Chapter No:2 Number System, Operation and Codes

CHAPTER OUTLINE

- Decimal Numbers
- Binary Numbers
- Decimal-to-Binary Conversion
- Binary Arithmetic
- Complements of Binary Numbers
- Signed Numbers
- Arithmetic Operations with Signed Numbers
- Hexadecimal Numbers
- Octal Numbers
- Binary Coded Decimal (BCD)
- Digital Codes
- Error Codes

DECIMAL NUMBERS

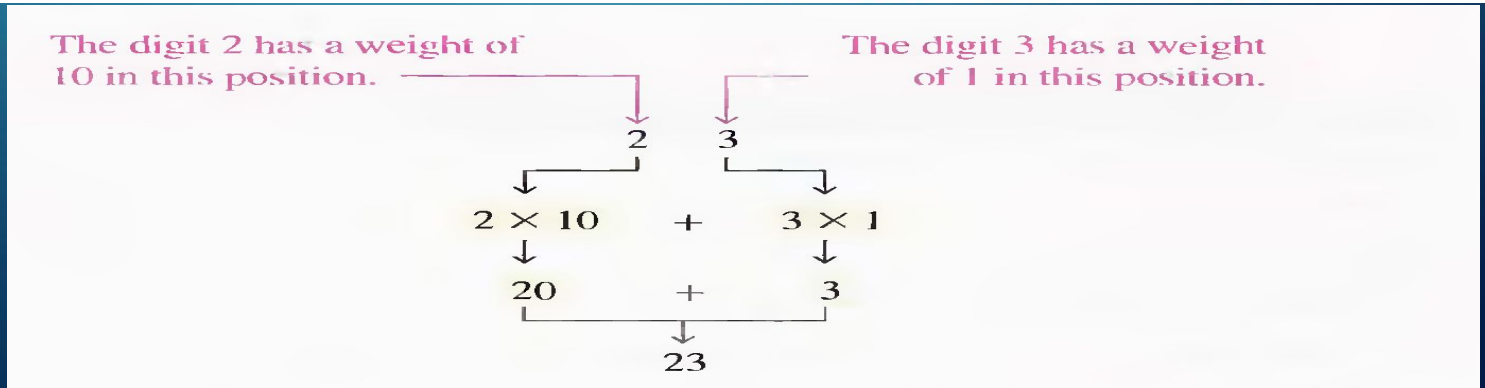
The position of each digit in a weighted number system is assigned a weight based on the **base** or **radix** of the system. The radix of decimal numbers is ten, because only ten symbols (0 through 9) are used to represent any number.

The column weights of decimal numbers are powers of ten that increase from right to left beginning with $10^0 = 1$:

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

For fractional decimal numbers, the column weights are negative powers of ten that decrease from left to right:

$10^2\ 10^1\ 10^0.\ 10^{-1}\ 10^{-2}\ 10^{-3}\ 10^{-4}\ \ldots$



Decimal numbers can be expressed as the sum of the products of each digit times the column value for that digit. Thus, the number 9240 can be expressed as

$$(9 \times 10^3) + (2 \times 10^2) + (4 \times 10^1) + (0 \times 10^0)$$

or

$$9 \times 1{,}000 + 2 \times 100 + 4 \times 10 + 0 \times 1$$

Example Express the number 480.52 as the sum of values of each digit.

Solution

$$480.52 = (4 \times 10^2) + (8 \times 10^1) + (0 \times 10^0) + (5 \times 10^{-1}) + (2 \times 10^{-2})$$

BINARY NUMBERS

For digital systems, the binary number system is used. Binary has a radix of two and uses the digits 0 and 1 to represent quantities.

The column weights of binary numbers are powers of two that increase from right to left beginning with $2^0 = 1$:

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

For fractional binary numbers, the column weights are negative powers of two that decrease from left to right:

$$2^2\ 2^1\ 2^0.\ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ \ldots$$

BINARY NUMBERS

A binary counting sequence for numbers from zero to fifteen is shown.

Notice the pattern of zeros and ones in each column.

Digital counters frequently have this same pattern of digits:

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

n bits you can count up to a number equal to $2^n - 1$

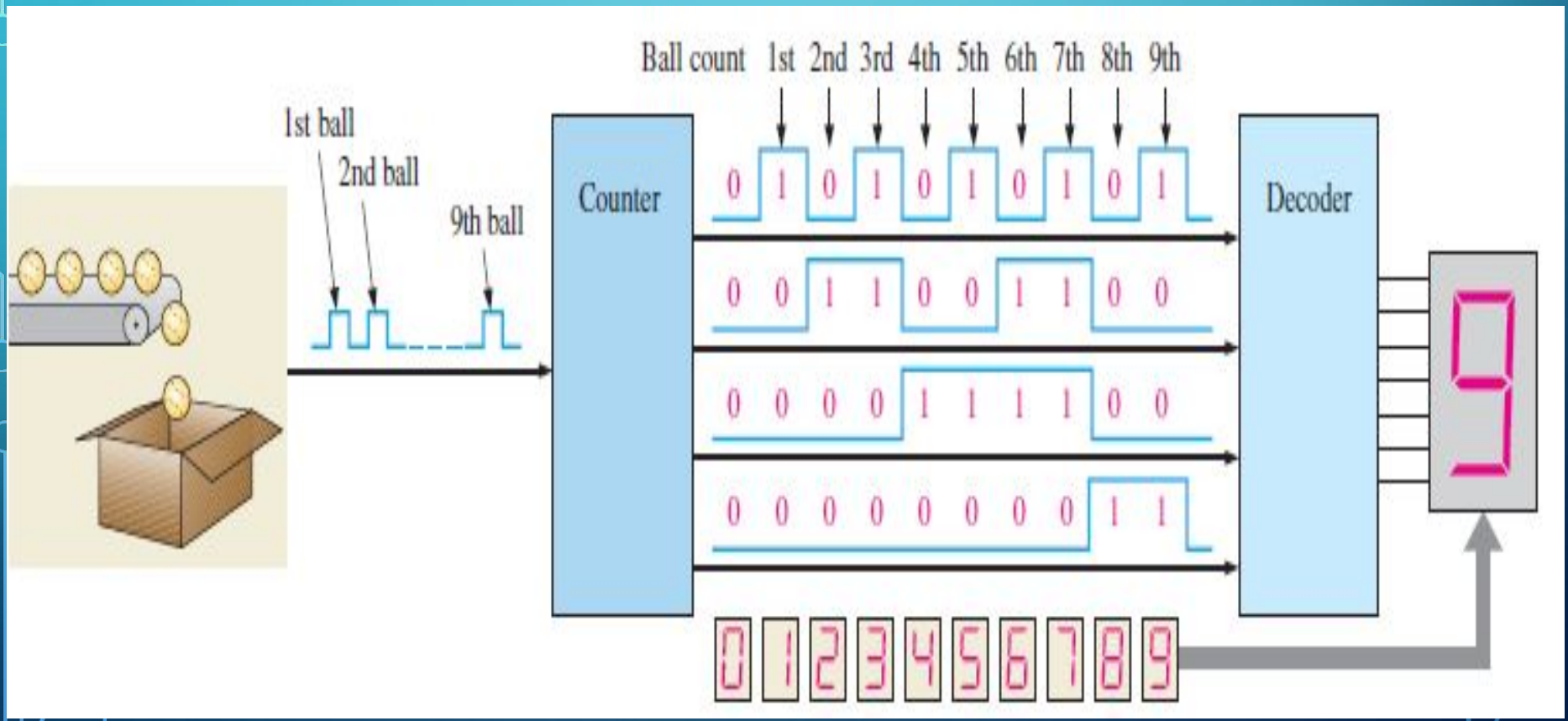
Largest decimal number $= 2^n - 1$

For example, with five bits ($n = 5$)

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

count from zero to thirty-one.

A SIMPLE BINARY COUNTING APPLICATION.



THE WEIGHTING STRUCTURE OF BINARY NUMBERS

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

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

$$\begin{array}{cccccc} 1 & 1 & 0 & 1 & 1_2 & \\ 2^4 + 2^3 + 0 + 2^1 + 2^0 & = & 16 + 8 + 2 + 1 & & & \\ & = & 27_{10} & & & \end{array}$$

Binary to Decimal Conversion

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

Binary number: 1 1 0 1 1 0 1

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

Convert the binary number 10010001 to decimal.

