CHAPTER 2

NUMBER SYSTEMS, OPERATIONS, AND CODES

数字系统、运算和编码

2-1 DECIMAL NUMBERS

十进制数

Decimal Review

- Numbers consist of a bunch of digits, each with a weight.

1	6	2	.	3	7	5	Digits
100	10	1		1/10	1/100	1/1000	Weights

- These weights are all powers of the base, which is 10. We can rewrite this:

1	6	2	.	3	7	5	Digits
10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	Weights

- To find the decimal value of a number, multiply each digit by its weight and sum the products.

$$1\times10^2+6\times10^1+2\times10^0+3\times10^{-1}+7\times10^{-2}+5\times10^{-3}=162.375$$

$$D=\sum k_i * 10^i$$

Nothing Special about 10!

- Decimal system (and the idea of "0") was invented in India around 100-500AD.
- Why did they use 10? Anything special about it?
  - Not really.
  - Probably the fact that we have 10 fingers influenced this.
- Will a base other than 10 work?
  - Sure.

$$345 \text{ in base } 9 = 3\times 9^2 + 4\times 9^1 + 5\times 9^0 = 284 \text{ in base } 10$$

- What about base 2?

$$D = \sum k_i * N^i$$

2-2 BINARY NUMBERS
二进制数

Introductory Paragraph

- The binary number system is simply another way to represent quantities.
- The binary system is **less complicate**d than the decimal system because it has only two digits. It may seem more difficult at first because it is unfamiliar to you.
- 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 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.

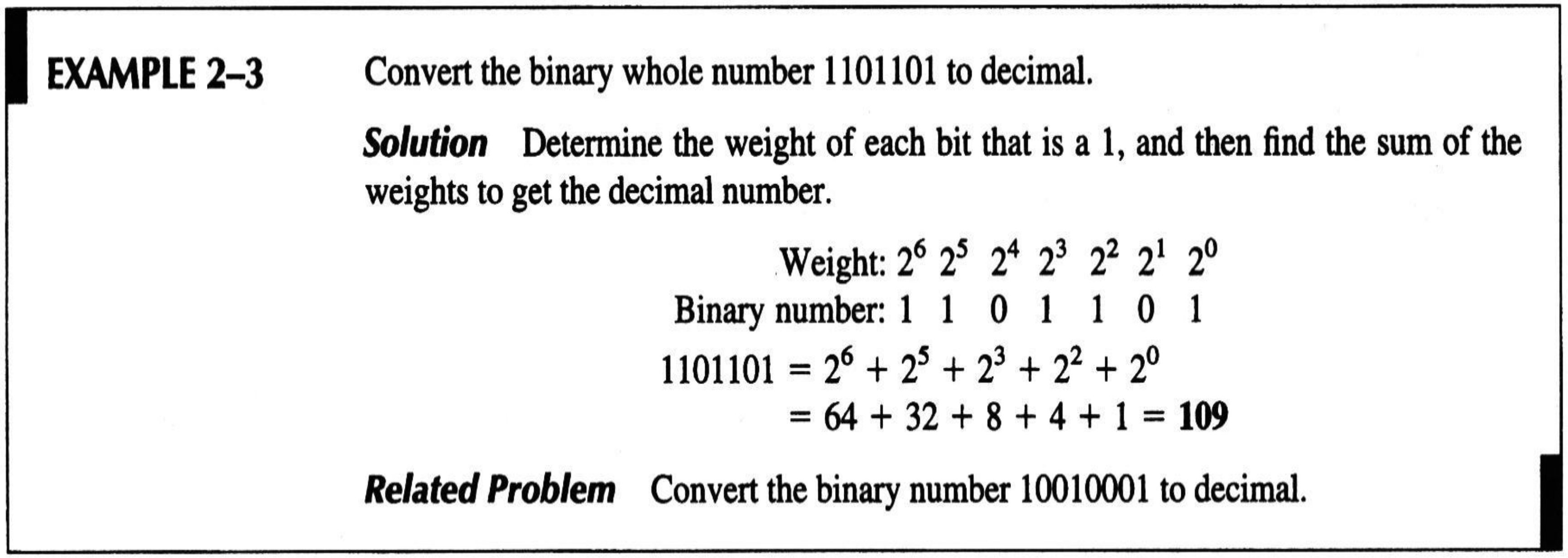
The Weighting Structure of Binary Numbers

- A binary number is a weighted number（加权数）.
- The right-most bit is the LSB（最低有效位 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（最高有效位 most significant bit）.
- Fractional numbers（小数） can also be represented in binary by placing bits to the right of the binary point. The left-most bit is the MSB in a binary fractional number and has a weight of $2^{-1}=0.5$. The fractional weights decreases from left to right by a negative power of two for each bit.
- The weight structure of a binary number is

$$\cdots\ 2^3 2^2 2^1 2^0 . 2^{-1} 2^{-2} 2^{-3}\ \cdots$$

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.

Weight:	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary number:	1	1	0	1	1	0	1

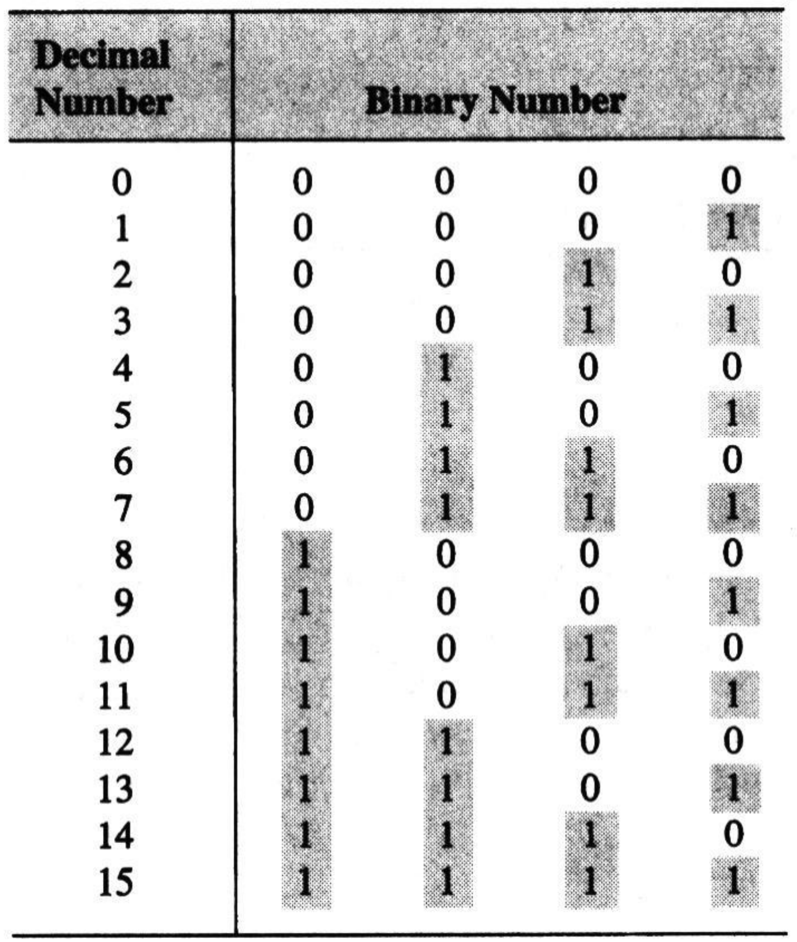
$$\begin{aligned} 1101101 &= 2^6 + 2^5 + 2^3 + 2^2 + 2^0 \\ &= 64 + 32 + 8 + 4 + 1 = \mathbf{109} \end{aligned}$$

