Vietnam National University HCMC

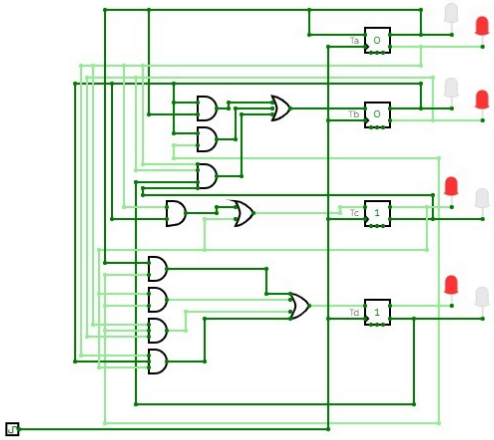
International University

School of Electrical Engineering

EE053IU

Digital Logic Design

Lecture 2: Number Systems, Operations, and Codes



INSTRUCTOR: Dr. Vuong Quoc Bao

Lecture Objectives

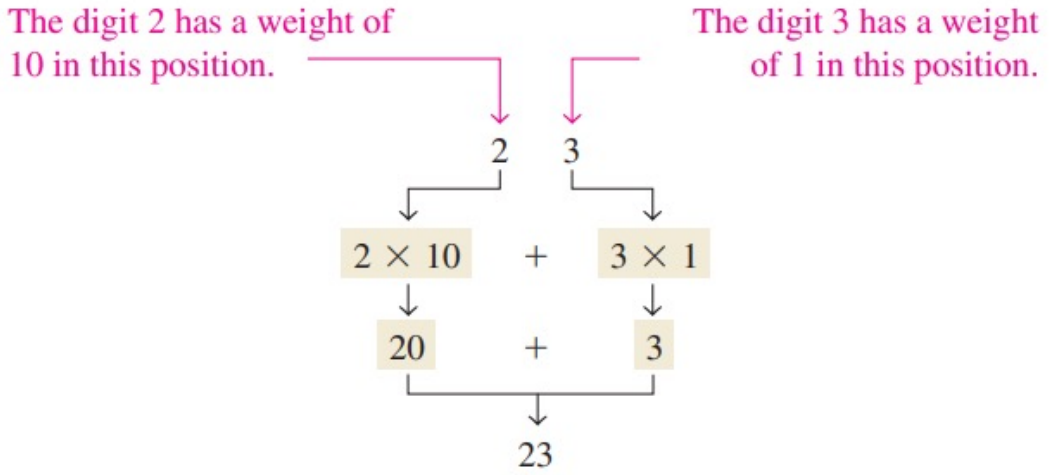
- Review the decimal number system
- Count in the binary number system
- Convert from decimal to binary and from binary to decimal
- Apply arithmetic operations to binary number
- Determine the 1's and 2's complements of a binary number
- Express signed binary numbers in sign-magnitude, 1's complement, 2's complement, and floating-point format
- Carry out arithmetic operations with signed binary numbers

Lecture Objectives

- Convert between the binary and hexadecimal number systems
- Add numbers in hexadecimal form
- Convert between the binary and octal number systems
- Express decimal numbers in binary coded decimal (BCD) form
- Add BCD numbers
- Convert between the binary system and the Gray code
- Interpret the American Standard Code for Information Interchange (ASCII)

I. Decimal Numbers

- In the decimal number system each of the ten digits, 0 through 9, represents a certain quantity.



- The decimal number system has a base of 10.

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

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

$$10^2\ 10^1\ 10^0.10^{-1}\ 10^{-2}\ 10^{-3}\ \ldots$$

Decimal point

EXAMPLE 2–1

Express the decimal number 47 as a sum of the values of each digit.

Solution

The digit 4 has a weight of 10, which is 10^1, as indicated by its position. The digit 7 has a weight of 1, which is 10^0, as indicated by its position.

$$\begin{aligned} 47 &= (4 \times 10^1) + (7 \times 10^0) \\ &= (4 \times 10) + (7 \times 1) = \mathbf{40 + 7} \end{aligned}$$

Related Problem*

Determine the value of each digit in 939.

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}.

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

Related Problem

Determine the value of each digit in 67.924.

II. Binary Numbers

- The binary number system has two digits (bits).
- The decimal number system has a base of 2.

Counting in Binary

- In general, with n bits you can count up to a number equal to $2^n - 1$.
- Largest decimal number = $2^n - 1$

TABLE 2–1

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

An Application

Learning to count in binary will help you to basically understand how digital circuits can be used to count events. Let's take a simple example of counting tennis balls going into a box from a conveyor belt. Assume that nine balls are to go into each box.

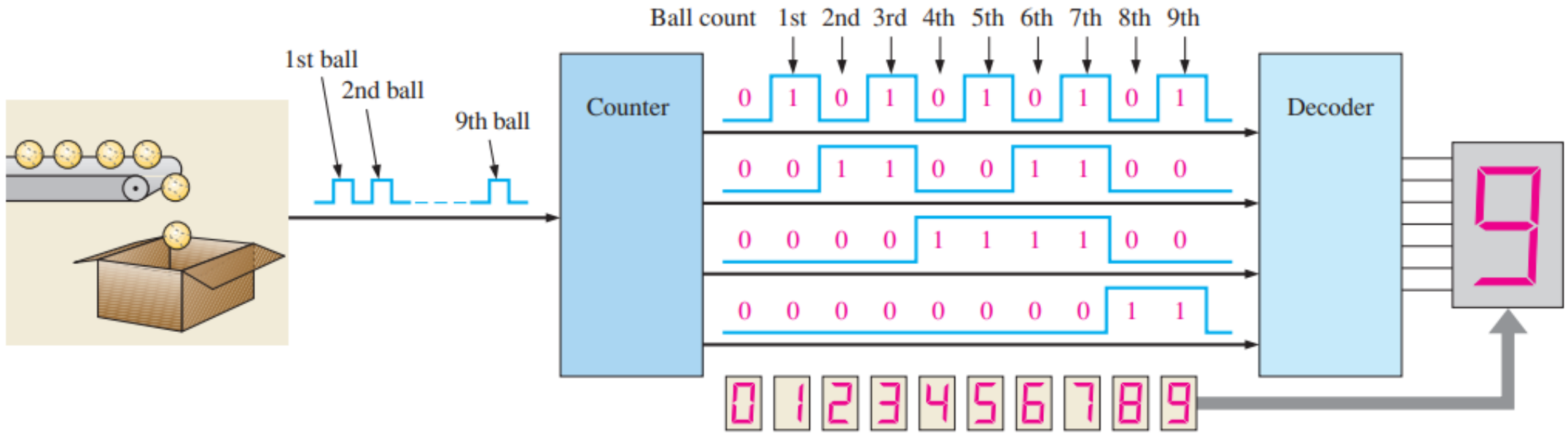


FIGURE 2–1 Illustration of a simple binary counting application.

The Weighting Structure of Binary Numbers

- A binary number is a weighted number. The right-most bit is the **LSB (least significant bit)** in a binary whole number and has a weight of $2^0 = 1$.
- The weights increase from right to left by a power of two for each bit. The left-most bit is the **MSB (most significant bit);** its weight depends on the size of the binary number.
- ***The weight structure of a binary number is***

$$2^{n-1} \ldots 2^3\ 2^2\ 2^1\ 2^0\ .\ 2^{-1}\ 2^{-2} \ldots 2^{-n}$$

Binary point

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 0.5	1/4 0.25	1/8 0.125	1/16 0.625	1/32 0.03125	1/64 0.015625

Binary-to-Decimal Conversion

- 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.

$$\text{Weight:} \quad 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$$

$$\text{Binary number:} \quad 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1$$

$$1101101 = 2^6 + 2^5 + 2^3 + 2^2 + 2^0$$

$$= 64 + 32 + 8 + 4 + 1 = \mathbf{109}$$

Related Problem

Convert the binary number 10010001 to decimal.

Binary-to-Decimal Conversion

- 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 2–4

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.

$$\begin{array}{rcccc} \text{Weight:} & 2^{-1} & 2^{-2} & 2^{-3} & 2^{-4} \\ \text{Binary number:} & 0\,.\,1 & 0 & 1 & 1 \end{array}$$

$$0.1011 = 2^{-1} + 2^{-3} + 2^{-4}$$

$$= 0.5 + 0.125 + 0.0625 = \mathbf{0.6875}$$

Related Problem

Convert the binary number 10.111 to decimal.

3. Decimal-to-Binary Conversion

Sum-of-Weights Method

- To get the binary number for a given decimal number, find the binary weights that add up to the decimal number.
- The decimal number 9, for example, can be expressed as the sum of binary weights as follows:

$$9 = 8 + 1 \quad \text{or} \quad 9 = 2^3 + 2^0$$

- Placing 1s in the appropriate weight positions, 2^3 and 2^0, and 0s in the 2^2 and 2^1 positions determines the binary number for decimal 9.