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

Binary number: 0 . 1 0 1 1

$$0.1011 = 2^{-1} + 2^{-3} + 2^{-4}$$
$$= 0.5 + 0.125 + 0.0625 = \mathbf{0.6875}$$

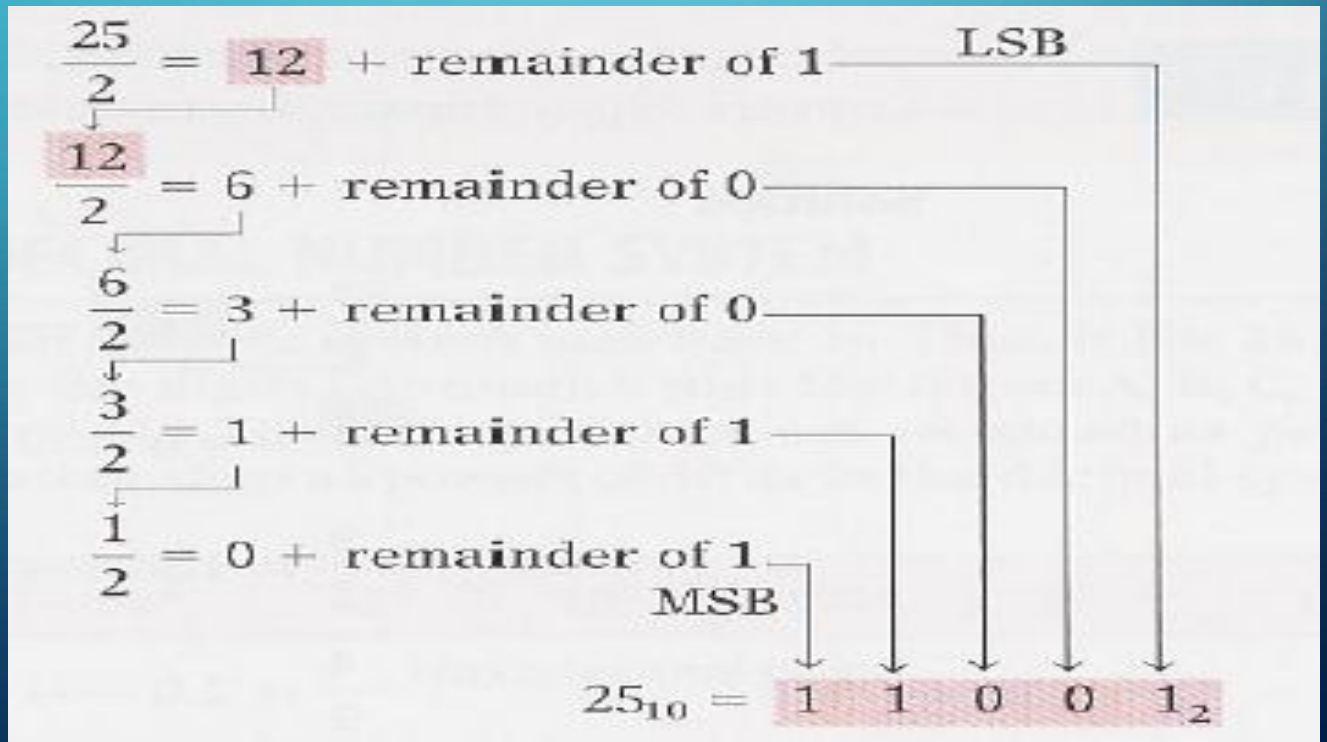
Convert the binary number 10.111 to decimal.

DECIMAL TO BINARY CONVERSION

Sum-of-Weights Method

$$45_{10} = 32 + 8 + 4 + 1 = 2^5 + 0 + 2^3 + 2^2 + 0 + 2^0$$
$$= 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1_2$$

Repeated Division-by-2 Method



DECIMAL TO BINARY CONVERSION

You can convert a decimal fraction to binary by repeatedly multiplying the fractional results of successive multiplications by 2. The carries form the binary number.

Example Convert the decimal fraction 0.188 to binary by repeatedly multiplying the fractional results by 2.

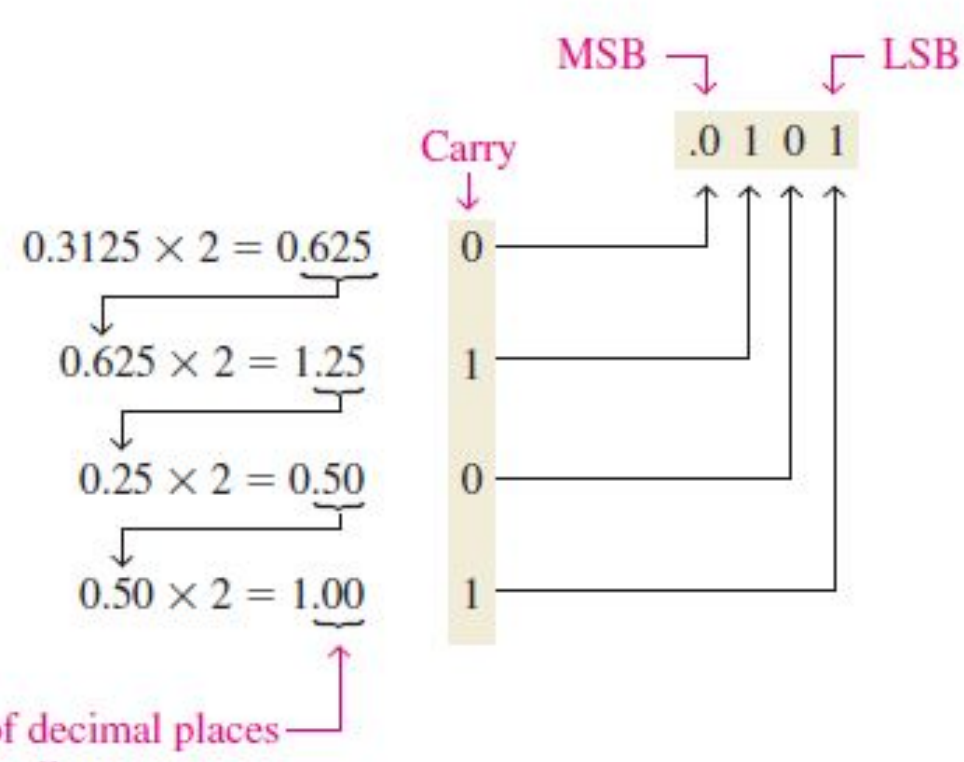
Solution

$0.188 \times 2 = 0.376$ carry = 0
$0.376 \times 2 = 0.752$ carry = 0
$0.752 \times 2 = 1.504$ carry = 1
$0.504 \times 2 = 1.008$ carry = 1
$0.008 \times 2 = 0.016$ carry = 0

MSB

Answer = .00110 (for five significant digits)

DECIMAL TO BINARY CONVERSION



Binary to Decimal Conversion

1. Convert 100011011011_2 to its decimal equivalent.
2. What is the weight of the MSB of a 16-bit number?

1. 2267 2. 32768

Decimal to Binary Conversion

1. Convert 83_{10} to binary
2. Convert 729_{10} to binary
3. How many bits are required to count up to decimal 1 million?

1. 1010011 2. 1011011001 3. 20bits

Convert the binary number 10010001 to decimal.

10010001 = 145

Convert the binary number 10.111 to decimal.

10.111 = 2.875

RULES OF 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

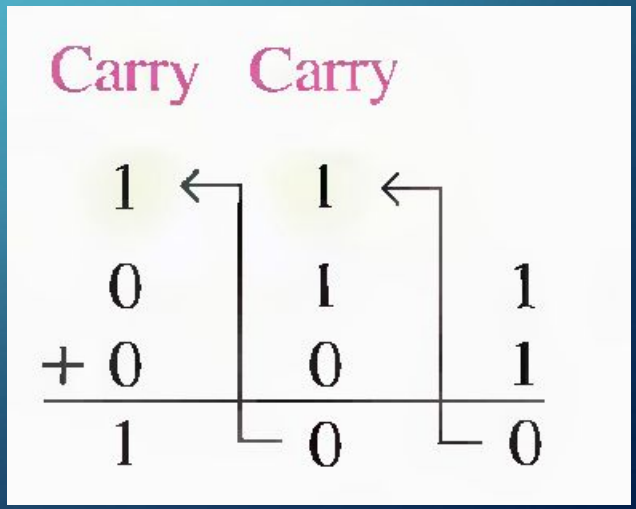
11 + 11 = 110 (3+3=6)

100+ 10 = 110 (4+2=6)

111+11 = 1010 (7+3=10)

110+100=1010 (6+4=10)

Note: (1+1+1 = 11)



RULES OF BINARY SUBTRACTION

The four basic rules for subtracting bits are as follows:

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

11-01 = 10 (3-1=2)

11-10 = 01 (3-2=1)

101 -011 = 010 (5-3=2)

11	3
− 01	− 1
10	2

101	5
−011	−3
010	2

RULES OF BINARY MULTIPLICATION

The four basic rules for multiplying bits are as follows:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

(a)