Related Problem Convert the binary number 10010001 to decimal.

Counting in Binary

- A binary count of 0 through 15 is shown below. As you will see, 4 bits are required to count from 0 to 15.

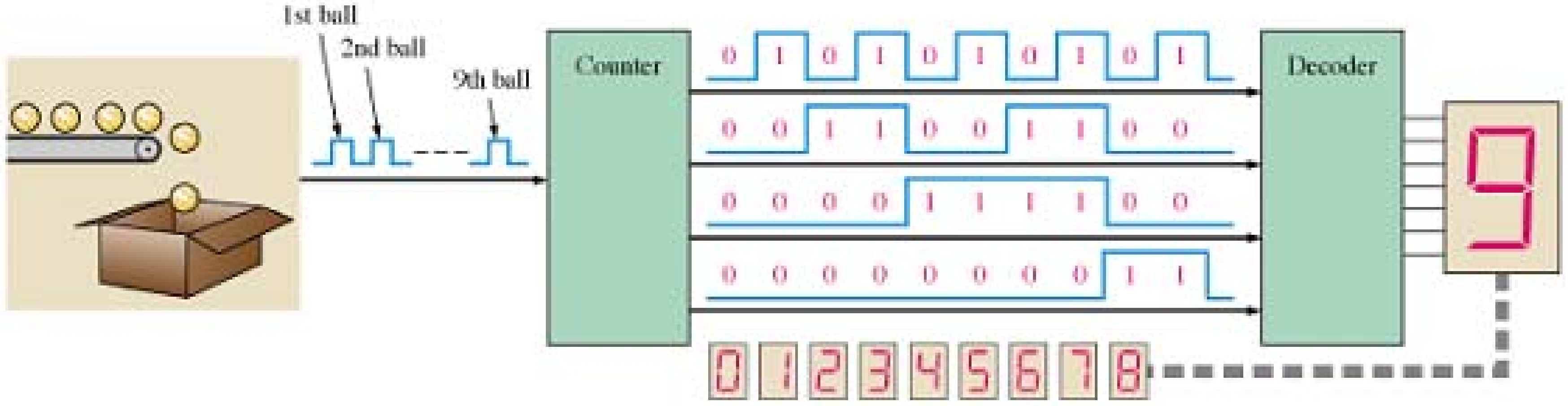
TABLE 2–1



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

Counting Tennis Balls Going into a box from a Conveyor Belt

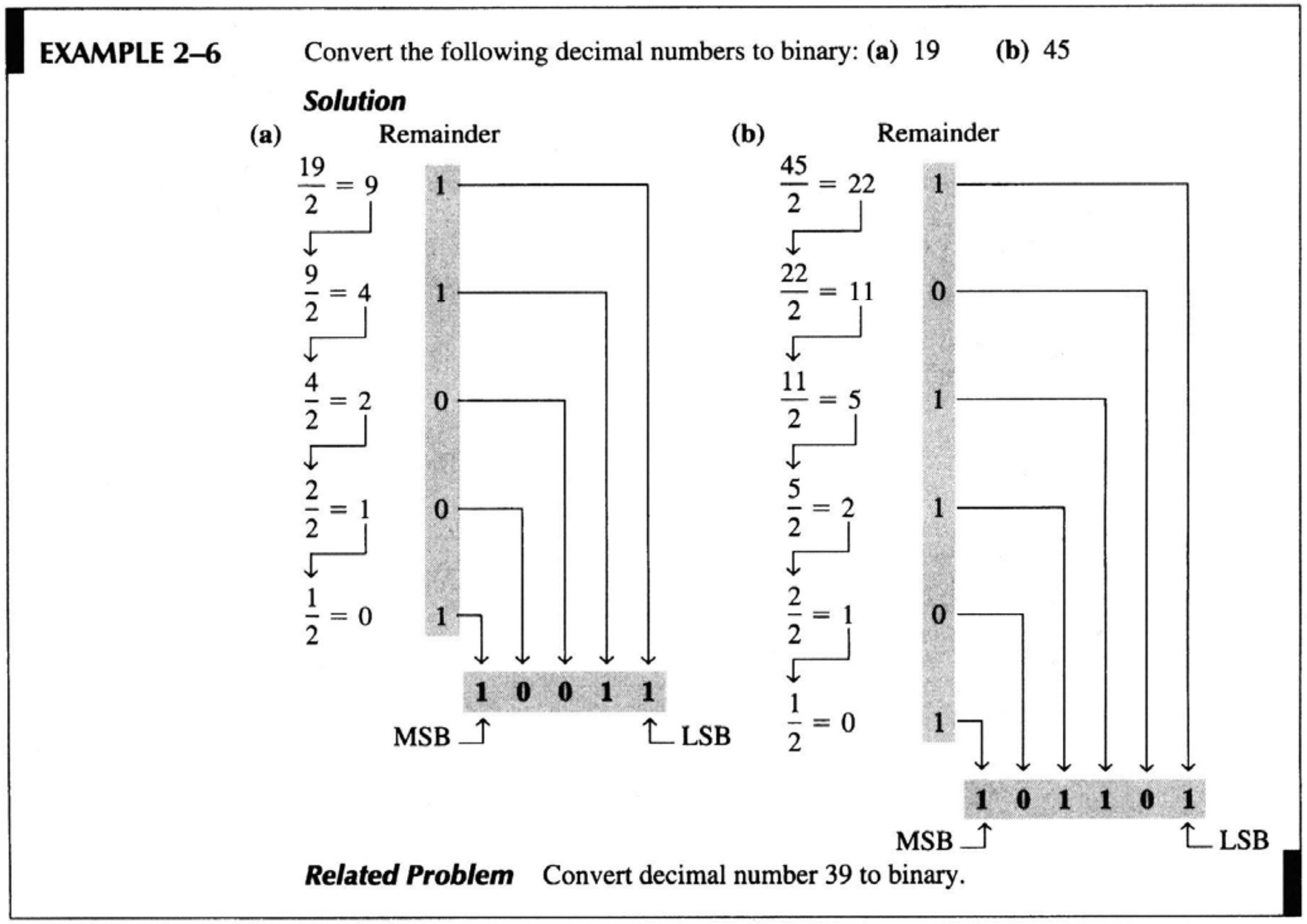
- The counter counts the pulses from a sensor that detects the passing of a ball and produces a sequence of logic levels (digital waveforms) on each of its four parallel outputs. Each set of logic levels represent a 4-bit binary number. The decoder decodes each set of four bits and converts it to the corresponding decimal number in the 7-segment display.