2^3	2^2	2^1	2^0	
1	0	0	1	Binary number for decimal 9

Sum-of-Weights Method

EXAMPLE 2–5

Convert the following decimal numbers to binary:

(a) 12 **(b)** 25

(c) 58 **(d)** 82

Solution

(a) $12 = 8 + 4 = 2^3 + 2^2 \longrightarrow$ **1100**

(b) $25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 \longrightarrow$ **11001**

(c) $58 = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1 \longrightarrow$ **111010**

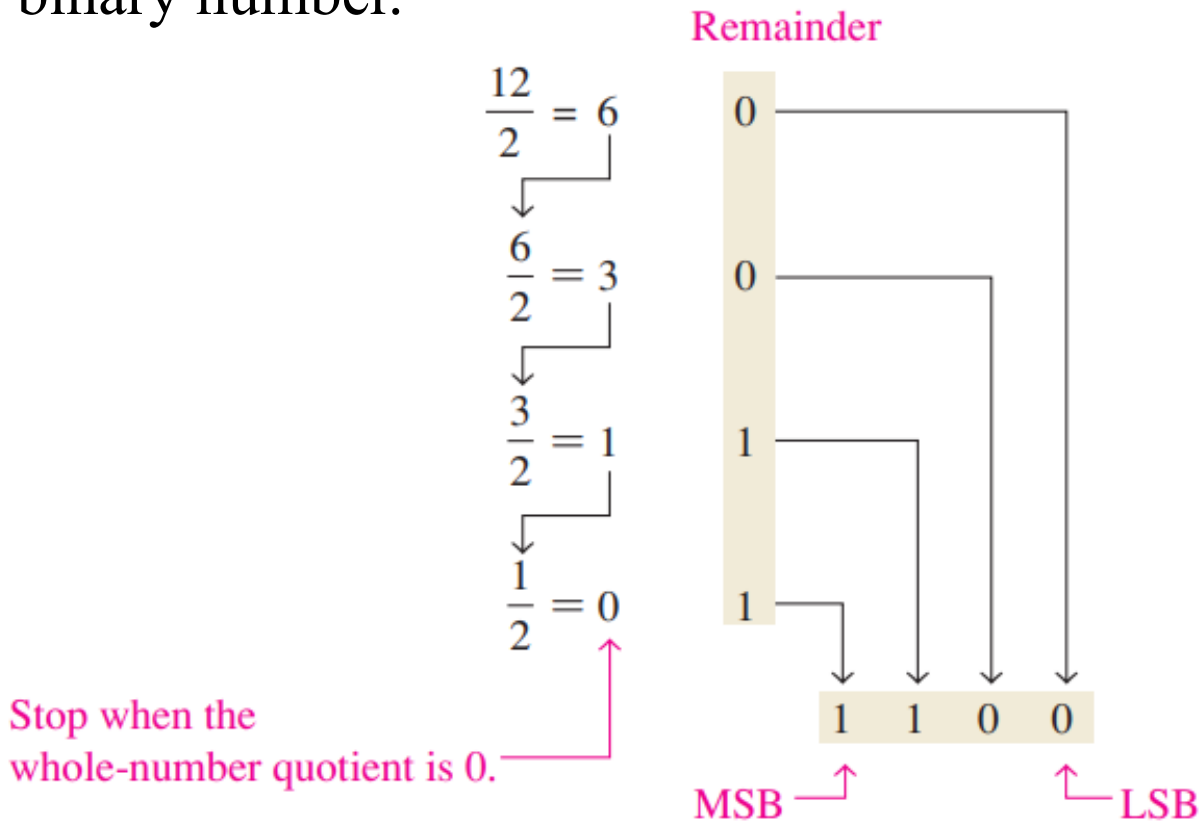
(d) $82 = 64 + 16 + 2 = 2^6 + 2^4 + 2^1 \longrightarrow$ **1010010**

Related Problem

Convert the decimal number 125 to binary.

Repeated Division-by-2 Method

- A systematic method of converting whole numbers from decimal to binary is the repeated division-by-2 process.
- 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)**.



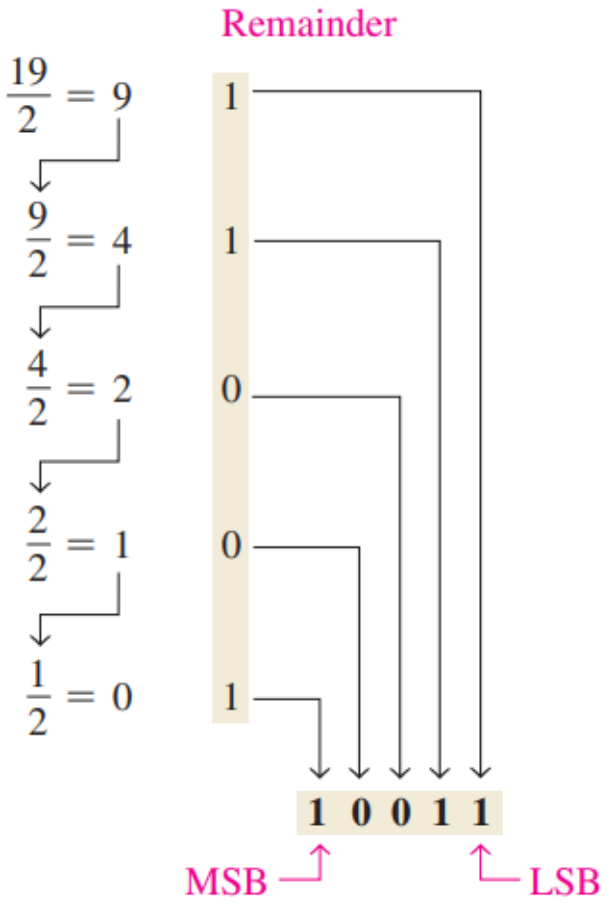
EXAMPLE 2–6

Convert the following decimal numbers to binary:

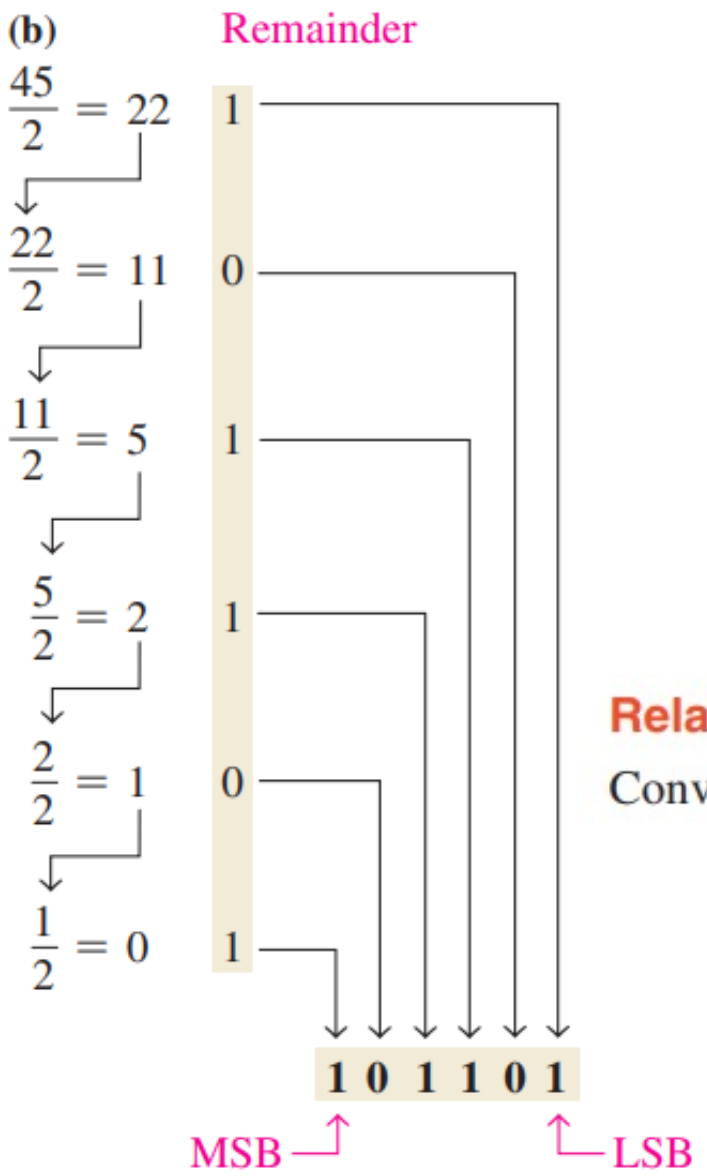
(a) 19 **(b)** 45

Solution

(a)



(b)



Related Problem

Convert decimal number 39 to binary.

Converting Decimal Fractions to Binary

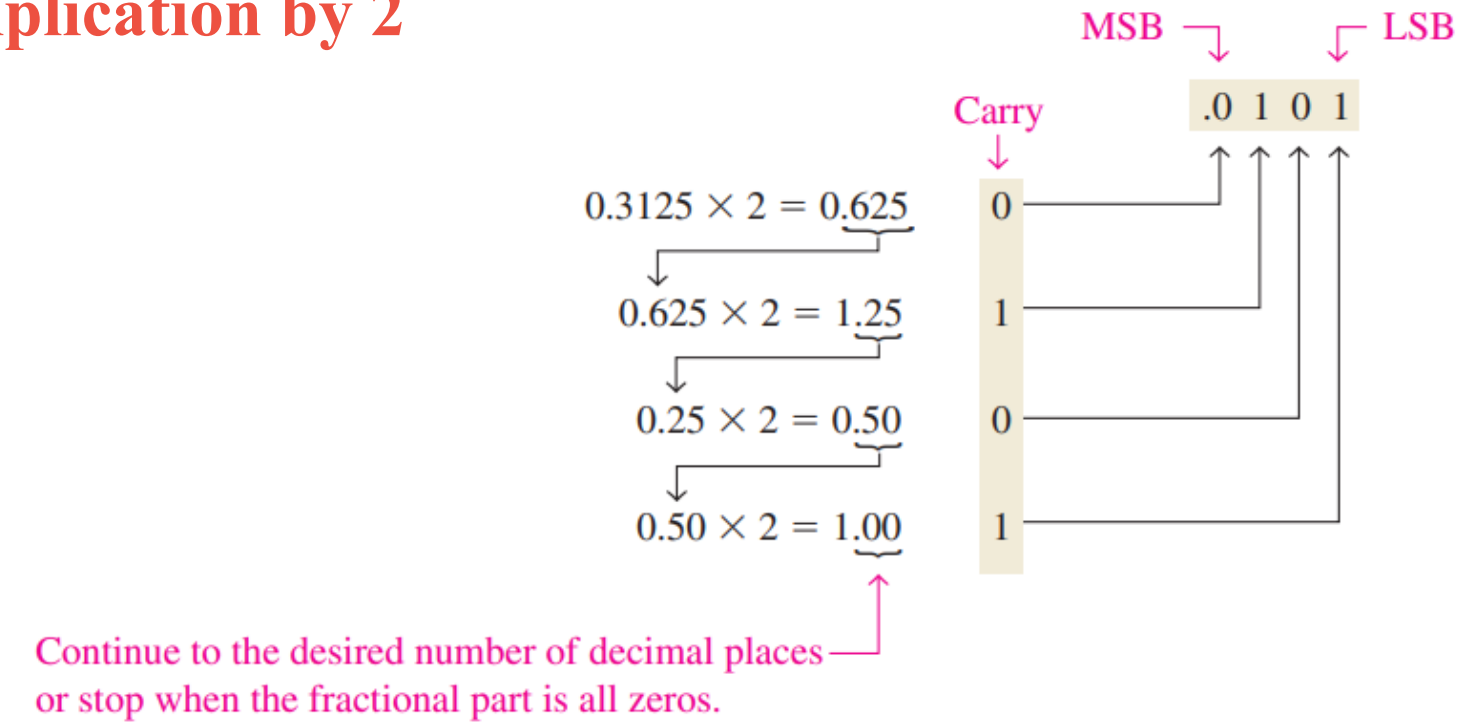
Sum-of-Weights

- The sum-of-weights method can be applied to fractional decimal numbers, as shown in the following example:

$$0.625 = 0.5 + 0.125 = 2^{-1} + 2^{-3} = 0.101$$

- There is a 1 in the 2^{-1} position, a 0 in the 2^{-2} position, and a 1 in the 2^{-3} position.