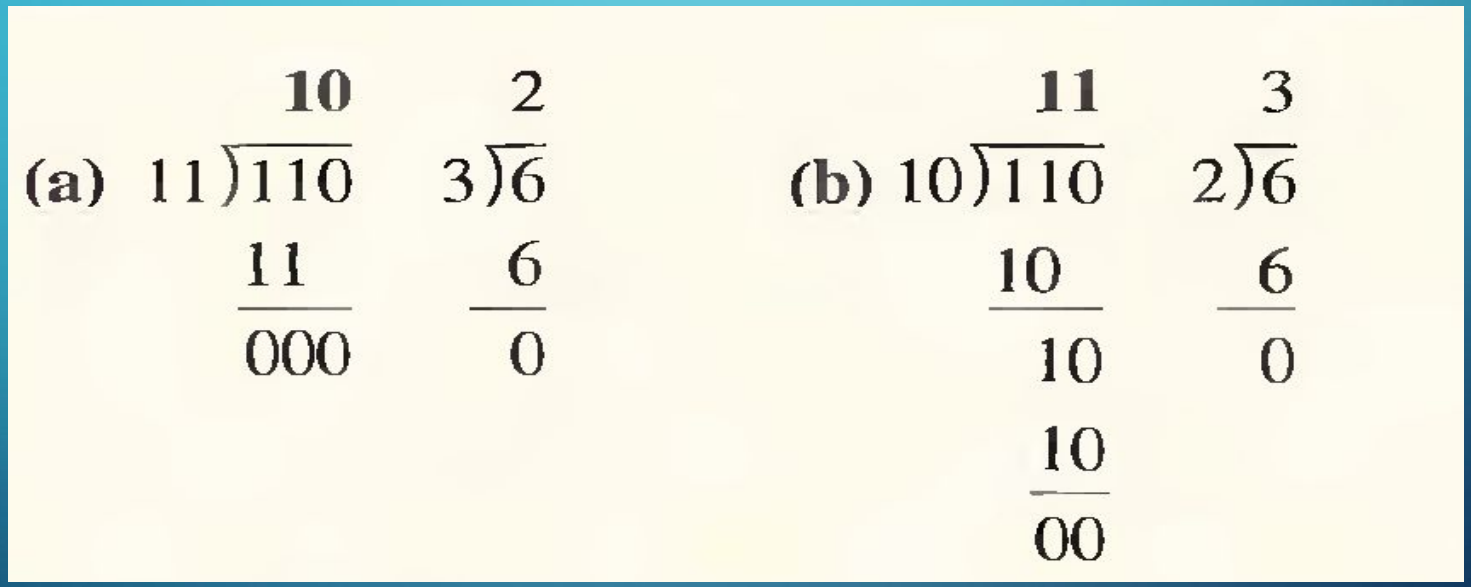
	11	3
	× 11	× 3
Partial products	11	9
	+11	
	1001	

(b)

	111	7
	× 101	× 5
Partial products	111	35
	000	
	+111	
	100011	

Rules of binary Division

Division in binary follows the same procedure as division in decimal. The equivalent decimal divisions are also given.



Examples

Perform the following binary additions:

(a) 1101 + 1010 (b) 10111 + 01101

(a) 1101 + 1010 = 10111 (b) 10111 + 01101 = 100100

Perform the following binary subtractions:

(a) 1101 − 0100 (b) 1001 − 0111

(a) 1101 − 0100 = 1001 (b) 1001 − 0111 = 0010

Perform the indicated binary operations:

(a) 110×111 (b) $1100 \div 011$

(a) $110 \times 111 = 101010$ (b) $1100 \div 011 = 100$

1' s and 2's Complements of Binary Numbers

Subtraction of a number from another can be accomplished by adding the complement of the subtrahend to the minuend.

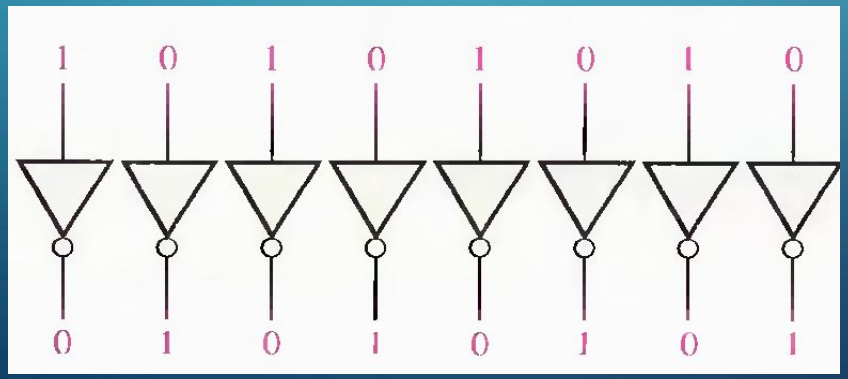
Subtraction of binary numbers using the 1's and 2's complements method allows subtraction only by addition.

1′ s 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



2's Complements of Binary Numbers

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.

2's complement = (1's complement) + 1

10110010	Binary number
01001101	1's complement
+ 1	Add 1
01001110	2's complement

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

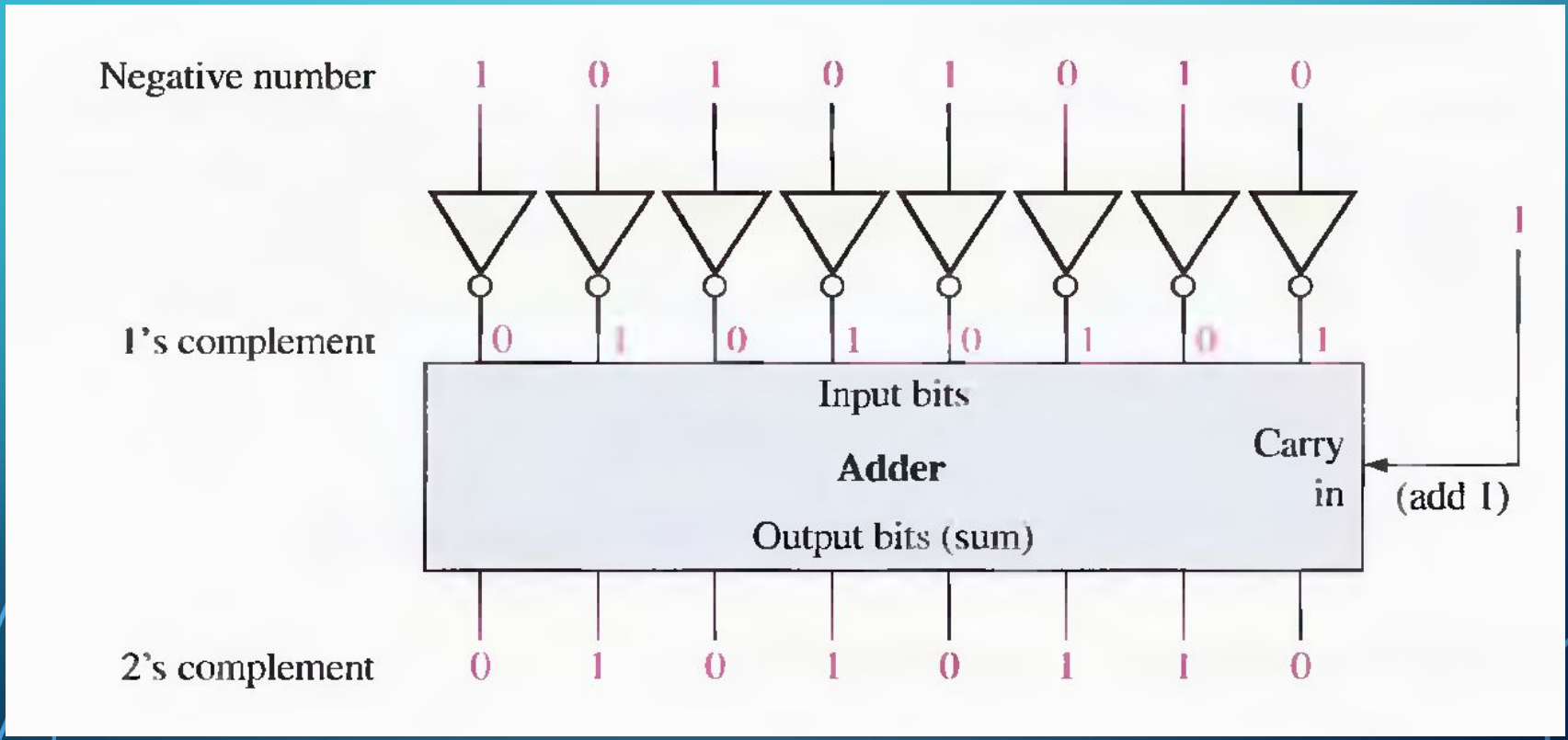
1. Start at the right with the LSB and write the bits as they are up to and including the first 1.
2. Take the 1's complements of the remaining bits.