2-3 DECIMAL-TO-BINARY CONVERSION

Repeated Division-by-2 Method

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



EXAMPLE 2–6 Convert the following decimal numbers to binary: **(a)** 19 **(b)** 45

Solution

(a) Remainder

$\frac{19}{2} = 9$ 1

$\frac{9}{2} = 4$ 1

$\frac{4}{2} = 2$ 0

$\frac{2}{2} = 1$ 0

$\frac{1}{2} = 0$ 1

1 0 0 1 1

MSB LSB

(b) Remainder

$\frac{45}{2} = 22$ 1

$\frac{22}{2} = 11$ 0

$\frac{11}{2} = 5$ 1

$\frac{5}{2} = 2$ 1

$\frac{2}{2} = 1$ 0

$\frac{1}{2} = 0$ 1

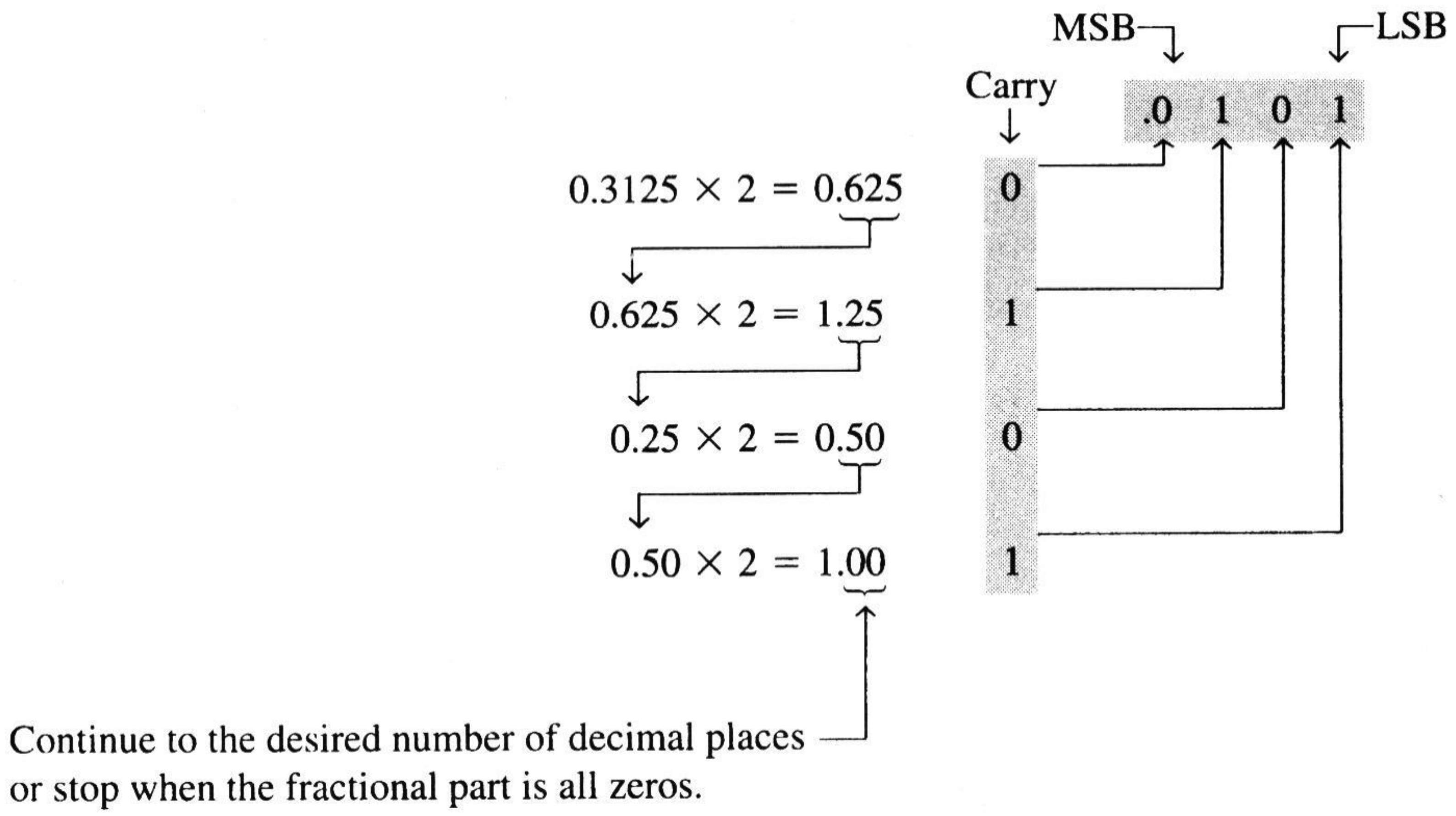
1 0 1 1 0 1

MSB LSB

Related Problem Convert decimal number 39 to binary.

Repeated Multiplication-by-2 Method

- *A systematic method of converting fractional numbers from decimal to binary is the repeated multiplication-by-2 process.*



MSB
LSB
Carry
.0 1 0 1
0.3125 × 2 = 0.625
0
0.625 × 2 = 1.25
1
0.25 × 2 = 0.50
0
0.50 × 2 = 1.00
1
Continue to the desired number of decimal places or stop when the fractional part is all zeros.

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

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

- [Example] Add 1111 and 1100.

Binary Subtraction

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

0 - 0 = 0 Difference of 0 with a borrow of 0

0 - 1 = 1 Difference of 1 with a borrow of 1

1 - 0 = 1 Difference of 1 with a borrow of 0

1 - 1 = 0 Difference of 0 with a borrow of 0

- [Example] Subtract 100 from 111.

Binary Multiplication

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

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

- [Example] Multiply 1101 by 1010.