Repeated Multiplication by 2

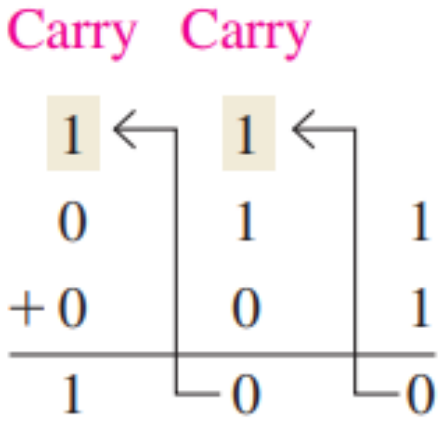


4. Binary Arithmetic

Binary Addition

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

0 + 0 = 0	Sum of 0 with a carry of 0
0 + 1 = 1	Sum of 1 with a carry of 0
1 + 0 = 1	Sum of 1 with a carry of 0
1 + 1 = 10	Sum of 0 with a carry of 1



Carry bits

1	+ 0 + 0 = 01	Sum of 1 with a carry of 0
1	+ 1 + 0 = 10	Sum of 0 with a carry of 1
1	+ 0 + 1 = 10	Sum of 0 with a carry of 1
1	+ 1 + 1 = 11	Sum of 1 with a carry of 1

In binary 1 + 1 = 10, not 2.

Binary Addition

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)

$$\begin{array}{r} 11 \\ +\,11 \\ \hline \mathbf{110} \end{array} \qquad \begin{array}{r} 3 \\ +\,3 \\ \hline 6 \end{array}$$

(b)

$$\begin{array}{r} 100 \\ +\,10 \\ \hline \mathbf{110} \end{array} \qquad \begin{array}{r} 4 \\ +\,2 \\ \hline 6 \end{array}$$

(c)

$$\begin{array}{r} 111 \\ +\,11 \\ \hline \mathbf{1010} \end{array} \qquad \begin{array}{r} 7 \\ +\,3 \\ \hline 10 \end{array}$$

(d)

$$\begin{array}{r} 110 \\ +\,100 \\ \hline \mathbf{1010} \end{array} \qquad \begin{array}{r} 6 \\ +\,4 \\ \hline 10 \end{array}$$

Related Problem

Add 1111 and 1100.

Binary Subtraction

- The four basic rules for subtracting bits are as follows:

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$10 - 1 = 1 \qquad 0 - 1 \text{ with a borrow of } 1$$

- 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.

In binary 10 - 1 = 1, not 9.

Binary Subtraction

EXAMPLE 2–8

Perform the following binary subtractions:

(a) 11 − 01 **(b)** 11 − 10

Solution

(a)	11	3	**(b)**	11	3
	−01	−1		−10	−2
	10	2		**01**	1

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

101	5
−011	−3
010	2

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

$$\begin{array}{r} {\scriptstyle 0\ 1\ 1} \\ 1\,01 \\ -0\,11 \\ \hline 0\,10 \end{array}$$

Related Problem

Subtract 101 from 110.

Binary Multiplication

- 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.

Binary Multiplication

EXAMPLE 2–10

Perform the following binary multiplications:

(a) 11×11 **(b)** 101×111

Solution

(a)

$$\begin{array}{r} 11 \\ \times\ 11 \\ \hline 11 \\ +11 \\ \hline \mathbf{1001} \end{array} \qquad \begin{array}{r} 3 \\ \times\ 3 \\ \hline 9 \end{array}$$

Partial products

(b)

$$\begin{array}{r} 111 \\ \times\ 101 \\ \hline 111 \\ 000 \\ +111 \\ \hline \mathbf{100011} \end{array} \qquad \begin{array}{r} 7 \\ \times\ 5 \\ \hline 35 \end{array}$$

Partial products

Related Problem

Multiply 1101×1010.

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.

EXAMPLE 2–11

Perform the following binary divisions:

(a) 110 ÷ 11 **(b)** 110 ÷ 10

Solution

(a)

$$\begin{array}{r} \mathbf{10} \\ 11\overline{)110} \\ \underline{11} \\ 000 \end{array} \qquad \begin{array}{r} 2 \\ 3\overline{)6} \\ \underline{6} \\ 0 \end{array}$$

(b)

$$\begin{array}{r} \mathbf{11} \\ 10\overline{)110} \\ \underline{10} \\ 10 \\ \underline{10} \\ 00 \end{array} \qquad \begin{array}{r} 3 \\ 2\overline{)6} \\ \underline{6} \\ 0 \end{array}$$

Related Problem

Divide 1100 by 100.

5. Complements of Binary Numbers

Finding 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	Binary number
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	
0 1 0 0 1 1 0 1	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)

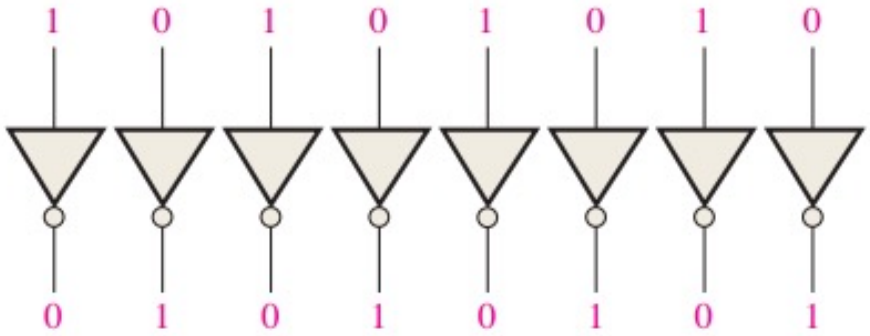


FIGURE 2–2 Example of inverters used to obtain the 1's complement of a binary number.

Finding 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
+	1	Add 1
	01001110	2's complement

Related Problem

Determine the 2's complement of 11001011.

Finding the 2's Complement

An alternative method of finding the 2's complement of a binary number is as follows:

- Start at the right with the LSB and write the bits as they are up to and including the first 1.
- Take the 1's complements of the remaining bits.

EXAMPLE 2–13

Find the 2's complement of 10111000 using the alternative method.

Solution

10111000 Binary number

01001000 2's complement

1's complements of original bits

These bits stay the same.

Related Problem

Find the 2's complement of 11000000.

Finding the 2's Complement

- The 2's complement of a negative binary number can be realized using inverters and an adder.
- This illustrates how an 8-bit number can be converted to its 2's complement by first inverting each bit (taking the 1's complement) and then adding 1 to the 1's complement with an adder circuit.

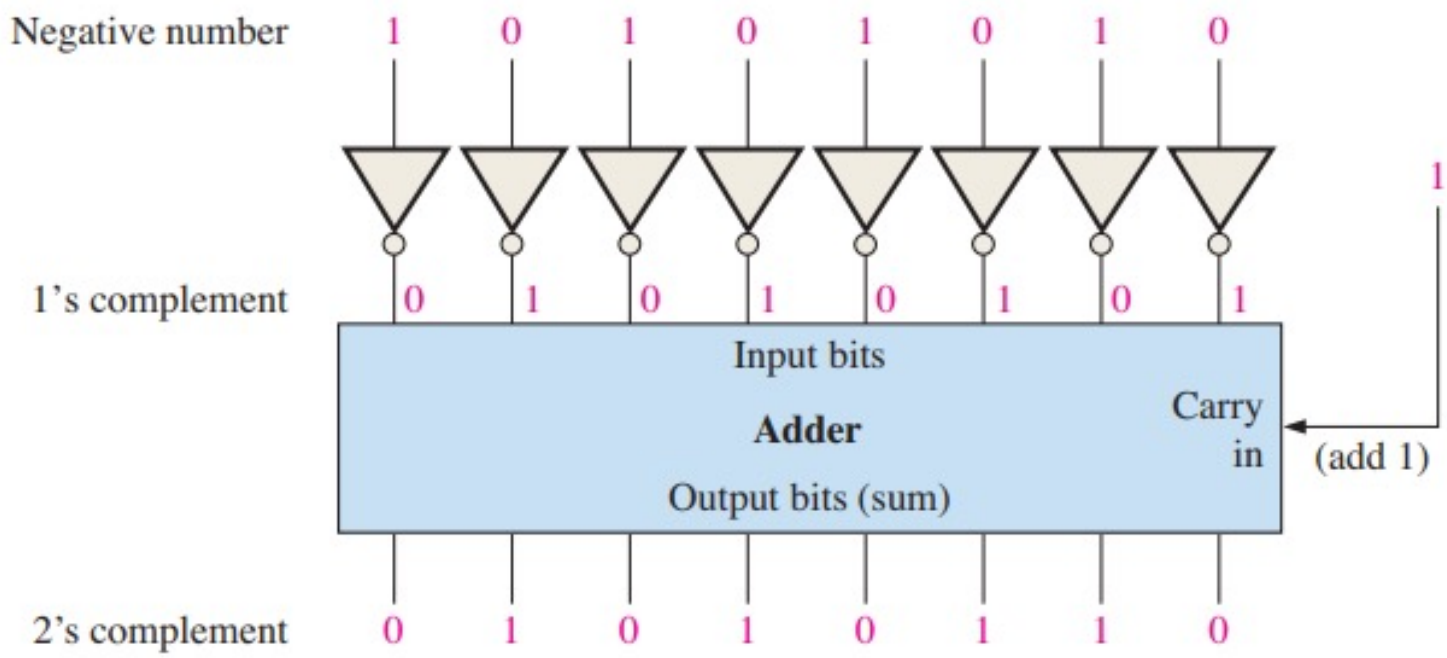


FIGURE 2–3 Example of obtaining the 2's complement of a negative binary number.