10111000	Binary number
01001000	2's complement

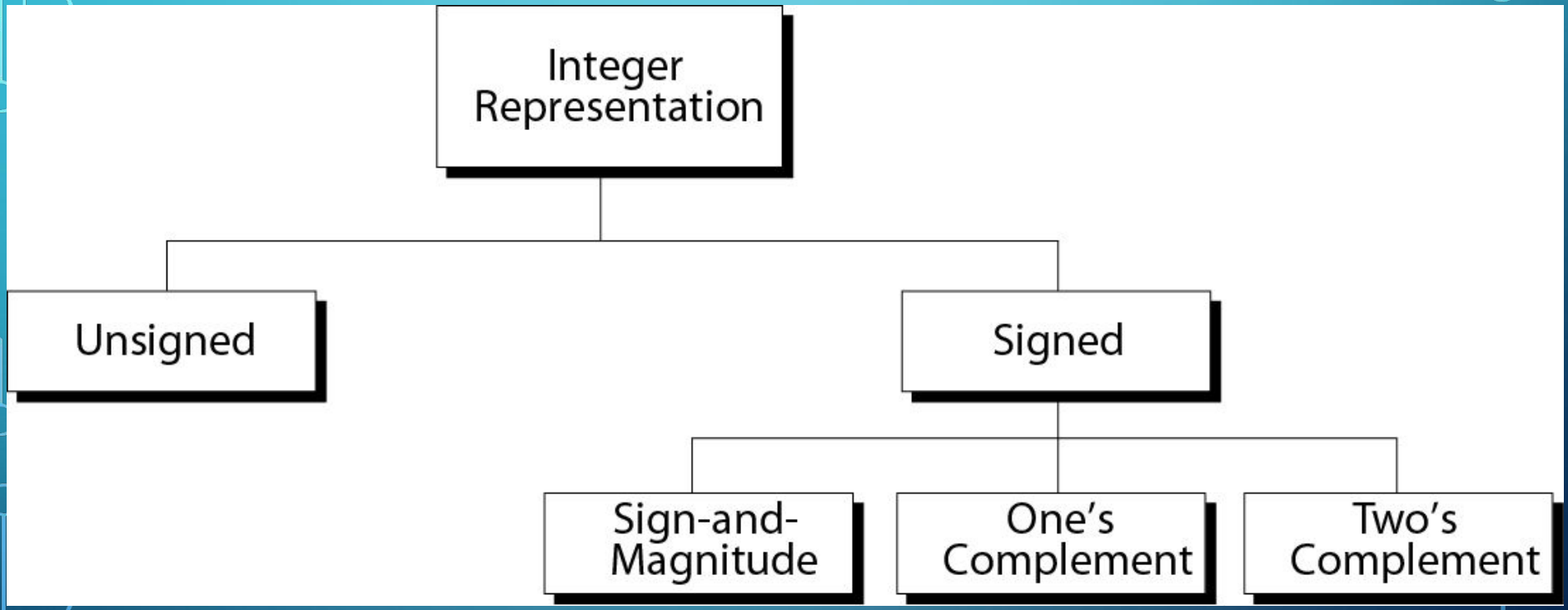
1's complements of original bits → 01001000

These bits stay the same.

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



Taxonomy of integers



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 indicates whether a number is positive or negative, and the magnitude is the value of the number. There are three forms in which signed integer (whole) numbers can be represented in binary:

- *Sign-magnitude,*
- *1's complement, and*
- *2' complement.*

The Sign Bit

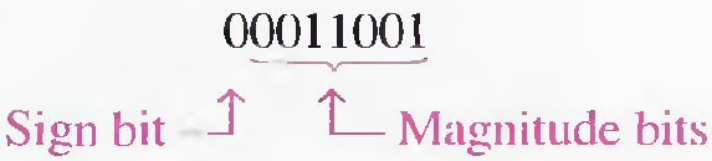
The left-most bit in a signed binary number is the **sign bit,** which tells you whether the number is positive or negative.

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

Signed Numbers

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

Signed Numbers
1's and 2's Complement forms

Positive numbers in I's and 2's complement forms are represented the same way as the positive sign- magnitude numbers. Negative numbers, however, are the l's and 2's complements of the corresponding positive numbers.

1's Complement :

The decimal number -25 is expressed as the I' s complement of + 25 (0001100 I) as 11100110

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

2's Complement :

The decimal number -25 is expressed as the 2' s complement of + 25 (0001100 I) as 11100111

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

Examples (Signed Numbers)

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

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

Examples (Signed Numbers)

Express + 19 and - 19 in sign-magnitude, I's complement, and 2's complement.

	SIGN-MAGNITUDE	1'S COMP	2'S COMP
+19	00010011	00010011	00010011
−19	10010011	11101100	11101101

The Decimal Value of Signed Numbers

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

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

$$16 + 4 + 1 = 21$$

Summing the weights where there are 1s,

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

Determine the decimal value of the sign-magnitude number 01110111

01110111 = +119 $_{10}$

The Decimal Value of Signed Numbers

1's Complement

Determine the decimal values of the signed binary numbers expressed in 1's complement: 00010111

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,

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

The Decimal Value of Signed Numbers

1's Complement

Determine the decimal values of the signed binary numbers expressed in 1's complement: 11101000

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

$$-24 + 1 = -23$$

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

The Decimal Value of Signed Numbers

2's Complement

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

(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,

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

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

Summary of integer representation

Contents of Memory	Unsigned	Sign-and-Magnitude	One's Complement	Two's Complement
0000	0	+0	+0	+0
0001	1	+1	+1	+1
0010	2	+2	+2	+2
0011	3	+3	+3	+3
0100	4	+4	+4	+4
0101	5	+5	+5	+5
0110	6	+6	+6	+6
0111	7	+7	+7	+7
1000	8	−0	−7	−8
1001	9	−1	−6	−7
1010	10	−2	−5	−6
1011	11	−3	−4	−5
1100	12	−4	−3	−4
1101	13	−5	−2	−3
1110	14	−6	−1	−2
1111	15	−7	−0	−1

Range of unsigned integers

# of Bits	*Range*	
8	0	255
16	0	65,535

Range of sign-and-magnitude integers

# of Bits	*Range*			
8	−127	−0	+0	+127
16	−32767	−0	+0	+32767
32	−2,147,483,647	−0	+0	+2,147,483,647

Note:

There are two 0s in one's complement

representation: positive and negative.

In an 8-bit allocation:

+0 □ 00000000

-0 □ 11111111

Range of two's complement integers

# of Bits	*Range*		
8	−128	0	+127
16	−32,768	0	+32,767
32	−2,147,483,648	0	+2,147,483,647

The Decimal Value of Signed Numbers
1's and 2's Complement

From these examples, you can see why the 2's complement form is preferred for representing signed integer numbers: To convert to decimal, it simply requires a summation of weights regardless of whether the number is positive or negative. The 1's complement system requires adding 1 to the summation of weights for negative numbers but not for positive numbers. Also, the 1's complement form is generally not used because two representations of zero (00000000 or 11111111) are possible.

ADD/SUB : 4 COMBINATIONS

Positive / Positive
Positive Answer

$$\begin{array}{r} 9 \\ +\ 5 \\ \hline 14 \end{array}$$

Negative / Positive
Negative Answer

$$\begin{array}{r} (-9) \\ +\ 5 \\ \hline -\ 4 \end{array}$$

Positive / Negative
Positive Answer

$$\begin{array}{r} 9 \\ +\ (-5) \\ \hline 4 \end{array}$$

Negative / Negative
Negative Answer

$$\begin{array}{r} (-9) \\ +\ (-5) \\ \hline -\ 14 \end{array}$$

POSITIVE / POSITIVE COMBINATION

Positive / Positive
Positive Answer

9	⟶	00001001
+ 5	⟶	+ 00000101
14	⟵	00001110

Both Positive Numbers
Use Straight Binary Addition

POSITIVE / NEGATIVE COMBINATION

Positive / Negative
Positive Answer

$$9 \longrightarrow 00001001$$
$$+\ (-5) \quad +\ 11111011$$
$$4 \longleftarrow 1]00000100$$