Binary Division

- Division in binary follows the same procedure as division in decimal.

- [Example] Divide 1100 by 100.

2-5 SIGNED NUMBERS

带符号数

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 is for positive, and a 1 is for negative.

2.5.1 Sign-Magnitude(符号-幅值) System

- 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. The magnitude bits are in true (uncomplemented) binary for both positive and negative numbers.

$$\underbrace{\overbrace{0}^{\textit{Sign bit}}\quad\underbrace{0011001}_{\textit{Magnitude bits}}}_{+25} \qquad\qquad \underbrace{\overbrace{1}^{\textit{Sign bit}}\quad\underbrace{0011001}_{\textit{Magnitude bits}}}_{-25}$$

- The decimal values are determined by summing the weights in all the magnitude bit positions where there are 1s. The sign is determined by examination of the sign bit.

Sign-Magnitude System

- [Example 2-15] Determine the decimal value of this signed binary number expressed in sign-magnitude: 10010101.

Solution.

Summing the weights in all the magnitude bit positions where there are 1s,

$$2^4 + 2^2 + 2^0 = 21$$

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

2.5.2 The Complement System of Binary Numbers

1’s complement and
2’s complement

1’s Complement of a Binary Number

- The 1’s complement of a binary number is found by changing all 1s to 0s and all 0s to 1s.

- [Example] Find the 1’s complement of binary number 10110010.

1's Complement System

- **Positive numbers** in the 1's complement system are represented the same way as the positive sign-magnitude numbers.
- **Negative numbers**, however, are the 1's complements of the corresponding positive numbers.

$$\underbrace{00011001}_{+25} \qquad \underbrace{11100110}_{-25}$$

- The decimal values of positive numbers are determined by summing the weights in all bit positions where there are 1s.
- The decimal values of negative numbers are determined by summing the weights in all bit positions where there are 1s, and adding 1 to the result. The weight of the sign bit is given a negative value.

1’s Complement System

- [Example 2-16] Determine the decimal value of the signed binary numbers expressed in 1’s complement: (a) 00010111 (b) 11101000.

Solution.

(a) Summing the weights in all the magnitude bit positions where there are 1s,

$$2^4 + 2^2 + 2^1 + 2^0 = +23$$

(b) Summing the weights in all the magnitude bit positions where there are 1s,

$$-2^7 + 2^6 + 2^5 + 2^3 = -24$$

Adding 1 to the result

$$-24 + 1 = -23$$

1's Complement System

- Why?

$$\underbrace{00011001}_{+25} \qquad \underbrace{11100110}_{-25}$$

$$1110\ 0110 = \boxed{11111111 - 00011001}$$

$$= 1\ 0000\ 0000 - 1 - 00011001$$

$$-00011001 = 1110\ 0110 - 1\ 0000\ 0000 + 1$$

$$= 1000\ 0000 + 110\ 0110 - 1\ 0000\ 0000 + 1$$

$$\boxed{= -1000\ 0000 + 110\ 0110 + 1}$$

$$= -2^7 + 2^6 + 2^5 + 2^2 + 2^1 + 1$$

2's Complement of a Binary Number

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

- [Example] Find the 2's complement of binary number 10110010.

2's Complement System

- **Positive numbers** in the 2's complement system are represented the same way as in sign-magnitude and 1's complement systems.
- **Negative numbers** are the 2's complements of the corresponding positive numbers.

$$\underbrace{00011001}_{+25} \qquad\qquad \underbrace{11100111}_{-25}$$

- The decimal values are determined by summing the weights in all bit positions where there are 1s. The weight of the sign bit is given a negative value.

2’s Complement System

- [Example 2-17] Determine the decimal value of the signed binary numbers expressed in 2’s complement: (a) 01010110 (b) 10101010.

Solution.

(a) Summing the weights in all the magnitude bit positions where there are 1s,

$$2^6 + 2^4 + 2^2 + 2^1 = +86$$

(b) Summing the weights in all the magnitude bit positions where there are 1s,

$$-2^7 + 2^5 + 2^3 + 2^1 = -86$$

Range of Signed Integer Numbers

- The number of different combinations of n bits is

$$Total\ combinations = 2^n$$

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

$$-2^{n-1} \quad to \quad (2^{n-1} - 1)$$

- For 1's complement signed numbers, the range of value for n-bit numbers is

$$-2^{n-1} + 1 \quad to \quad (2^{n-1} - 1)$$

2-6 ARITHMETIC OPERATIONS WITH SIGNED NUMBERS
带符号数的算术运算

Addition（加法）

- The two numbers in an addition are the augend（被加数）and the addend（加数）. The results are the sum （和）and the carry（进位）.

$$[x+y]_{2c} = [x]_{2c} + [y]_{2c}$$

- The addition process is stated as follows: **add the two numbers and discard any final carry bit.**

- [Example]
  (a) 00000111(7) + 00000100(4) = ?
  (b) 00001111(15) + 11111010(-6) = ?
  (c) 00010000(16) + 11101000(-24) = ?
  (d) 11111011(-5) + 11110111(-9) = ?

00001011 (11)

1 00001001 (9)

Carry, discard

Overflow Condition（溢出条件）

- 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.

- [Example]
  (a) 01111101(125) + 00111010(58) =10110111(183) ?
  (b) 10001000 + 11101101 = ?

Subtraction（减法）

- The two numbers in a subtraction are the minuend（被减数） and the subtrahend（减数）. The results are the difference（差） and the borrow（借位）.

$$[x-y]_{2c}=[x+(-y)]_{2c}=[x]_{2c}+[-y]_{2c}$$

minuend $\nearrow$ $\nwarrow$ substrahend

- The subtraction process is stated as follows: **take the 2's complement of the subtrahend and add. Discard any final carry bit.**

- [Example]
  (a) 0001000 - 00000011 = ?
  (b) 11100111 - 00010011 = ?

Multiplication（乘法）

- The two numbers in a multiplication are the multiplicand（被乘数）and the multiplier（乘数）. The result is the product（积）.

- The multiplication operation in most computers is accomplished using partial product method（部分积方法）. The basic steps in the process are as follows:
  - Determine if the signs of the two numbers are the same. This determines what the sign of the product will be.
  - Change any negative number to **true (uncomplemented) form**.
  - Starting with the LSB of the multiplier, generate the partial products. Shift each successive partial product one bit to the left.
  - Add each partial product to the sum of the previous partial products to get the final product.
  - If the sign of the product is negative, take the 2's complement of the product. Attach the sign bit to the product.