6. Signed Numbers

- Digital systems, such as the computer, must be able to handle both positive and negative numbers.
- A signed binary number consists of both sign and magnitude information.

The Sign Bit

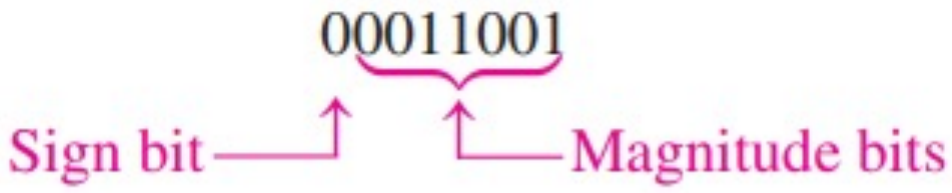
The left-most bit in a signed binary number is the **sign bit**, which tells you whether the number is positive or negative.

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

Sign-Magnitude Form

When a signed binary number is represented in sign-magnitude, the left-most bit is the sign bit and the remaining bits are the magnitude bits.

For example, the decimal number +25 is expressed as an 8-bit signed binary number using the sign-magnitude form as



The decimal number −25 is expressed as

10011001

In the sign-magnitude form, a negative number has the same magnitude bits as the corresponding positive number but the sign bit is a 1 rather than a zero.

1’s Complement Form

- Positive numbers in 1’s complement form are represented the same way as the positive sign-magnitude numbers.
- Negative numbers, however, are the 1’s complements of the corresponding positive numbers. For example, using eight bits, the decimal number 225 is expressed as the 1’s complement of +25 (00011001) as 11100110

In the 1’s complement form, a negative number is the 1’s complement of the corresponding positive number

2's Complement Form

- Positive numbers in 2's complement form are represented the same way as in the sign magnitude and 1's complement forms.
- Negative numbers are the 2's complements of the corresponding positive numbers. Again, using eight bits, let's take decimal number 225 and express it as the 2's complement of +25 (00011001).
- Inverting each bit and adding 1, you get -25 = 11100111

In the 2's complement form, a negative number is the 2's complement of the corresponding positive number

EXAMPLE 2–14

Express the decimal number −39 as an 8-bit number in the sign-magnitude, 1's complement, and 2's complement forms.

Solution

First, write the 8-bit number for +39.

00100111

In the *sign-magnitude form,* −39 is produced by changing the sign bit to a 1 and leaving the magnitude bits as they are. The number is

10100111

In the *1's complement form,* −39 is produced by taking the 1's complement of +39 (00100111).

11011000

In the *2's complement form,* −39 is produced by taking the 2's complement of +39 (00100111) as follows:

```
 11011000    1's complement
+       1
----------
 11011001    2's complement
```

Related Problem

Express +19 and −19 as 8-bit numbers in sign-magnitude, 1's complement, and 2's complement.

The Decimal Value of Signed Numbers

Sign-magnitude

- Decimal values of positive and negative numbers in the sign-magnitude form are determined by summing the weights in all the magnitude bit positions where there are 1s and ignoring those positions where there are zeros.
- The sign is determined by examination of the sign bit.

EXAMPLE 2–15

Determine the decimal value of this signed binary number expressed in sign-magnitude: 10010101.

Solution

The seven magnitude bits and their powers-of-two weights are as follows:

2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	1	0	1	0	1

Summing the weights where there are 1s,

$$16 + 4 + 1 = 21$$

The sign bit is 1; therefore, the decimal number is **−21**.

Related Problem

Determine the decimal value of the sign-magnitude number 01110111.

1's Complement

- Decimal values of positive numbers in the 1's complement form are determined by summing the weights in all bit positions where there are 1s and ignoring those positions where there are zeros.

- Decimal values of negative numbers are determined by assigning a negative value to the weight of the sign bit, summing all the weights where there are 1s, and adding 1 to the result.

EXAMPLE 2–16

Determine the decimal values of the signed binary numbers expressed in 1's complement:

(a) 00010111 **(b)** 11101000

Solution

(a) The bits and their powers-of-two weights for the positive number are as follows:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	1	0	1	1	1

Summing the weights where there are 1s,

$$16 + 4 + 2 + 1 = \mathbf{+23}$$

(b) The bits and their powers-of-two weights for the negative number are as follows. Notice that the negative sign bit has a weight of -2^7 or -128.

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	0	1	0	0	0

Summing the weights where there are 1s,

$$-128 + 64 + 32 + 8 = -24$$

Adding 1 to the result, the final decimal number is

$$-24 + 1 = \mathbf{-23}$$

Related Problem

Determine the decimal value of the 1's complement number 11101011.

2's Complement

- Decimal values of positive and negative numbers in the 2's complement form are determined by summing the weights in all bit positions where there are 1s and ignoring those positions where there are zeros.

- The weight of the sign bit in a negative number is given a negative value.

EXAMPLE 2–17

Determine the decimal values of the signed binary numbers expressed in 2's complement:

(a) 01010110 **(b)** 10101010

Solution

(a) The bits and their powers-of-two weights for the positive number are as follows:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	0	1	1	0

Summing the weights where there are 1s,

$$64 + 16 + 4 + 2 = \mathbf{+86}$$

(b) The bits and their powers-of-two weights for the negative number are as follows. Notice that the negative sign bit has a weight of $-2^7 = -128$.

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	0

Summing the weights where there are 1s,

$$-128 + 32 + 8 + 2 = \mathbf{-86}$$

Related Problem

Determine the decimal value of the 2's complement number 11010111.

Range of Signed Integer Numbers

- We have used 8-bit numbers for illustration because the 8-bit grouping is common in most computers and has been given the special name byte. With one byte or eight bits, you can represent 256 different numbers. With two bytes or sixteen bits, you can represent 65,536 different numbers. The formula for finding the number of different combinations of n bits is:

$$\text{Total combinations} = 2^n$$

- For 2's complement signed numbers, the range of values for n-bit numbers is

$$\text{Range} = -(2^{n-1}) \text{ to } +(2^{n-1} - 1)$$

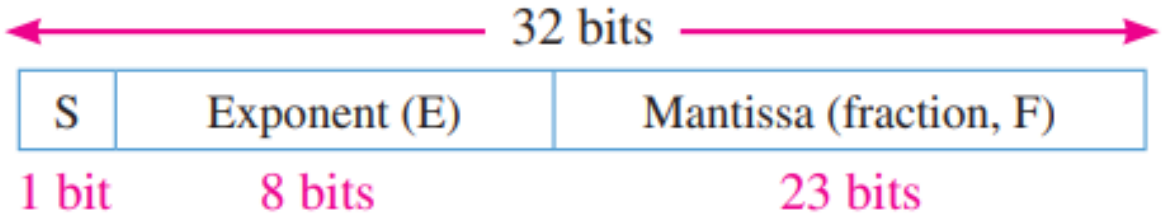
The range of magnitude values represented by binary numbers depends on the number of bits (n).

Floating-Point Numbers

- The floating-point number system, based on scientific notation, is capable of representing very large and very small numbers without an increase in the number of bits and also for representing numbers that have both integer and fractional components.
- A floating-point number (also known as a real number) consists of two parts plus a sign. The **mantissa** is the part of a floating-point number that represents the magnitude of the number and is between 0 and 1. The **exponent** is the part of a floating-point number that represents the number of places that the decimal point (or binary point) is to be moved.

Single-Precision Floating-Point Binary Numbers

- In the standard format for a single-precision binary number, the sign bit (S) is the left-most bit, the exponent (E) includes the next eight bits, and the mantissa or fractional part (F) includes the remaining 23 bits, as shown next.



- The general approach to determining the value of a floating-point number is expressed by the following formula:

$$\text{Number} = (-1)^S(1 + F)(2^{E-127})$$

- To illustrate, let's consider the following floating-point binary number:

S	E	F
1	10010001	10001110001000000000000

- The sign bit is 1. The biased exponent is 10010001 = 145. Applying the formula, we get

$$\begin{aligned}\text{Number} &= (-1)^1\,(1.10001110001)(2^{145-127}) \\ &= (-1)(1.10001110001)(2^{18}) = -1100011100010000000\end{aligned}$$

Single-Precision Floating-Point Binary Numbers

EXAMPLE 2–18

Convert the decimal number 3.248×10^4 to a single-precision floating-point binary number.

Solution

Convert the decimal number to binary.

$$3.248 \times 10^4 = 32480 = 1111111011100000_2 = 1.11111011100000 \times 2^{14}$$

The MSB will not occupy a bit position because it is always a 1. Therefore, the mantissa is the fractional 23-bit binary number 11111011100000000000000 and the biased exponent is

$$14 + 127 = 141 = 10001101_2$$

The complete floating-point number is

0	**10001101**	**11111011100000000000000**

Related Problem

Determine the binary value of the following floating-point binary number:

0 10011000 10000100010100110000000

7. Arithmetic Operations with Signed Numbers

Addition

The two numbers in an addition are the addend and the augend. The result is the sum.

There are four cases that can occur when two signed binary numbers are added.

- Both numbers positive
- Positive number with magnitude larger than negative number
- Negative number with magnitude larger than positive number
- Both numbers negative

Both numbers positive:

00000111	7
+ 00000100	+ 4
00001011	11

The sum is positive and is therefore in true (uncomplemented) binary.

Addition of two positive numbers yields a positive number.

Positive number with magnitude larger than negative number:

	00001111	15
	+ 11111010	+ −6
Discard carry →	1 00001001	9

The final carry bit is discarded. The sum is positive and therefore in true (uncomplemented) binary.

Addition of a positive number and a smaller negative number yields a positive number.

Negative number with magnitude larger than positive number:

00010000	16
+ 11101000	+ −24
11111000	−8

The sum is negative and therefore in 2's complement form.

Addition of a positive number and a larger negative number or two negative numbers yields a negative number in 2's complement.