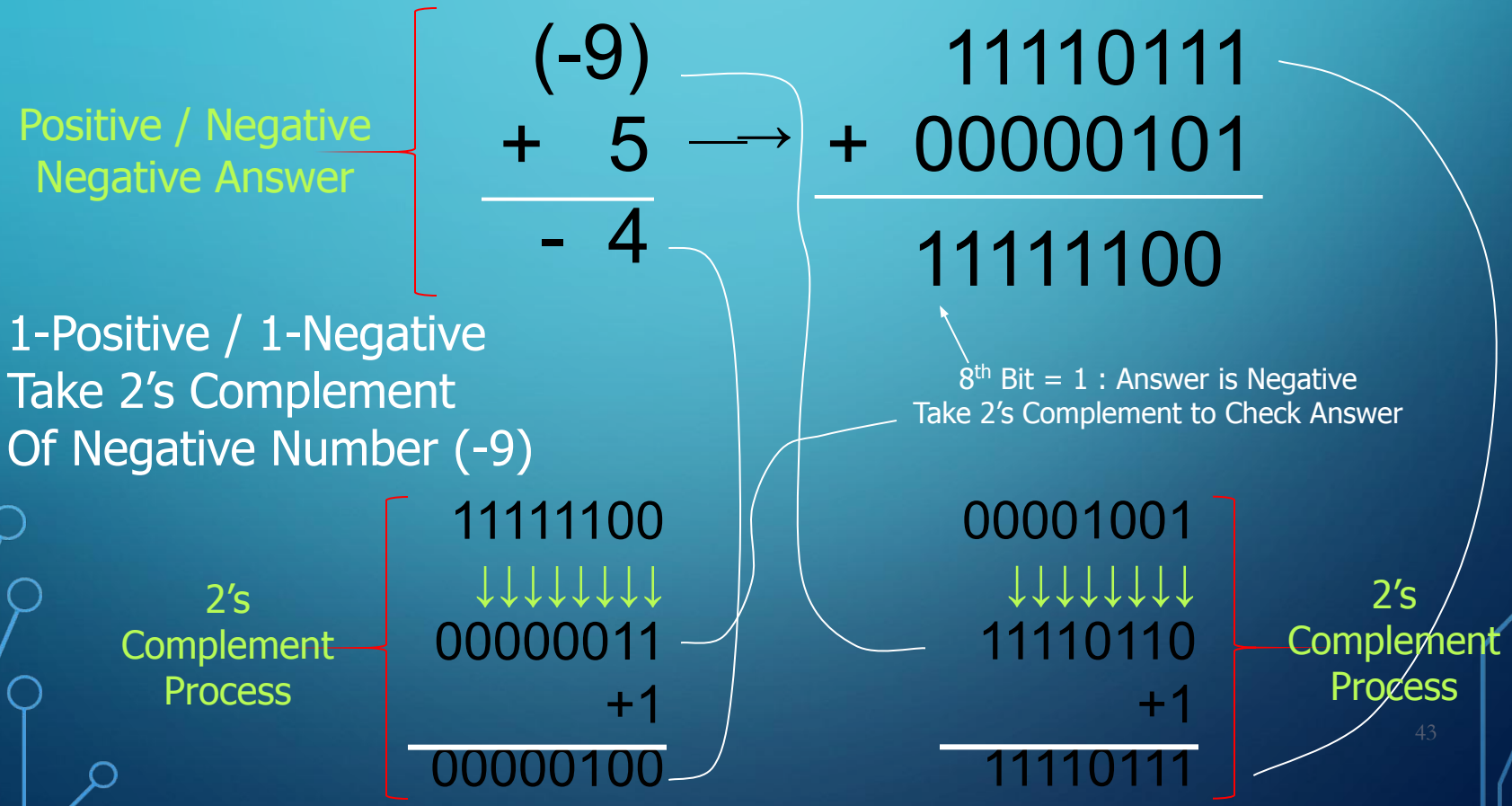
8th Bit = 0 : Answer is Positive
Disregard 9th Bit

1-Positive / 1-Negative
Take 2's Complement
Of Negative Number (-5)

$$00000101$$
$$\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$$
$$11111010$$
$$+1$$
$$11111011$$

2's Complement Process

NEGATIVE / POSITIVE COMBINATION



NEGATIVE / NEGATIVE COMBINATION

Negative / Negative
Negative Answer

$$\begin{array}{rcl} (-9) & \longrightarrow & 11110111 \\ +\ (-5) & \longrightarrow & +\ 11111011 \\ \hline -\ 14 & & \cancel{1}]11110010 \end{array}$$

2's Complement Numbers, See Conversion Process In Previous Slides

8^{th} Bit = 1 : Answer is Negative
Disregard 9^{th} Bit
Take 2's Complement to Check Answer

2-Negative
Take 2's Complement Of
Both Negative Numbers

2's Complement Process

$$\begin{array}{r} 11110010 \\ \downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow \\ 00001101 \\ +1 \\ \hline 00001110 \end{array}$$

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. The following 8-bit example will illustrate this condition.

01111101	125
+ 00111010	+ 58
10110111	183

Sign incorrect

Magnitude incorrect

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

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

Subtraction (Examples)

Perform each of the following subtractions of the signed numbers:

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

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)

Signed 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

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

$$8 + 8 + 8 = 24$$

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.

Signed Multiplication

$13 \times 11 = 143$

				1	1	0	1
			x	1	0	1	1
				1	1	0	1
			1	1	0	1	
		0	0	0	0		
+	1	1	0	1			
1	0	0	0	1	1	1	1

Signed Multiplication

The following example shows signed 2's complement representation can be used to represent negative operands as well as positive ones in multiplication(using 6bit representation).

Example: $(-5) \times (-4) = 20$

$4 : 000100; \quad -4 : 111100; \quad 5 : 000101; \quad -5 : 111011$

					1	1	1	0	1	1
				x	1	1	1	1	0	0
1	1	1	1	1	0	0	1	1		
1	1	1	1	0	0	1	1			
1	1	1	0	0	1	1				
1	1	0	0	1	1					
.	.	.	.	.	0	1	0	1	0	0

The last 6 bits of the result are 010100, representing a positive product $010100_2 = 20_{10}$.

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
100110010001	Sum
+ 0000000	7th partial product
100110010001	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.

$$100110010001 \longrightarrow 011001101111$$

Attach the sign bit →

1 011001101111

Signed 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 division operation in computers is accomplished using subtraction. Since subtraction is done with an adder, division can also be accomplished with an adder.

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

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.

HEXADECIMAL NUMBER SYSTEM

The **hexadecimal number system** uses base 16. Thus, it has 16 possible digit symbols. It uses the digits 0 through 9 plus the letters A, B, C, D, E, and F as

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

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.

(a) $\underbrace{1100}_{\downarrow C}\underbrace{1010}_{\downarrow A}\underbrace{0101}_{\downarrow 5}\underbrace{0111}_{\downarrow 7} = CA57_{16}$

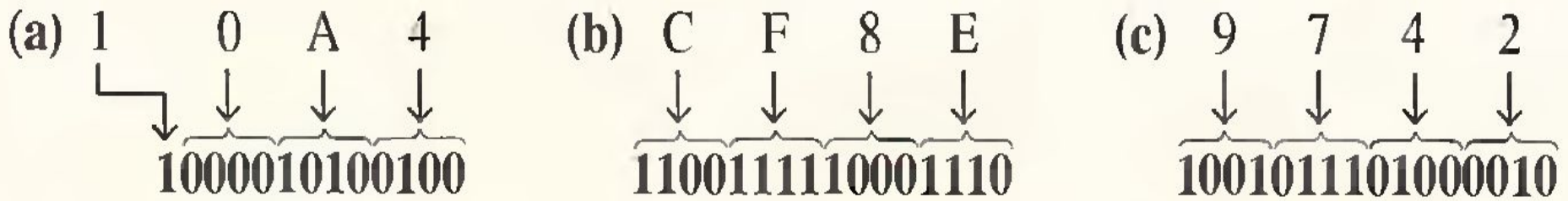
(b) $\underbrace{0011}_{\downarrow 3}\underbrace{1111}_{\downarrow F}\underbrace{0001}_{\downarrow 1}\underbrace{0110}_{\downarrow 6}\underbrace{1001}_{\downarrow 9} = 3F169_{16}$

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

Convert the binary number 100111101111001100 to hexadecimal.

$4F79C_{16}$

Hexadecimal-to-Binary Conversion



Convert the hexadecimal number 6BD3 to binary.

