

## SECTION 2-9

### REVIEW

1. Convert the following octal numbers to decimal:  
(a)  $73_8$       (b)  $125_8$
2. Convert the following decimal numbers to octal:  
(a)  $98_{10}$       (b)  $163_{10}$
3. Convert the following octal numbers to binary:  
(a)  $46_8$       (b)  $723_8$       (c)  $5624_8$
4. Convert the following binary numbers to octal:  
(a)  $110101111$       (b)  $1001100010$       (c)  $10111111001$

## 2-10 BINARY CODED DECIMAL (BCD)

Binary coded decimal (BCD) is a way to express each of the decimal digits with a binary code. There are only ten code groups in the BCD system, so it is very easy to convert between decimal and BCD. Because we like to read and write in decimal, the BCD code provides an excellent interface to binary systems. Examples of such interfaces are keypad inputs and digital readouts.

To express any decimal number in BCD, simply replace each decimal digit with the appropriate 4-bit code, as shown by Example 2-33.

### EXAMPLE 2-33

Convert each of the following decimal numbers to BCD:

- (a) 35      (b) 98      (c) 170      (d) 2469

*Solution*

(a)    3    5  
      ↓    ↓  
      00110101

(b)    9    8  
      ↓    ↓  
      10011000

(c)    1    7    0  
      ↓    ↓    ↓  
      000101110000

(d)    2    4    6    9  
      ↓    ↓    ↓    ↓  
      0010010001101001

*Related Problem* Convert the decimal number 9673 to BCD.



It is equally easy to determine a decimal number from a BCD number. Start at the rightmost bit and break the code into groups of four bits. Then write the decimal digit represented by each 4-bit group.

### EXAMPLE 2-34

Convert each of the following BCD codes to decimal:

(a) 10000110

(b) 001101010001

(c) 1001010001110000

*Solution*

(a)  $\begin{array}{c} 10000110 \\ \hline \downarrow \quad \downarrow \\ 8 \quad 6 \end{array}$

(b)  $\begin{array}{c} 001101010001 \\ \hline \downarrow \quad \downarrow \quad \downarrow \\ 3 \quad 5 \quad 1 \end{array}$

(c)  $\begin{array}{c} 1001010001110000 \\ \hline \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ 9 \quad 4 \quad 7 \quad 0 \end{array}$

*Related Problem* Convert the BCD code 10000010001001110110 to decimal.

## ASCII

**ASCII** is the abbreviation for American Standard Code for Information Interchange. Pronounced “askee,” ASCII is a universally accepted alphanumeric code used in most computers and other electronic equipment. Most computer keyboards are standardized with the ASCII. When you enter a letter, a number, or control command, the corresponding ASCII code goes into the computer.

ASCII has 128 characters and symbols represented by a 7-bit binary code. Actually, ASCII can be considered an 8-bit code with the MSB always 0. This 8-bit code is 00 through 7F in hexadecimal. The first thirty-two ASCII characters are nongraphic commands that are never printed or displayed and are used only for control purposes. Examples of the control characters are “null,” “line feed,” “start of text,” and “escape.” The other characters are graphic symbols that can be printed or displayed and include the letters of the alphabet (lowercase and uppercase), the ten decimal digits, punctuation signs and other commonly used symbols.



CONTROL CHARACTERS				GRAPHIC SYMBOLS											
NAME	DEC	BINARY	HEX	SYMBOL	DEC	BINARY	HEX	SYMBOL	DEC	BINARY	HEX	SYMBOL	DEC	BINARY	HEX
NUL	0	0000000	00	space	32	0100000	20	@	64	1000000	40	`	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	\$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	-	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C		124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F