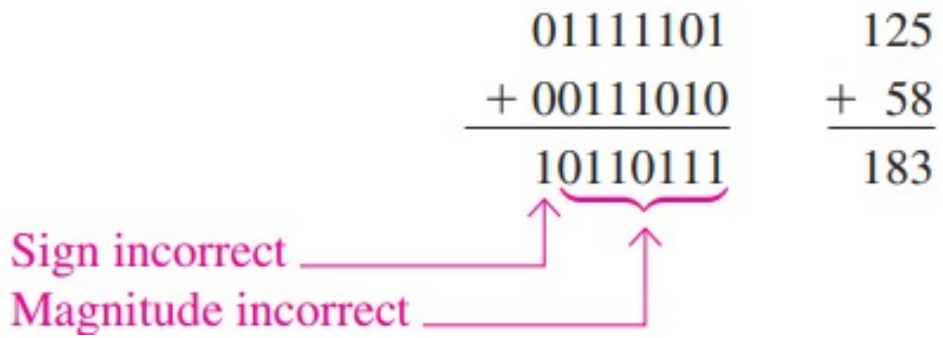
Both numbers negative:

	11111011	−5
	+ 11110111	+ −9
Discard carry →	1 11110010	−14

The final carry bit is discarded. The sum is negative and therefore in 2's complement form.

Overflow Condition:

- When two numbers are added and the number of bits required to represent the sum exceeds the number of bits in the two numbers, an overflow results as indicated by an incorrect sign bit.
- An overflow can occur only when both numbers are positive or both numbers are negative. If the sign bit of the result is different than the sign bit of the numbers that are added, overflow is indicated.



Numbers Added Two at a Time

Now let's look at the addition of a string of numbers, added two at a time. This can be accomplished by adding the first two numbers, then adding the third number to the sum of the first two, then adding the fourth number to this result, and so on. This is how computers add strings of numbers.

Numbers Added Two at a Time

EXAMPLE 2–19

Add the signed numbers: 01000100, 00011011, 00001110, and 00010010.

Solution

The equivalent decimal additions are given for reference.

68	01000100	
+ 27	+ 00011011	Add 1st two numbers
95	01011111	1st sum
+ 14	+ 00001110	Add 3rd number
109	01101101	2nd sum
+ 18	+ 00010010	Add 4th number
127	**01111111**	Final sum

Related Problem

Add 00110011, 10111111, and 01100011. These are signed numbers.

Subtraction

- Subtraction is a special case of addition. For example, subtracting +6 (the subtrahend) from +9 (the minuend) is equivalent to adding 26 to +9. Basically, the subtraction operation changes the sign of the subtrahend and adds it to the minuend.

- The result of a subtraction is called the difference.

The sign of a positive or negative binary number is changed by taking its 2's complement.

To subtract two signed numbers, take the 2's complement of the subtrahend and add. Discard any final carry bit.

EXAMPLE 2–20

Perform each of the following subtractions of the signed numbers:

(a) 00001000 − 00000011 **(b)** 00001100 − 11110111

(c) 11100111 − 00010011 **(d)** 10001000 − 11100010

Solution

Like in other examples, the equivalent decimal subtractions are given for reference.

(a) In this case, $8 - 3 = 8 + (-3) = 5$.

	00001000	Minuend (+8)
	+ 11111101	2's complement of subtrahend (−3)
Discard carry ⟶	**1 00000101**	Difference (+5)

(b) In this case, $12 - (-9) = 12 + 9 = 21$.

00001100	Minuend (+12)
+ 00001001	2's complement of subtrahend (+9)
00010101	Difference (+21)

(c) In this case, $-25 - (+19) = -25 + (-19) = -44$.

	11100111	Minuend (−25)
	+ 11101101	2's complement of subtrahend (−19)
Discard carry	**1 11010100**	Difference (−44)

(d) In this case, $-120 - (-30) = -120 + 30 = -90$.

10001000	Minuend (−120)
+ 00011110	2's complement of subtrahend (+30)
10100110	Difference (−90)

Related Problem

Subtract 01000111 from 01011000.

Multiplication

- The numbers in a multiplication are the multiplicand, the multiplier, and the product.
- These are illustrated in the following decimal multiplication:

8	Multiplicand
× 3	Multiplier
24	Product

- Direct addition and partial products are two basic methods for performing multiplication using addition.

The direct addition method

- In the direct addition method, you add the multiplicand a number of times equal to the multiplier

The direct addition method

EXAMPLE 2–21

Multiply the signed binary numbers: 01001101 (multiplicand) and 00000100 (multiplier) using the direct addition method.

Solution

Since both numbers are positive, they are in true form, and the product will be positive. The decimal value of the multiplier is 4, so the multiplicand is added to itself four times as follows:

01001101	1st time
+ 01001101	2nd time
10011010	Partial sum
+ 01001101	3rd time
11100111	Partial sum
+ 01001101	4th time
100110100	Product

Since the sign bit of the multiplicand is 0, it has no effect on the outcome. All of the bits in the product are magnitude bits.

Related Problem

Multiply 01100001 by 00000110 using the direct addition method.

The partial products method

- The partial products method is perhaps the more common one because it reflects the way you multiply longhand.
- The multiplicand is multiplied by each multiplier digit beginning with the least significant digit.
- The result of the multiplication of the multiplicand by a multiplier digit is called a partial product.
- Each successive partial product is moved (shifted) one place to the left and when all the partial products have been produced, they are added to get the final product.
- Here is a decimal example

239	Multiplicand
× 123	Multiplier
717	1st partial product (3 × 239)
478	2nd partial product (2 × 239)
+ 239	3rd partial product (1 × 239)
29,397	Final product

- The sign of the product of a multiplication depends on the signs of the multiplicand and the multiplier according to the following two rules:
  - ✓ *If the signs are the same, the product is positive.*
  - ✓ *If the signs are different, the product is negative.*
- The basic steps in the partial products method of binary multiplication are as follows:
  - ✓ **Step 1:** Determine if the signs of the multiplicand and multiplier are the same or different. This determines what the sign of the product will be.
  - ✓ **Step 2:** Change any negative number to true (uncomplemented) form. Because most computers store negative numbers in 2's complement, a 2's complement operation is required to get the negative number into true form.
  - ✓ **Step 3:** Starting with the least significant multiplier bit, generate the partial products. When the multiplier bit is 1, the partial product is the same as the multiplicand. When the multiplier bit is 0, the partial product is zero. Shift each successive partial product one bit to the left.
  - ✓ **Step 4:** Add each successive partial product to the sum of the previous partial products to get the final product.
  - ✓ **Step 5:** If the sign bit that was determined in step 1 is negative, take the 2's complement of the product. If positive, leave the product in true form. Attach the sign bit to the product.

EXAMPLE 2–22

Multiply the signed binary numbers: 01010011 (multiplicand) and 11000101 (multiplier).

Solution

Step 1: The sign bit of the multiplicand is 0 and the sign bit of the multiplier is 1. The sign bit of the product will be 1 (negative).

Step 2: Take the 2's complement of the multiplier to put it in true form.

$$11000101 \longrightarrow 00111011$$

Step 3 and 4: The multiplication proceeds as follows. Notice that only the magnitude bits are used in these steps.

1010011	Multiplicand
× 0111011	Multiplier
1010011	1st partial product
+ 1010011	2nd partial product
11111001	Sum of 1st and 2nd
+ 0000000	3rd partial product
011111001	Sum
+ 1010011	4th partial product
1110010001	Sum
+ 1010011	5th partial product
100011000001	Sum
+ 1010011	6th partial product
1001100100001	Sum
+ 0000000	7th partial product
1001100100001	Final product

Step 5: Since the sign of the product is a 1 as determined in step 1, take the 2's complement of the product.

$$1001100100001 \longrightarrow 0110011011111$$

Attach the sign bit →

1 0110011011111

Related Problem

Verify the multiplication is correct by converting to decimal numbers and performing the multiplication.

Division

- The numbers in a division are the dividend, the divisor, and the quotient. These are illustrated in the following standard division format.

$$\frac{\text{dividend}}{\text{divisor}} = \text{quotient}$$

- The result of a division is called the quotient; the quotient is the number of times that the divisor will go into the dividend. This means that the divisor can be subtracted from the dividend a number of times equal to the quotient, as illustrated by dividing 21 by 7.

21	Dividend
− 7	1st subtraction of divisor
14	1st partial remainder
− 7	2nd subtraction of divisor
7	2nd partial remainder
− 7	3rd subtraction of divisor
0	Zero remainder

- The sign of the quotient depends on the signs of the dividend and the divisor according to the following two rules:
  - ✓ *If the signs are the same, the quotient is positive.*
  - ✓ *If the signs are different, the quotient is negative.*
- When two binary numbers are divided, both numbers must be in true (uncomplemented) form. The basic steps in a division process are as follows:
  - ✓ **Step 1:** Determine if the signs of the dividend and divisor are the same or different. This determines what the sign of the quotient will be. Initialize the quotient to zero.
  - ✓ **Step 2:** Subtract the divisor from the dividend using 2's complement addition to get the first partial remainder and add 1 to the quotient. If this partial remainder is positive, go to step 3. If the partial remainder is zero or negative, the division is complete.
  - ✓ **Step 3:** Subtract the divisor from the partial remainder and add 1 to the quotient. If the result is positive, repeat for the next partial remainder. If the result is zero or negative, the division is complete.

EXAMPLE 2–23

Divide 01100100 by 00011001.

Solution

Step 1: The signs of both numbers are positive, so the quotient will be positive. The quotient is initially zero: 00000000.

Step 2: Subtract the divisor from the dividend using 2's complement addition (remember that final carries are discarded).

01100100	Dividend
+ 11100111	2's complement of divisor
01001011	Positive 1st partial remainder

Add 1 to quotient: 00000000 + 00000001 = 00000001.

Step 3: Subtract the divisor from the 1st partial remainder using 2's complement addition.

01001011	1st partial remainder
+ 11100111	2's complement of divisor
00110010	Positive 2nd partial remainder

Add 1 to quotient: 00000001 + 00000001 = 00000010.

Step 4: Subtract the divisor from the 2nd partial remainder using 2's complement addition.

00110010	2nd partial remainder
+ 11100111	2's complement of divisor
00011001	Positive 3rd partial remainder

Add 1 to quotient: 00000010 + 00000001 = 00000011.

Step 5: Subtract the divisor from the 3rd partial remainder using 2's complement addition.

00011001	3rd partial remainder
+ 11100111	2's complement of divisor
00000000	Zero remainder

Add 1 to quotient: 00000011 + 00000001 = **00000100** (final quotient). The process is complete.