Multiplication

- [Example 2-22] Multiply the signed binary number 01010011 and 11000101.

```
0   01010011          1   00111011
↑                     ↑
sign                  sign

                         01010011
         ×               00111011
         ------------------------
                         01010011
                        01010011
                      01010011
                     01010011
         +          01010011
         ------------------------
sign  →  1          1001100100001
         ------------------------
                         ↓
                   101100110011111
```

Division （除法）

- The two numbers in a division are the dividend（被除数） and the divisor（除数）. The results are the quotient（商） and the remainder（余数）.

- The basic steps in a division the process are as follows:
  - Determine if the signs of the two numbers are the same. This determines what the sign of the quotient will be.
  - Change any negative number to **true (uncomplemented) form**.
  - Initialize the quotient to zero and initialize the partial remainder to the dividend.
  - Subtract the divisor from the partial remainder using 2's complement addition to get the next partial remainder. If the result is positive, add 1 to the quotient and repeat for the next partial remainder; otherwise, the division is complete.

Division

- [Example 2-23] Divide 01100100（100） by 00011001（25）
- [Example 2-23a] Divide 01100100（100） by 00011010（26）
- [Example 2-23b] Divide 01100100（100） by 11100111（-26）
- [Example 2-23c] Divide 10011100（-100） by 00011010 （+26）
- [Example 2-23d] Divide 10011100（-100） by 11100111 （-26）

2’s Complement Advantage

- To convert to decimal
  - The 2’s complement system simply requires a summation of weights regardless of whether the number is positive or negative.

(在二进制转十进制的计算中，采用2的补码形式表示的数字，不管是正数还是负数仅需要一个统一的权重求和的计算方式就可以实现)

  - The sign-magnitude system requires two steps - sum the weights of the magnitude bits and examine the sign bit to determine if the number is positive or negative.

(而采用'符号-幅值'表示形式则需要两个步骤——（1）对幅值表示部分进行权重求和；（2）判断符号位是正还是负)

2’s Complement Advantage

- The 1's complement system requires adding 1 to the summation of weights for negative numbers but not for positive numbers.

(1的补码形式表示的二进制数，对于复数需要在最后的结果进行加1，对于正数则不需要，因此无法实现计算算法的统一)

- Also, the 1's complement system is not used because two representations of zero (00000000 or 11111111) are possible.

(1的补码形式另一个缺陷，零的表示有两种表示（ 00000000 or 11111111 ），表达形式不唯一)

2-8 HEXADECIMAL NUMBERS（十六进制数）

Why Hexadecimal?

- As you are probably aware, long binary numbers are difficult to read and write because it is easy to drop or transpose a bit. Since computers and microprocessors understand only 1s and 0s, it is necessary to use these digits when you program in machine language.

- 用二进制来表示一个数字需要占用较多的位数，书写起来较为麻烦，容易出错。但是计算机系统中只能识别0和1两个数字，因此在计算机的实际执行代码中必须采用二进制。

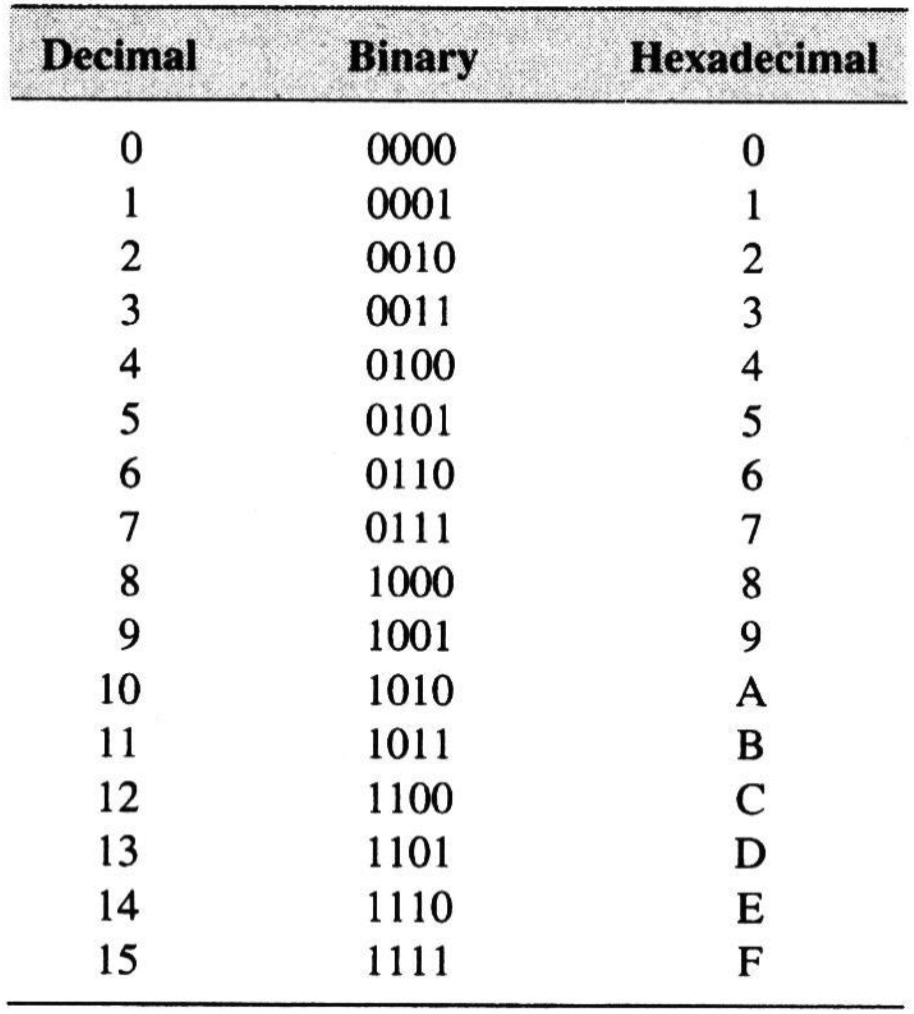
Why Hexadecimal?

- The hexadecimal number system has 16 digits and is used primarily as a compact way of displaying or writing binary numbers because it is very easy to convert between binary and hexadecimal.
- 十六进制数具有16个数字，是一种高效紧凑表示二进制数的一种技术方法，同时十六进制数和二进制数之间的转换也非常方便。
- 10 numeric digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) and 6 alphabetic characters (A, B, C, D, E, F) make up the hexadecimal number system.