0110101111010011_2

Hexadecimal-to-Decimal Conversion

(a) 1 C

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

(b) A 8 5

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

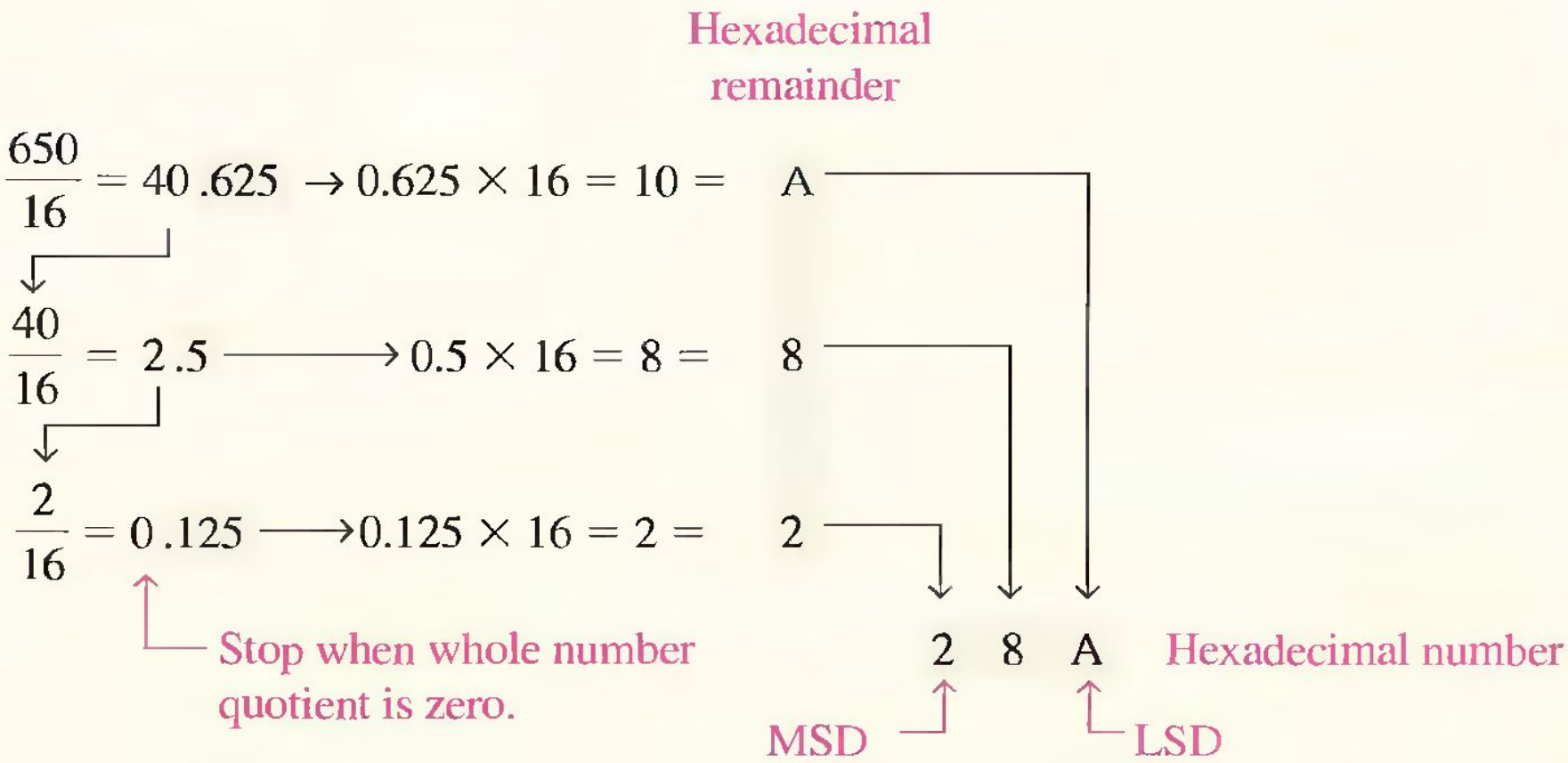
Another way to convert a hexadecimal number to its decimal equivalent is

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

$$\begin{aligned} 356_{16} &= 3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0 \\ &= 768 + 80 + 6 \\ &= 854_{10} \end{aligned}$$

$$\begin{aligned} 2AF_{16} &= 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\ &= 512 + 160 + 15 \\ &= 687_{10} \end{aligned}$$

Decimal-to-Hexadecimal Conversion



USEFULNESS OF HEX

Hex is often used in a digital system as sort of a "Shorthand" way to represent string of bits.

When dealing with the large numbers of bits , it is more convenient and less error to write the binary in hex.

Would you rather check 50 numbers like this one 0110111001100111

or 50 numbers like this one 6E67 ?

Digital circuit work in binary.

Hex is simply used as a convenience for

Octal Numbers

Octal uses eight characters the numbers 0 through 7 to represent numbers. There is no 8 or 9 character in octal.

Binary number can easily be converted to octal by grouping bits 3 at a time and writing the equivalent octal character for each group.

Example Express $1\ 001\ 011\ 000\ 001\ 110_2$ in octal:

Solution Group the binary number by 3-bits starting from the right. Thus, 113016_8

Decimal	Octal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111

Octal Numbers

Octal is also a weighted number system. The column weights are powers of 8, which increase from right to left.

Column weights	8^3	8^2	8^1	8^0	.
	512	64	8	1	.

Example Express 3702_8 in decimal.

Solution Start by writing the column weights:

512 64 8 1

3 7 0 2_8

$3(512) + 7(64) + 0(8) + 2(1) = 1986_{10}$

Decimal	Octal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111

Code:

When the numbers , letters or word are represented by a special group of symbols, we say they are being encoded, and the group of symbols is called a Code.

Straight binary Code:

When a decimal number is represented by its equivalent binary number, called Straight binary Code.

BCD Code:

If each digit of a decimal number is represented by its binary equivalent, the result is a code called Binary –Coded –Decimal (BCD)

Since a decimal digit can be as large as 9, four bits are required to code each digit .

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

BINARY CODED DECIMAL

8	7	4	(decimal)
↓	↓	↓	
1000	0111	0100	(BCD)

0110	1000	0011	1001
6	8	3	9

If any of the "forbidden" four bit numbers ever occurs in a machine using the BCD code, it is usually an indication that an error has occurred.

0111	1100	0001
7	↓	1

The forbidden code group indicates an error in the BCD number

COMPARISON OF BCD AND BINARY

BCD is not another number system like binary , octal and hexadecimal .

It is the decimal system with each digit encoded in its binary equivalent.

$137_{10} = 10001001_2$ (binary)
$137_{10} = 0001\ 0011\ 0111$ (BCD)

Advantage:

Easy conversion

Disadvantage:

BCD required more bits

Example:	*Decimal*	*Binary*	*BCD*
	11	***1011***	***0001 0001***

ADDITION OF BCD NUMBERS

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)
$$\begin{array}{r} 0011 \\ +\,0100 \\ \hline \mathbf{0111} \end{array} \qquad \begin{array}{r} 3 \\ +\,4 \\ \hline 7 \end{array}$$

(b)
$$\begin{array}{rr} 0010 & 0011 \\ +\,0001 & 0101 \\ \hline \mathbf{0011} & \mathbf{1000} \end{array} \qquad \begin{array}{r} 23 \\ +\,15 \\ \hline 38 \end{array}$$

(c)
$$\begin{array}{rr} 1000 & 0110 \\ +\,0001 & 0011 \\ \hline \mathbf{1001} & \mathbf{1001} \end{array} \qquad \begin{array}{r} 86 \\ +\,13 \\ \hline \mathbf{99} \end{array}$$

(d)
$$\begin{array}{rrr} 0100 & 0101 & 0000 \\ +\,0100 & 0001 & 0111 \\ \hline \mathbf{1000} & \mathbf{0110} & \mathbf{0111} \end{array} \qquad \begin{array}{r} 450 \\ +\,417 \\ \hline \mathbf{867} \end{array}$$

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.

Addition of BCD numbers

NOTE THAT IN EACH CASE THE SUM IN ANY 4 BIT COLUMN DOES NOT EXCEED 9, AND THE RESULT ARE INVALID BCD NUMBERS.

Add the following BCD numbers:

(a) 1001 + 0100

(b) 1001 + 1001

(c) 00010110 + 00010101

(d) 01100111 + 01010011

Addition of BCD numbers

The decimal number additions are shown for comparison.

(a)

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

(b)

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

Addition of BCD numbers

(c)

0001	0110		16
+ 0001	0101		+ 15
0010	1011	Right group is invalid (>9), left group is valid.	31
	+ 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	Both groups are invalid (>9)	120
	+ 0110	+ 0110	Add 6 to both groups	
0001	0010	0000	Valid BCD number	
↓	↓	↓		
1	2	0		

Related Problem

Add the BCD numbers: 01001000 + 00110100.

GRAY CODE:

The bits in a binary count sequence, it is clear that there are often several bits that must change states at the same time. Consider when 3 - bit binary number for 3 change to 4 : all three bits must change state.

011 (decimal 3)

100 (decimal 4)

In order to reduce the likelihood of a digital circuit misinterpreting a changing input , the Gray code has been developed as way to represent a sequence of numbers.

The unique aspect of the Gray code is that only one bit ever changes between two successive numbers in sequence.