Related Problem

Verify that the process is correct by converting to decimal numbers and performing the division.

8. Hexadecimal Numbers

- The hexadecimal number system has sixteen characters; it is used primarily as a compact way of displaying or writing binary numbers because it is very easy to convert between binary and hexadecimal.

- 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

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

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)** 1111110001011010 01

Solution

(a) 1100 1010 0101 0111

C A 5 7 $= \mathbf{CA57}_{16}$

(b) 0011 1111 0001 0110 1001

3 F 1 6 9 $= \mathbf{3F169}_{16}$

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

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

(a) 1 0 A 4 → **1000010100100**

(b) C F 8 E → **1100111110001110**

(c) 9 7 4 2 → **1001011101000010**

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.

Hexadecimal-to-Decimal Conversion

One way to find the decimal equivalent of a hexadecimal number is to first convert the hexadecimal number to binary and then convert from binary to decimal.

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.

(a)

$$\overbrace{0001}^{1}\overbrace{1100}^{C} = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28}_{10}$$

(b)

$$\overbrace{1010}^{A}\overbrace{1000}^{8}\overbrace{0101}^{5} = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693}_{10}$$

Related Problem

Convert the hexadecimal number 6BD to decimal.

Hexadecimal-to-Decimal Conversion

Another way to convert a hexadecimal number to its decimal equivalent is to multiply the decimal value of each hexadecimal digit by its weight and then take the sum of these products. The weights of a hexadecimal number are increasing powers of 16 (from right to left).

EXAMPLE 2–27

Convert the following hexadecimal numbers to decimal:

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

Solution

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

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

(b) $B2F8_{16} = (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1)$

$= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1)$

$= 45{,}056 + 512 + 240 + 8 = \mathbf{45{,}816}_{10}$

Related Problem

Convert $60A_{16}$ to decimal.

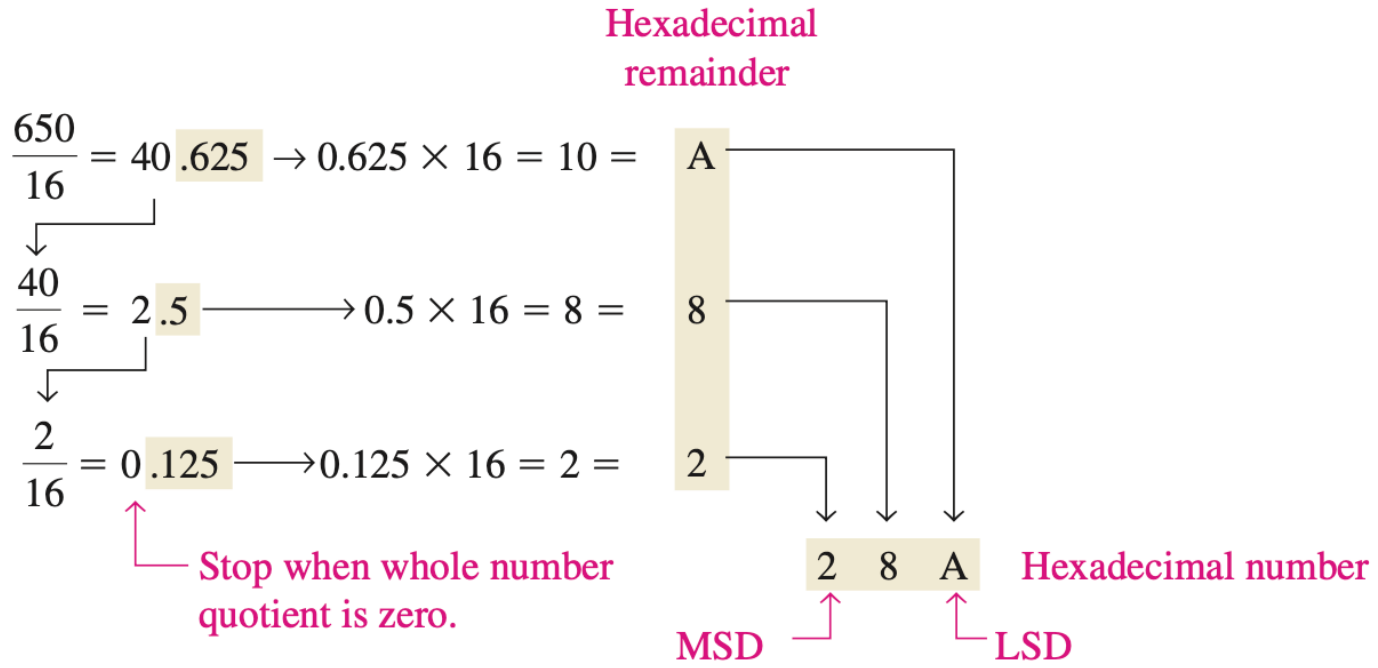
Decimal-to-Hexadecimal Conversion

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.

EXAMPLE 2–28

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

Solution



Related Problem

Convert decimal 2591 to hexadecimal.

Hexadecimal Addition

Addition can be done directly with hexadecimal numbers by remembering that the hexadecimal digits 0 through 9 are equivalent to decimal digits 0 through 9 and that hexadecimal digits A through F are equivalent to decimal numbers 10 through 15. When adding two hexadecimal numbers, use the following rules.

- In any given column of an addition problem, think of the two hexadecimal digits in terms of their decimal values. For instance, $5_{16} = 5_{10}$ and $C_{16} = 12_{10}$.
- If the sum of these two digits is 15_{10} or less, bring down the corresponding hexadecimal digit.
- Ifthesumofthesetwodigitsisgreaterthan1510,bringdowntheamountofthesum that exceeds 16_{10} and carry a 1 to the next column.

EXAMPLE 2–29

Add the following hexadecimal numbers:

(a) $23_{16} + 16_{16}$ **(b)** $58_{16} + 22_{16}$ **(c)** $2B_{16} + 84_{16}$ **(d)** $DF_{16} + AC_{16}$

Solution

(a)
$$\begin{array}{r} 23_{16} \\ +\ 16_{16} \\ \hline \mathbf{39_{16}} \end{array}$$

right column: $3_{16} + 6_{16} = 3_{10} + 6_{10} = 9_{10} = 9_{16}$

left column: $2_{16} + 1_{16} = 2_{10} + 1_{10} = 3_{10} = 3_{16}$

(b)
$$\begin{array}{r} 58_{16} \\ +\ 22_{16} \\ \hline \mathbf{7A_{16}} \end{array}$$

right column: $8_{16} + 2_{16} = 8_{10} + 2_{10} = 10_{10} = A_{16}$

left column: $5_{16} + 2_{16} = 5_{10} + 2_{10} = 7_{10} = 7_{16}$

(c)
$$\begin{array}{r} 2B_{16} \\ +\ 84_{16} \\ \hline \mathbf{AF_{16}} \end{array}$$

right column: $B_{16} + 4_{16} = 11_{10} + 4_{10} = 15_{10} = F_{16}$

left column: $2_{16} + 8_{16} = 2_{10} + 8_{10} = 10_{10} = A_{16}$

(d)
$$\begin{array}{r} DF_{16} \\ +\ AC_{16} \\ \hline \mathbf{18B_{16}} \end{array}$$

right column: $F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10}$

$27_{10} - 16_{10} = 11_{10} = B_{16}$ with a 1 carry

left column: $D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10}$

$24_{10} - 16_{10} = 8_{10} = 8_{16}$ with a 1 carry

Related Problem

Add $4C_{16}$ and $3A_{16}$.

Hexadecimal Subtraction

There are three ways to get the 2's complement of a hexadecimal number.

Method 1:

Convert the hexadecimal number to binary. Take the 2's complement of the binary number. Convert the result to hexadecimal.

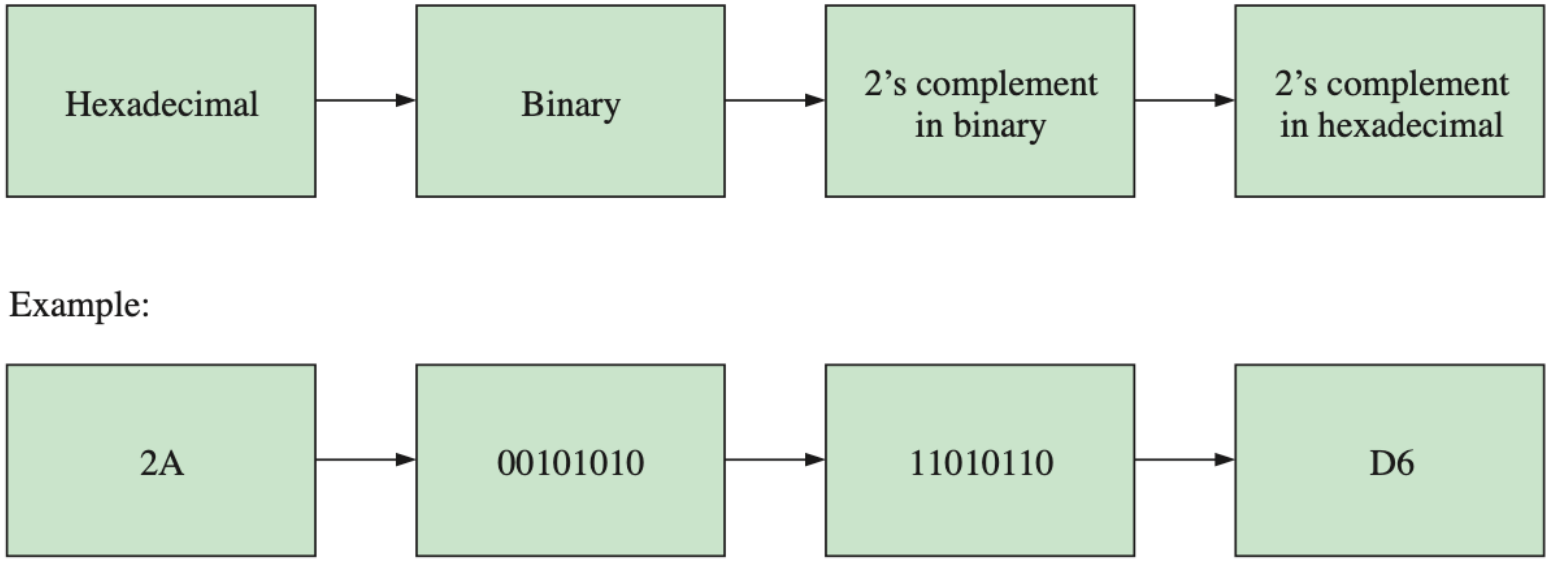
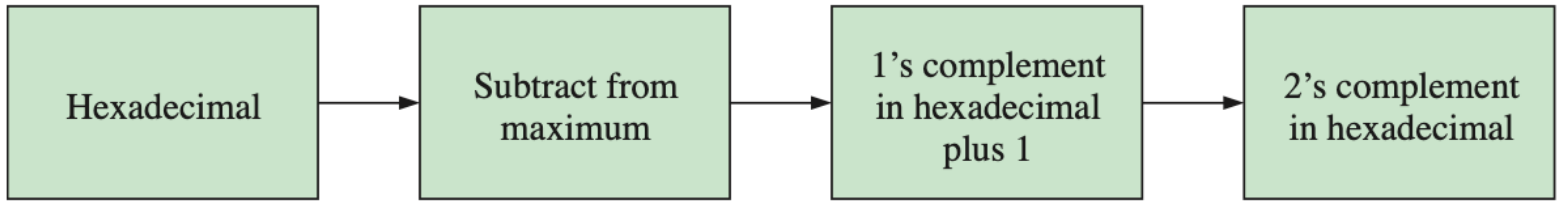