Relationship between hexadecimal and binary

- 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

- Very straightforward! 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) 1001111011110011100

$$\underbrace{1100}_{\downarrow\ C}\ \underbrace{1010}_{\downarrow\ A}\ \underbrace{0101}_{\downarrow\ 5}\ \underbrace{0111}_{\downarrow\ 7}$$

= CA57

4F79C

Hexadecimal-to-Binary Conversion

- Very straightforward! Simply replace each hexadecimal symbol with the equivalent 4-bit group.
- [Example 2-25] Determine the binary numbers for the following hexadecimal numbers:

(a) $10A4_h$ (b) $CF8E_h$ (c) 9742_h

1 0000 1010 0100

1100 1111 1000 1110

1001 0111 0100 0010

Hexadecimal-to-Decimal Conversion

- Multiply the decimal value of each hexadecimal digit by its weight and then take the sum of these products.
- [Example 2-27] Convert the following hexadecimal numbers to decimal:

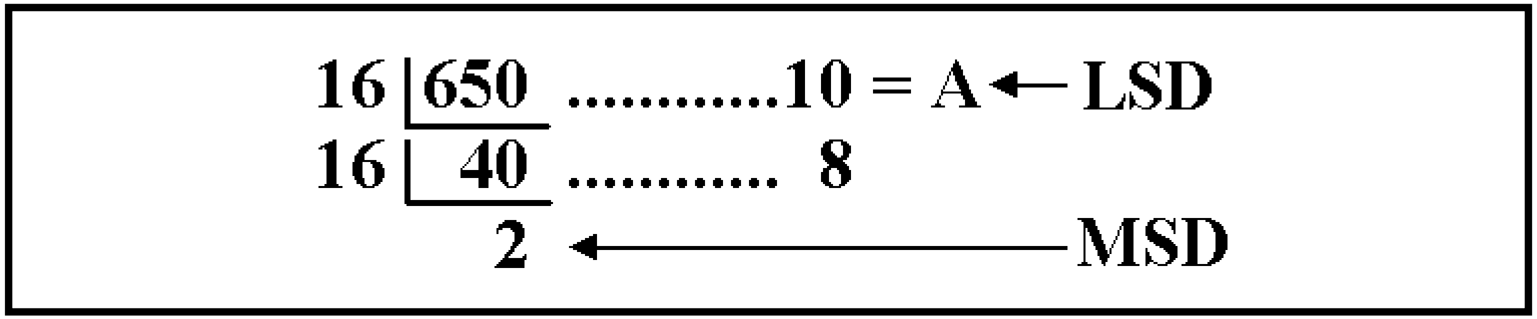
(a) $E5_h$ (b) $B2F8_h$

229

45816

Decimal-to-Hexadecimal Conversion

- Repeated division of a decimal number by 16 will produce the equivalent hexadecimal number.
- [Example 2-28] Convert the decimal number 650 to hexadecimal by repeated division by 16.



2-9 BINARY CODED DECIMAL (BCD)

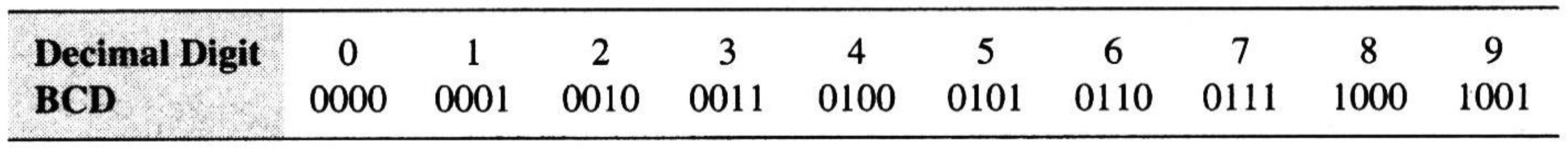
Introductory Paragraph

- Binary coded decimal (BCD) is a way to express each of the decimal digits with a binary code. Since there are only ten code groups in the BCD system, 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.

The 8421 Code

- The 8421 code is a type of BCD code.
- BCD means that each decimal digit, 0 through 9, is represented by a binary code of four bits.

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

- The designation 8421 indicates the binary weights of the four bits.
- 1010, 1011, 1100, 1101, 1110, and 1111 are invalid codes.
- The 8421 code is the predominant BCD code, and when we refer to BCD, we always mean the 8421 code unless otherwise stated.

The 8421 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

- 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:

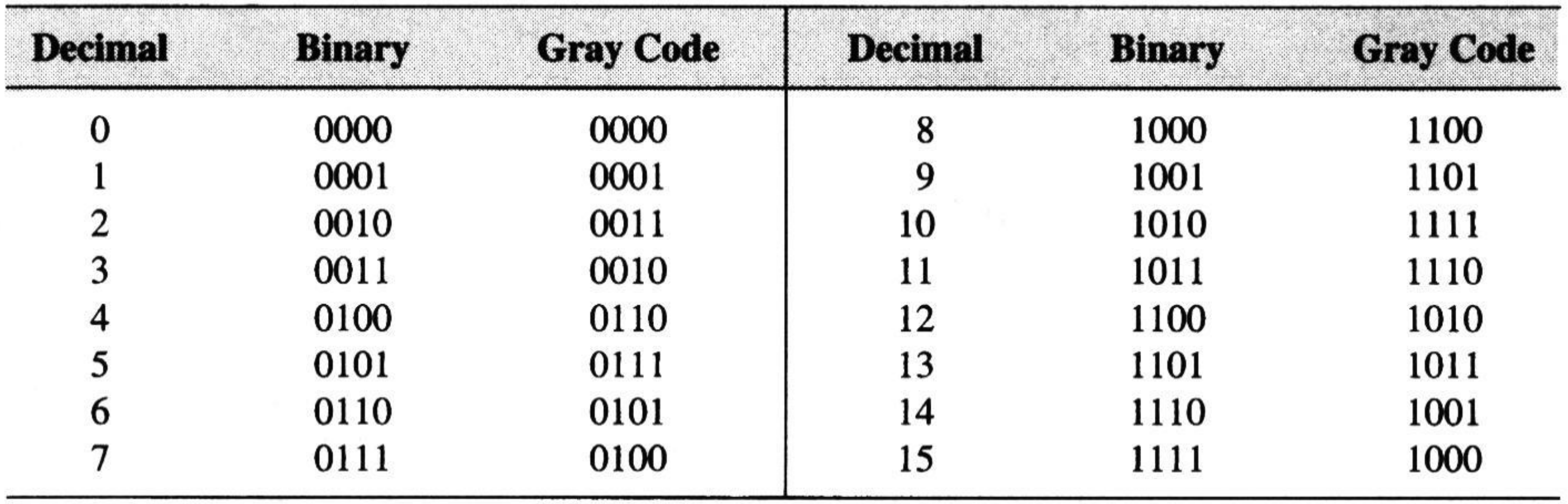
  (a) 10000110 (b) 001101010001 (c) 1001010001110000