The Gray code is also known as The Reflected Code

THREE BIT BINARY AND GRAY CODE EQUIVALENTS

B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

AN APPLICATION
THE CONCEPT OF A 3-BIT SHAFT

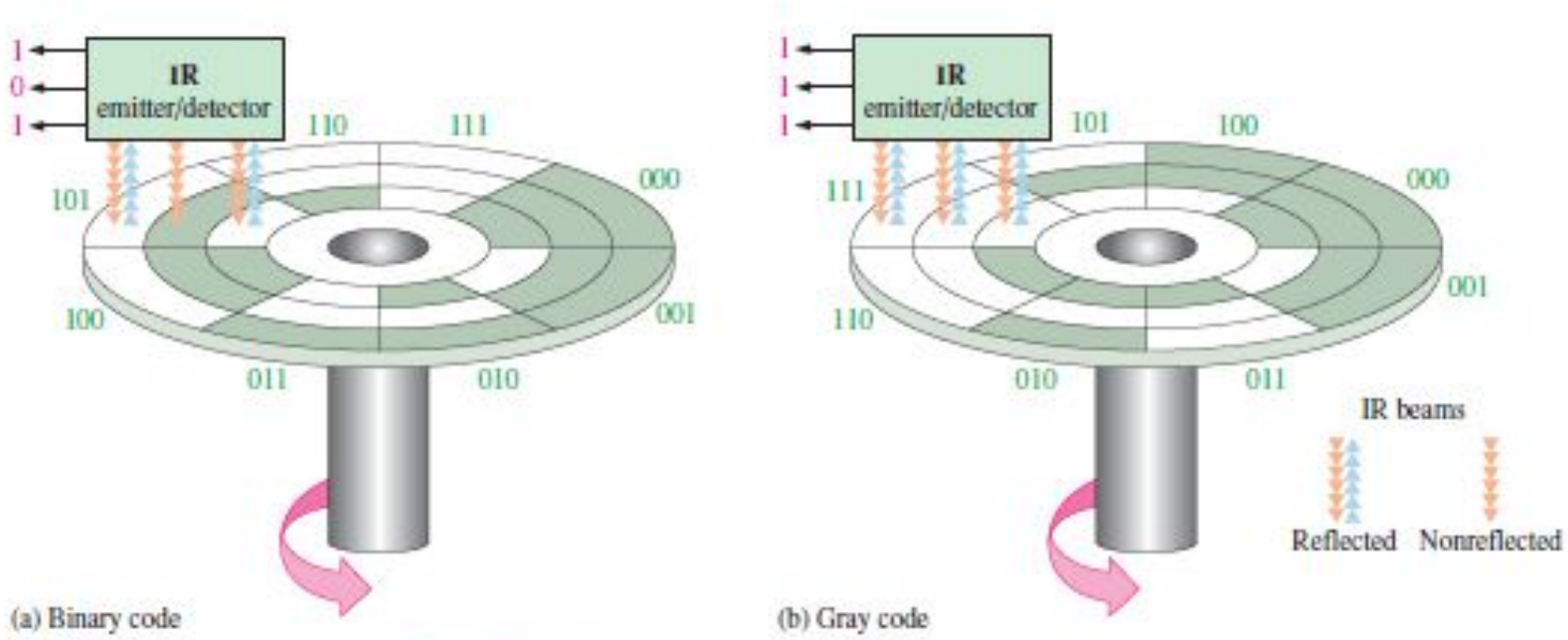
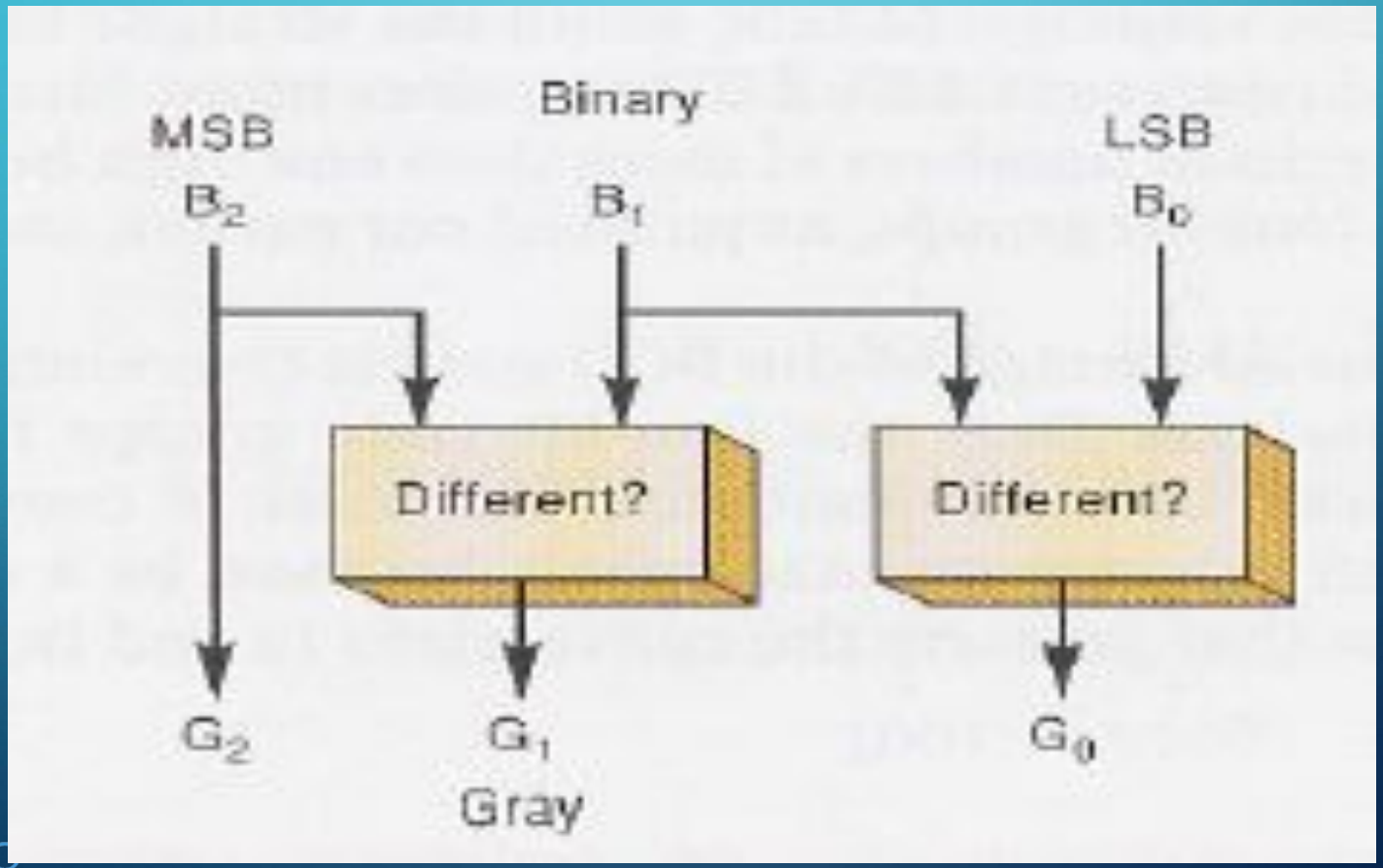


FIGURE 2–7 A simplified illustration of how the Gray code solves the error problem in shaft position encoders. Three bits are shown to illustrate the concept, although most shaft encoders use more than 10 bits to achieve a higher resolution.

BINARY TO GRAY CODE CONVERSION

- *MSB Binary use as Gray MSB*
- *Compare MSB Binary with next binary bit (B1)*
- *If they are same G1 = 0*
- *If they are different G1 = 1*
- *G0 can be found by comparing B1 and B0*