FIGURE 2–4 Getting the 2's complement of a hexadecimal number, Method 1.

Method 2:

Subtract the hexadecimal number from the maximum hexadecimal number and add 1.



Example:

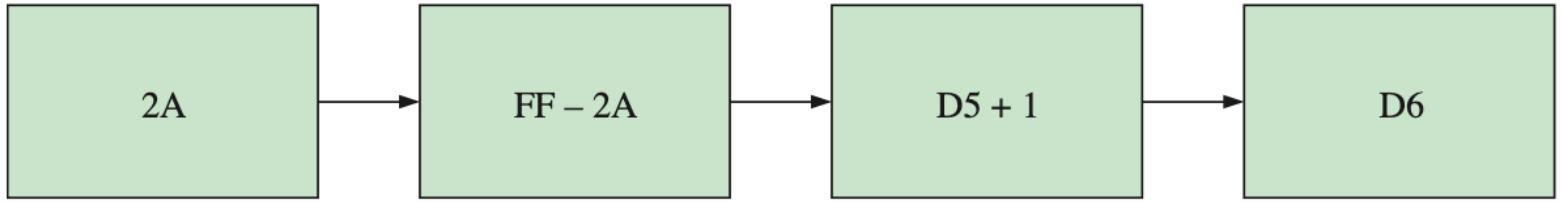


FIGURE 2–5 Getting the 2's complement of a hexadecimal number, Method 2.

Method 3:

Write the sequence of single hexadecimal digits. Write the sequence in reverse below the forward sequence. The 1's complement of each hex digit is the digit directly below it. Add 1 to the resulting number to get the 2's complement.

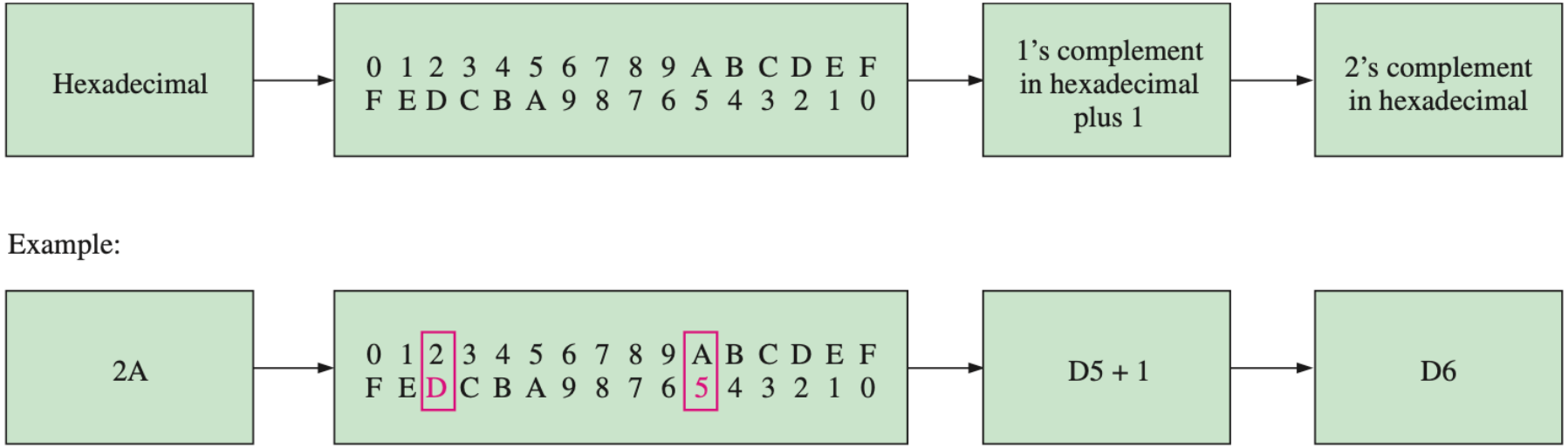


FIGURE 2–6 Getting the 2's complement of a hexadecimal number, Method 3.

EXAMPLE 2–30

Subtract the following hexadecimal numbers:

(a) $84_{16} - 2A_{16}$ **(b)** $C3_{16} - 0B_{16}$

Solution

(a) $2A_{16} = 00101010$

2's complement of $2A_{16} = 11010110 = D6_{16}$ (using Method 1)

$$\begin{array}{rl} 84_{16} & \\ +\ D6_{16} & \text{Add} \\ \hline \not{1}5A_{16} & \text{Drop carry, as in 2's complement addition} \end{array}$$

The difference is $\mathbf{5A_{16}}$.

(b) $0B_{16} = 00001011$

2's complement of $0B_{16} = 11110101 = F5_{16}$ (using Method 1)

$$\begin{array}{rl} C3_{16} & \\ +\ F5_{16} & \text{Add} \\ \hline \not{1}B8_{16} & \text{Drop carry} \end{array}$$

The difference is $\mathbf{B8}_{16}$.

Related Problem

Subtract 173_{16} from BCD_{16}.

9. Octal Numbers

- The octal number system provides a convenient way to express binary numbers and codes
- 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, …

- To distinguish octal numbers from decimal numbers or hexadecimal numbers, we will use the subscript 8 to indicate an octal number.

Octal-to-Decimal Conversion

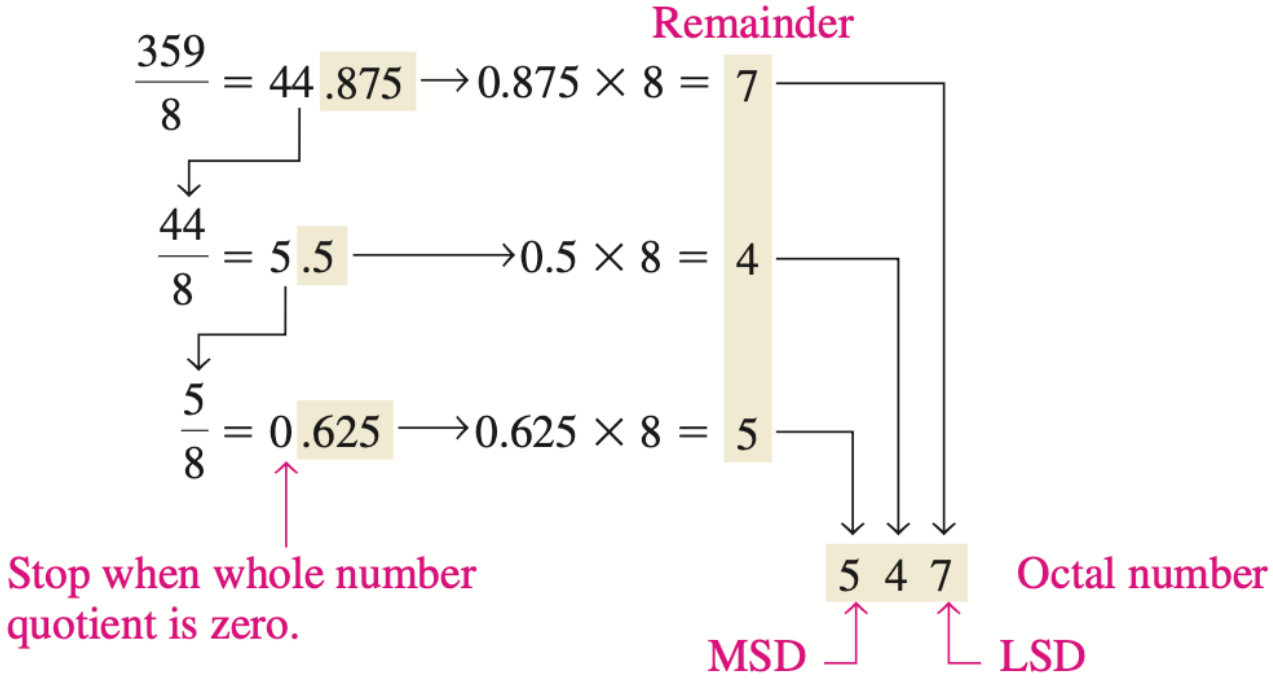
- Since the octal number system has a base of eight, each successive digit position is an increasing power of eight, beginning in the right-most column with 8^0.
- The evaluation of an octal number in terms of its decimal equivalent is accomplished by multiplying each digit by its weight and summing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

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

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:

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) 1 3 → **001011**

(b) 2 5 → **010101**

(c) 1 4 0 → **001100000**

(d) 7 5 2 6 → **111101010110**

Binary-to-Octal Conversion

- Conversion of a binary number to an octal number is the reverse of the octal-to-binary conversion.
- The procedure is as follows:

✓ Start with the right-most group of three bits and, moving from right to left, convert each 3-bit group to the equivalent octal digit.

✓ If there are not three bits available for the left-most group, add either one or two zeros to make a complete group.

✓ These leading zeros do not affect the value of the binary number.