2-10 DIGITAL CODES AND PARITY
数字编码和奇偶校验

The Gray Code（格雷码）

- 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 number to the next.相邻码

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

- The MSB 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.

$$\begin{tikzcd}
1 \arrow[r, "+" description] \arrow[d] & 1 \arrow[r, "+" description] \arrow[d] & 0 \arrow[r, "+" description] \arrow[d] & 0 \arrow[r, "+" description] \arrow[d] & 0 \arrow[r, "+" description] \arrow[d] & 1 \arrow[r, "+" description] \arrow[d] & 1 \arrow[r, "+" description] \arrow[d] & 0 \arrow[d] \\
\mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1}
\end{tikzcd}$$

Gray-to-Binary Conversion

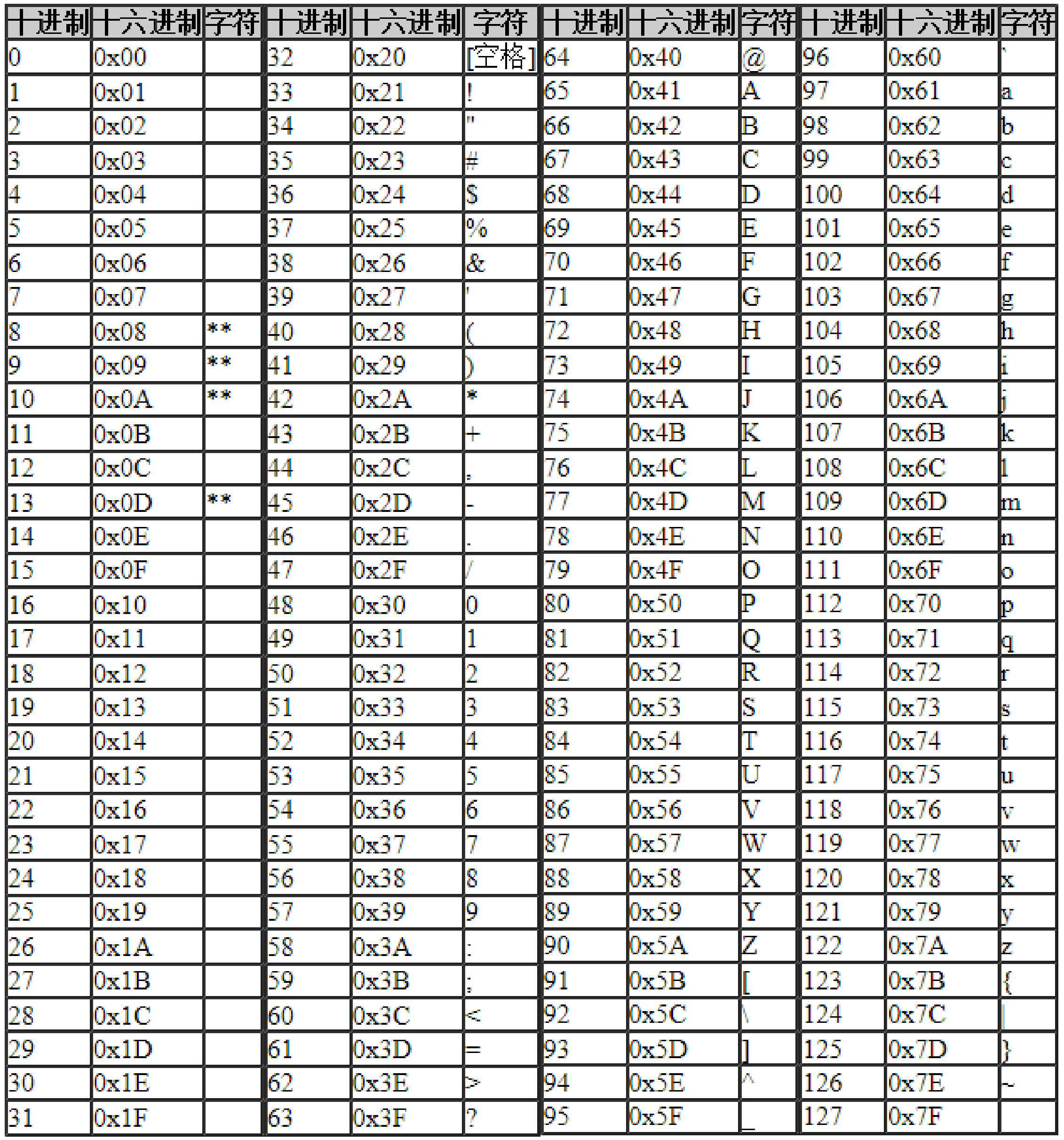
- The MSB in the binary code is the same as the corresponding MSB in the Gray code.
- Going from left to right, add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries.

$$\begin{tikzcd}
1 \arrow[d] & 0 \arrow[d] & 1 \arrow[d] & 0 \arrow[d] & 1 \arrow[d] & 1 \arrow[d] & 1 \arrow[d] & 1 \arrow[d] \\
1 \arrow[ur, "+"] & 1 \arrow[ur, "+"] & 0 \arrow[ur, "+"] & 0 \arrow[ur, "+"] & 1 \arrow[ur, "+"] & 0 \arrow[ur, "+"] & 1 \arrow[ur, "+"] & 0
\end{tikzcd}$$

Alphanumeric Codes（字母数字编码）

- In the strictest sense, alphanumeric codes are codes that represent numbers and alphabetic characters (letters). Most such codes, however, also represent other characters such as symbols and various instructions necessary for conveying information. The ASCII is the most common alphanumeric code.
- ASCII has 128 characters represented by a 7-bit binary code.
  - The first 32 are nongraphic characters (control characters). That are never printed or displayed and used only for control purposes.
  - The others are graphic characters 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.