EXAMPLE 2–32

Convert each of the following binary numbers to octal:

(a) 110101 **(b)** 101111001 **(c)** 100110011010 **(d)** 11010000100

Solution

(a) $\underbrace{110}_{6}\underbrace{101}_{5} = \mathbf{65}_8$

(b) $\underbrace{101}_{5}\underbrace{111}_{7}\underbrace{001}_{1} = \mathbf{571}_8$

(c) $\underbrace{1001}_{4}\underbrace{1001}_{6}\underbrace{1010}_{3}\underbrace{}_{2} = \mathbf{4632}_8$

(d) $\underbrace{0110}_{3}\underbrace{1000}_{2}\underbrace{0100}_{0}\underbrace{}_{4} = \mathbf{3204}_8$

Related Problem

Convert the binary number 1010101000111110010 to octal.

10. Binary Coded Decimal (BCD)

- In BCD, 4 bits represent each decimal digit.
- 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.

The 8421 BCD Code

- The 8421 code is a type of BCD (binary coded decimal) code. Binary coded decimal means that each decimal digit, 0 through 9, is represented by a binary code of four bits. The designation 8421 indicates the binary weights of the four bits (2^3, 2^2, 2^1, 2^0).
- The ease of conversion between 8421 code numbers and the familiar decimal numbers is the main advantage of this code.
- All you have to remember are the ten binary combinations that represent the ten decimal digits as shown in Table 2–5.
- The 8421 code is the predominant BCD code, and when we refer to BCD, we always mean the 8421 code unless otherwise stated.

TABLE 2–5

Decimal/BCD conversion.

Decimal Digit	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Invalid Codes

- You should realize that, with four bits, sixteen numbers (0000 through 1111) can be represented but that, in the 8421 code, only ten of these are used. The six code combinations that are not used—1010, 1011, 1100, 1101, 1110, and 1111—are invalid in the 8421 BCD code.

- To express any decimal number in BCD, simply replace each decimal digit with the appropriate 4-bit code.

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.

Invalid Codes

- It is equally easy to determine a decimal number from a BCD number. Start at the right-most 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) 10000110 → 8 6

(b) 001101010001 → 3 5 1

(c) 1001010001110000 → 9 4 7 0

Related Problem

Convert the BCD code 1000001000100111010 to decimal.

BCD Addition

BCD is a numerical code and can be used in arithmetic operations. Addition is the most important operation because the other three operations (subtraction, multiplication, and division) can be accomplished by the use of addition. Here is how to add two BCD numbers:

✓ **Step 1:**

Add the two BCD numbers, using the rules for binary addition.

✓ **Step 2:**

If a 4-bit sum is equal to or less than 9, it is a valid BCD number.

✓ **Step 3:**

If a 4-bit sum is greater than 9, or if a carry out of the 4-bit group is generated, it is an invalid result. Add 6 (0110) to the 4-bit sum in order to skip the six invalid states and return the code to 8421. If a carry results when 6 is added, simply add the carry to the next 4-bit group.

EXAMPLE 2–35

Add the following BCD numbers:

(a) 0011 + 0100

(b) 00100011 + 00010101

(c) 10000110 + 00010011

(d) 010001010000 + 010000010111

Solution

The decimal number additions are shown for comparison.

(a)

0011	3
+ 0100	+ 4
0111	7

(b)

0010	0011	23
+ 0001	0101	+ 15
0011	**1000**	38

(c)

1000	0110	86
+ 0001	0011	+ 13
1001	**1001**	**99**

(d)

0100	0101	0000	450
+ 0100	0001	0111	+ 417
1000	**0110**	**0111**	**867**

Note that in each case the sum in any 4-bit column does not exceed 9, and the results are valid BCD numbers.

Related Problem

Add the BCD numbers: 1001000001000011 + 0000100100100101.

EXAMPLE 2–36

Add the following BCD numbers:

(a) 1001 + 0100 **(b)** 1001 + 1001

(c) 00010110 + 00010101 **(d)** 01100111 + 01010011

Solution

The decimal number additions are shown for comparison.

(a)

```
  1001          9
+ 0100        + 4
  1101         13   Invalid BCD number (>9)
+ 0110              Add 6
------
 0001  0011         Valid BCD number
   ↓     ↓
   1     3
```

(b)

```
  1001          9
+ 1001        + 9
 1 0010       18   Invalid because of carry
+ 0110              Add 6
------
 0001  1000         Valid BCD number
   ↓     ↓
   1     8
```

(c)

```
  0001  0110          16
+ 0001  0101        + 15
  0010  1011         31   Right group is invalid (>9),
                          left group is valid.
      + 0110         Add 6 to invalid code. Add
                          carry, 0001, to next group.
------
 0011  0001          Valid BCD number
   ↓     ↓
   3     1
```

(d)

```
    0110  0111          67
  + 0101  0011        + 53
    1011  1010         120   Both groups are invalid (>9)
  + 0110  + 0110             Add 6 to both groups
--------------
 0001  0010  0000            Valid BCD number
   ↓     ↓     ↓
   1     2     0
```

Related Problem

Add the BCD numbers: 01001000 + 00110100.

10. Digital Codes

The Gray Code

- The Gray code is unweighted and is not an arithmetic code; that is, there are no specific weights assigned to the bit positions.
- The important feature of the Gray code is that it exhibits only a single bit change from one code word to the next in sequence.
- This property is important in many applications, such as shaft position encoders, where error susceptibility increases with the number of bit changes between adjacent numbers in a sequence.

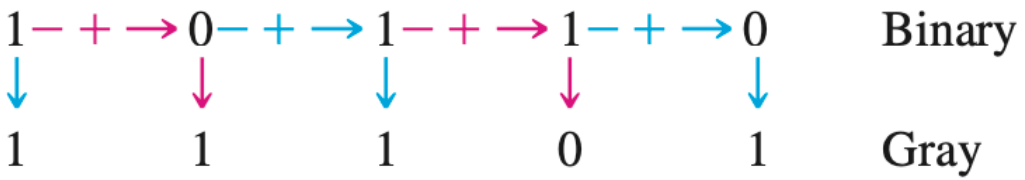
TABLE 2–6

Four-bit Gray code.

Decimal	Binary	Gray Code	Decimal	Binary	Gray Code
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

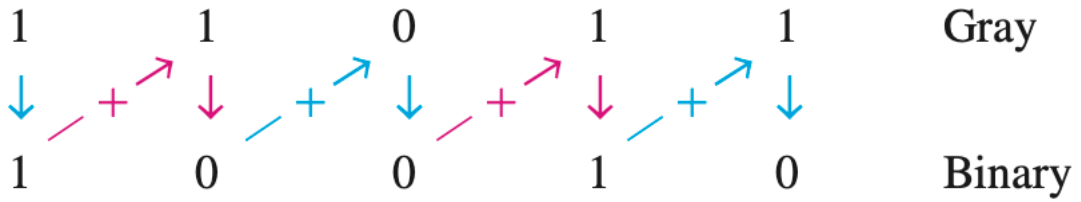
Binary-to-Gray Code Conversion

- Conversion between binary code and Gray code is sometimes useful. The following rules explain how to convert from a binary number to a Gray code word:
  - ✓ The most significant bit (left-most) in the Gray code is the same as the corresponding MSB in the binary number.
  - ✓ Going from left to right, add each adjacent pair of binary code bits to get the next Gray code bit. Discard carries.
- For example, the conversion of the binary number 10110 to Gray code is as follows:



Gray-to-binary Code Conversion

- To convert from Gray code to binary, use a similar method; however, there are some differences. The following rules apply:
  - ✓ The most significant bit (left-most) in the binary code is the same as the corresponding bit in the Gray code.
  - ✓ Add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries.
- For example, the conversion of the Gray code word 11011 to binary is as follows:



Gray-to-binary Code Conversion

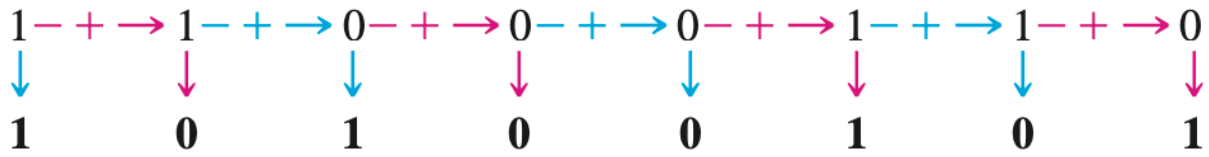
EXAMPLE 2–37

(a) Convert the binary number 11000110 to Gray code.

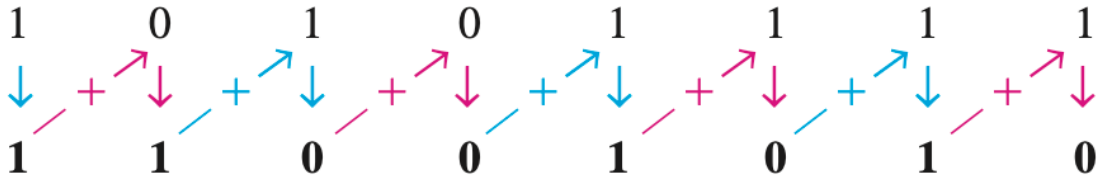
(b) Convert the Gray code 10101111 to binary.

Solution

(a) Binary to Gray code:



(b) Gray code to binary:



Related Problem

(a) Convert binary 101101 to Gray code.

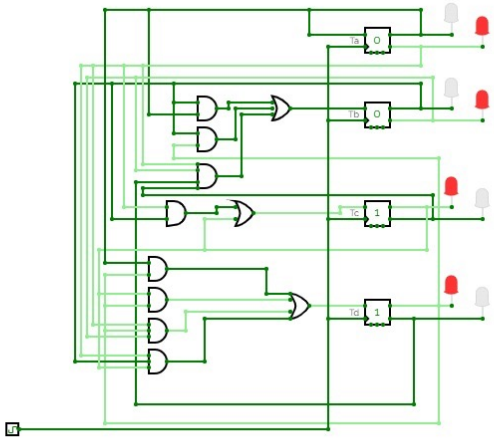
(b) Convert Gray code 100111 to binary.

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.
- 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.

THE END

Lecture 2: Number Systems, Operations, and Codes



INSTRUCTOR: Dr. Vuong Quoc Bao