ASCII Code



十进制	十六进制	字符	十进制	十六进制	字符	十进制	十六进制	字符	十进制	十六进制	字符
0	0x00		32	0x20	[空格]	64	0x40	@	96	0x60	`
1	0x01		33	0x21	!	65	0x41	A	97	0x61	a
2	0x02		34	0x22	"	66	0x42	B	98	0x62	b
3	0x03		35	0x23	#	67	0x43	C	99	0x63	c
4	0x04		36	0x24	$	68	0x44	D	100	0x64	d
5	0x05		37	0x25	%	69	0x45	E	101	0x65	e
6	0x06		38	0x26	&	70	0x46	F	102	0x66	f
7	0x07		39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	**	40	0x28	(	72	0x48	H	104	0x68	h
9	0x09	**	41	0x29	)	73	0x49	I	105	0x69	i
10	0x0A	**	42	0x2A	*	74	0x4A	J	106	0x6A	j
11	0x0B		43	0x2B	+	75	0x4B	K	107	0x6B	k
12	0x0C		44	0x2C	,	76	0x4C	L	108	0x6C	l
13	0x0D	**	45	0x2D	-	77	0x4D	M	109	0x6D	m
14	0x0E		46	0x2E	.	78	0x4E	N	110	0x6E	n
15	0x0F		47	0x2F	/	79	0x4F	O	111	0x6F	o
16	0x10		48	0x30	0	80	0x50	P	112	0x70	p
17	0x11		49	0x31	1	81	0x51	Q	113	0x71	q
18	0x12		50	0x32	2	82	0x52	R	114	0x72	r
19	0x13		51	0x33	3	83	0x53	S	115	0x73	s
20	0x14		52	0x34	4	84	0x54	T	116	0x74	t
21	0x15		53	0x35	5	85	0x55	U	117	0x75	u
22	0x16		54	0x36	6	86	0x56	V	118	0x76	v
23	0x17		55	0x37	7	87	0x57	W	119	0x77	w
24	0x18		56	0x38	8	88	0x58	X	120	0x78	x
25	0x19		57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1A		58	0x3A	:	90	0x5A	Z	122	0x7A	z
27	0x1B		59	0x3B	;	91	0x5B	[	123	0x7B	{
28	0x1C		60	0x3C	<	92	0x5C	\	124	0x7C	|
29	0x1D		61	0x3D	=	93	0x5D	]	125	0x7D	}
30	0x1E		62	0x3E	>	94	0x5E	^	126	0x7E	~
31	0x1F		63	0x3F	?	95	0x5F	_	127	0x7F	

Parity Method for Error Detection
（用于错误检测的奇偶校验法）

- Many systems use a parity bit as a means for bit error detection.
- Any group of bits contain either an even or an odd number of 1s. A parity bit is attached to a group of bits to make the total number of 1s in a group always even or always odd. An even/odd parity bit makes the total number of 1s even/odd.

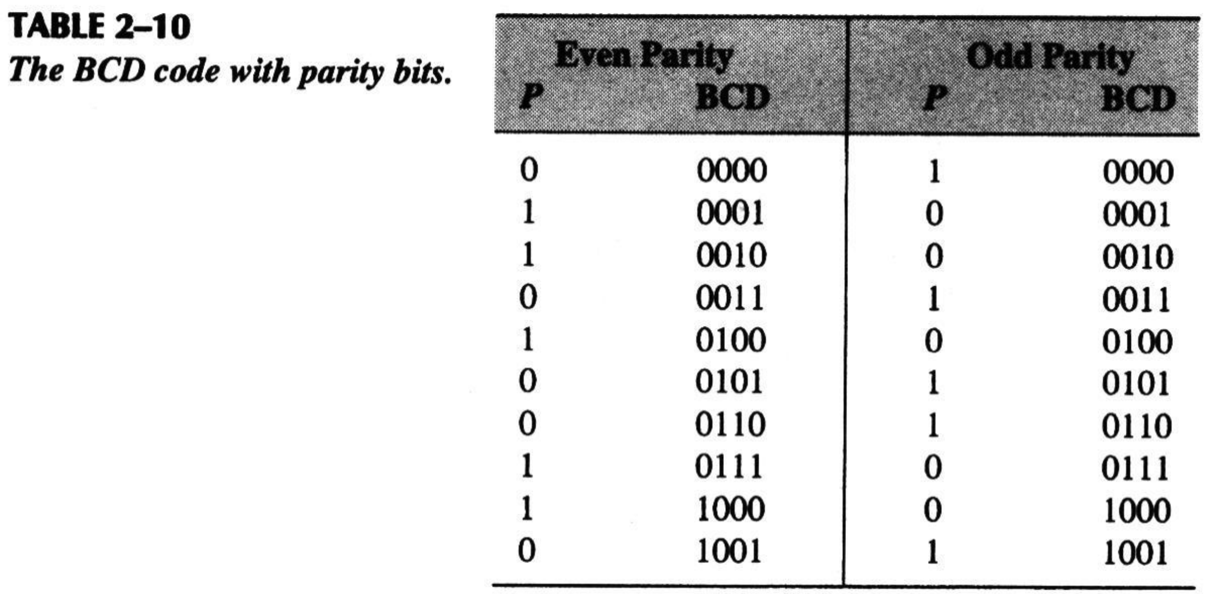


TABLE 2–10
The BCD code with parity bits.

Even Parity		Odd Parity	
P	**BCD**	***P***	**BCD**
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

Quiz

1. For the binary number 1000, the weight of the column with the 1 is

   a. 4

   b. 6

   c. 8

   d. 10

Quiz

2. The 2’s complement of 1000 is

   a. 0111

   b. 1000

   c. 1001

   d. 1010

Quiz

3. The fractional binary number 0.11 has a decimal value of

a. ¼

b. ½

c. ¾

d. none of the above

Quiz

4. The hexadecimal number 2C has a decimal equivalent value of

a. 14

b. 44

c. 64

d. none of the above

Quiz

5. Assume that a floating point number is represented in binary. If the sign bit is 1, the

   a. number is negative

   b. number is positive

   c. exponent is negative

   d. exponent is positive

Quiz

6. When two positive signed numbers are added, the result may be larger that the size of the original numbers, creating overflow. This condition is indicated by

a. a change in the sign bit

b. a carry out of the sign position

c. a zero result

d. smoke

Quiz

7. The number 1010 in BCD is

   a. equal to decimal eight

   b. equal to decimal ten

   c. equal to decimal twelve

   d. invalid

Quiz

8. An example of an unweighted code is

a. binary

b. decimal

c. BCD

d. Gray code

Quiz

9. An example of an alphanumeric code is

   a. hexadecimal

   b. ASCII

   c. BCD

   d. CRC

Quiz

10. An example of an error detection method for transmitted data is the

a. parity check

b. CRC

c. both of the above

d. none of the above

Homework

- Problems: 4, 6, 7(a), 12, 13(h), 14(c), 15(f), 16, 21(f), 22(h), 23(c)(d), 24(c)(d), 25(c)(d), 28(a)(b), 32(b), 33(b), 34, 37(g), 38(f), 39(h), 40(h), 47(a), 51(j), 53(f), 56(a), 57(b), 63(a), 65(a).

Answers

1. c	6. a
2. b	7. d
3. c	8. d
4. b	9. b
5. a	10. c