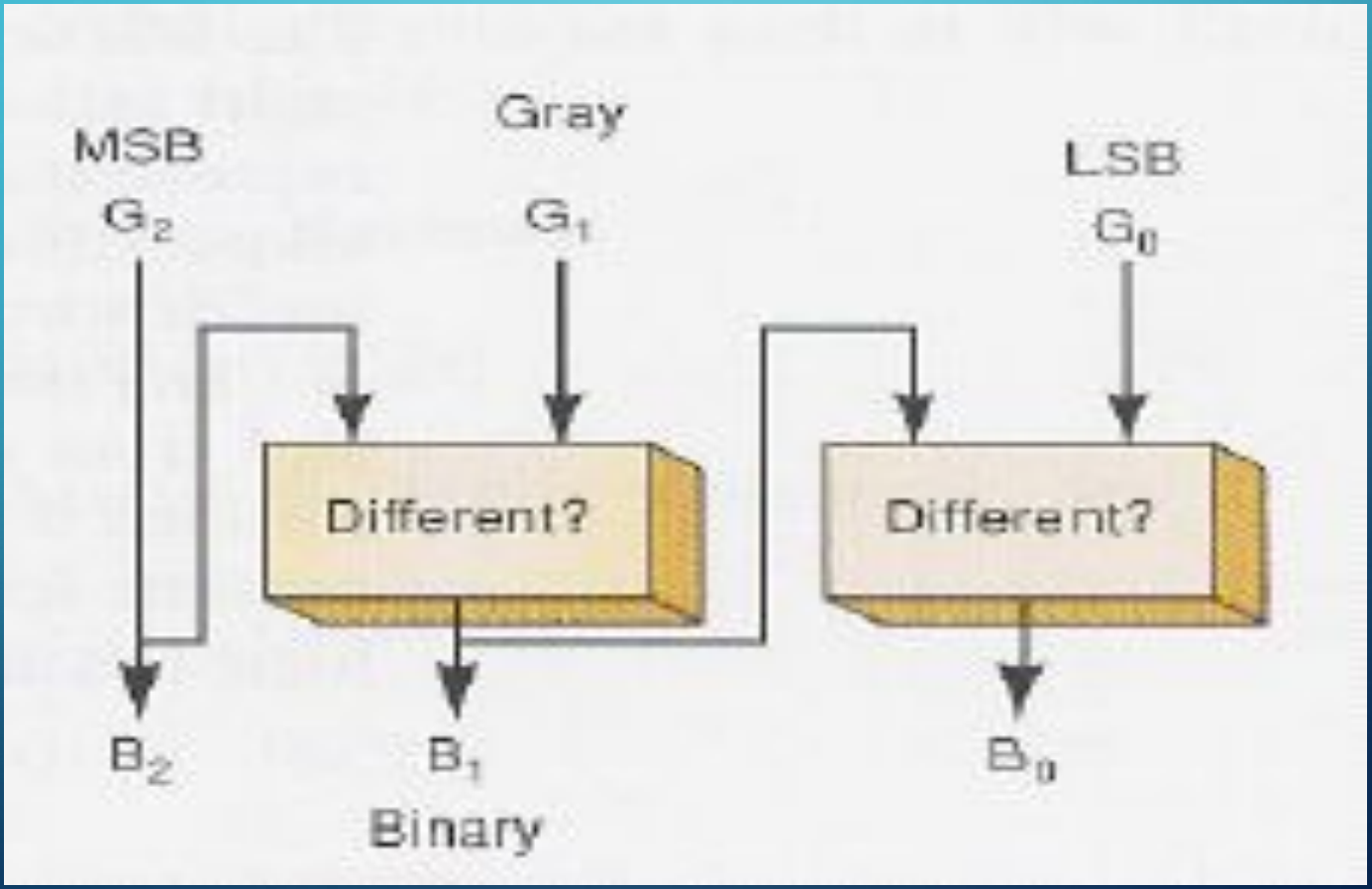
CONVERTING BINARY TO GRAY CODE



GRAY CODE TO BINARY CONVERSION

- *Gray MSB used as MSB binary*
- *Compare Gray MSB with binary bit (G1)*
- *If they are same B1 = 0*
- *If they are different B1 = 1*
- *B0 can be found by comparing B1 and G0*

CONVERTING GRAY TO BINARY CODE



PUTTING ALL TOGETHER

Decimal	Binary	Hexadecimal	BCD	GRAY
0	0	0	0000	0000
1	1	1	0001	0001
2	10	2	0010	0011
3	11	3	0011	0010
4	100	4	0100	0110
5	101	5	0101	0111
6	110	6	0110	0101
7	111	7	0111	0100
8	1000	8	1000	1100
9	1001	9	1001	1101
10	1010	A	0001 0000	1111
11	1011	B	0001 0001	1110
12	1100	C	0001 0010	1010
13	1101	D	0001 0011	1011
14	1110	E	0001 0100	1001
15	1111	F	0001 0101	1000

THE BYTE , NIBBLE AND WORD

The Byte :

A string of eight bits is called a Byte .

The Nibble :

A group of four bit(Half of a byte) is called a Nibble.

The Word :

A word is a group of bits that represent s a certain unit of information

ALPHANUMERIC CODES:

The codes that represent letters of the alphabet , punctuation marks, and other special characters as well as numbers are called Alphanumeric Codes.

- *26 lowercase letter*
- *26 uppercase letter*
- *10 numeric digits*
- *7 punctuation marks*
- *20 to 40 other characters such as + , / , %, * and so on*

AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE(ASCII)CODE:

ASCII is widely used alphanumeric code.

ASCII is a seven –bits code (2^7 =128 possible code group)

ASCII code for each character can be obtain by converting the hex value to binary.

Character	ACSII code	Hex	Decimal
A	100 0001	41	65
B	100 0010	42	66
0	011 0000	30	48

AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII)CODE:

The ASCII code is used for the transfer of alphanumeric information between a computer and external devices such as printer or another computer.

Example :

The binary string that will be entered into memory when the operator types

if (x >3).

i	69	0110	1001
f	66	0110	0110
space	20	0010	0000
(	28	0010	1000
x	78	0111	1000
>	3E	0011	1110
3	33	0011	0011
)	29	0010	1001

Note that a 0 was added to the leftmost bit of each ASCII code because the codes must be stored as bytes (eight bits). This adding of an extra bit is called *padding with 0s.*

ERROR DETECTION CODES

Examples of the movement of binary data and code :

- *The transmission of digital voice over a microwave link.*
- *The storage of data in and retrieval of data from external memory devices.*
- *Sending and receiving information on the Internet.*

Whenever information transmitted errors can occur such that receiver does not receive the identical information that was sent by transmitter. The major cause of transmission error is ***electric noise*** *(fluctuation of current and voltage)*

An error detection code can be used to detect error during transmission.

Most modern digital equipment is designed to be relatively error free and the probability of error is very low

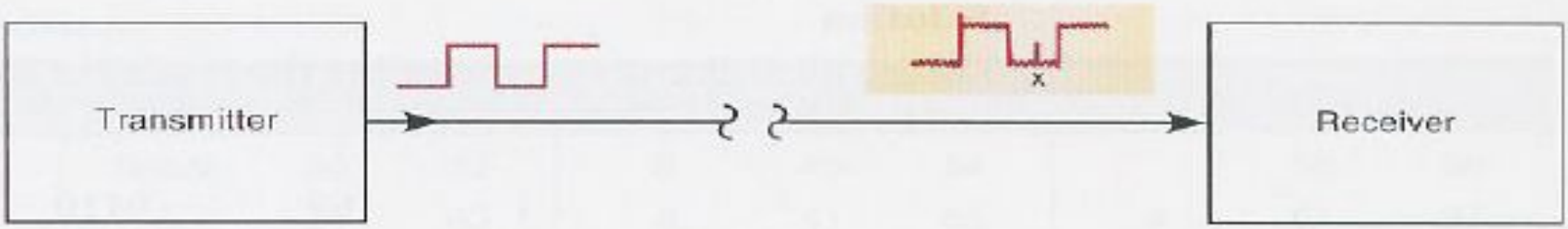


FIGURE 2-4 Example of noise causing an error in the transmission of digital data.

PARITY METHOD FOR ERROR DETECTION

*One of the simplest and most widely used schemes for error detection is the **Parity Method***

Parity Bit:

*A parity bit is an extra bit that is attached to a code group that is being transferred from one location to another. It is made either **0** or **1**.*

There are two parity methods:

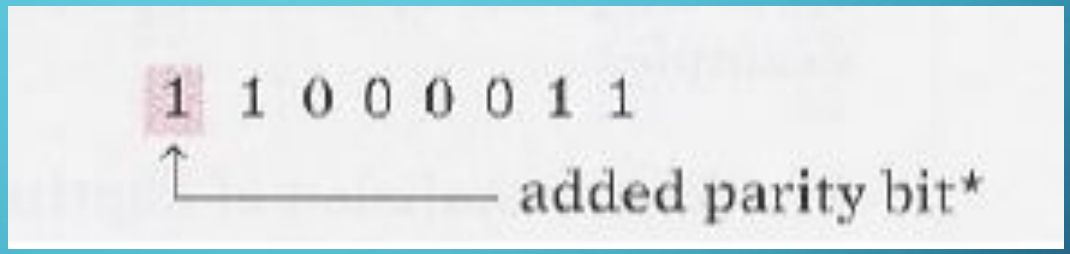
Even Parity method

Odd Parity method

EVEN PARITY METHOD :

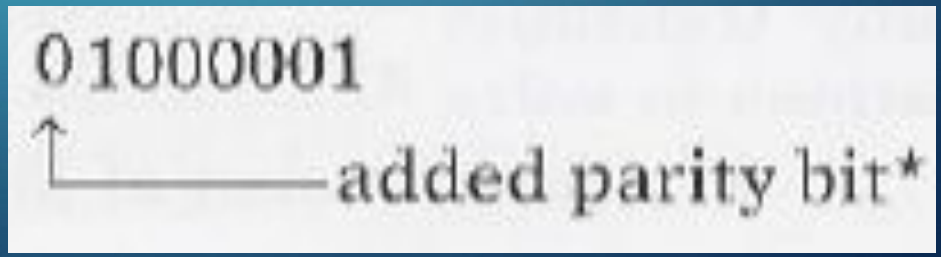
If number of 1s in the code group odd then add a parity bit of 1 .

For example:



If number of 1s in the code group even then add a parity bit of 0 .

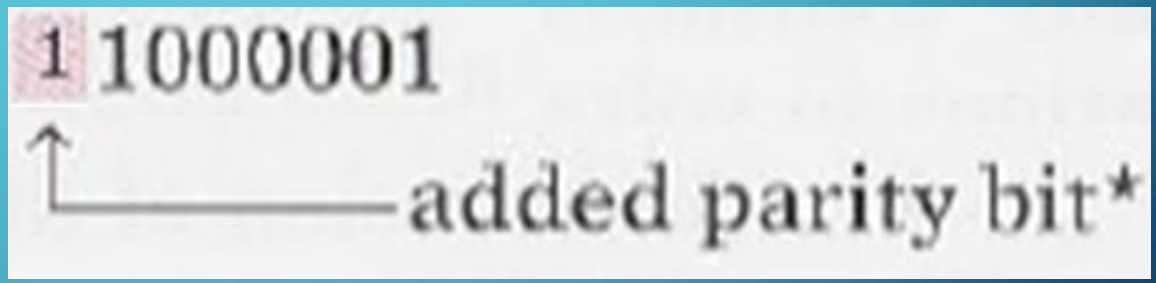
For example:



ODD PARITY METHOD :

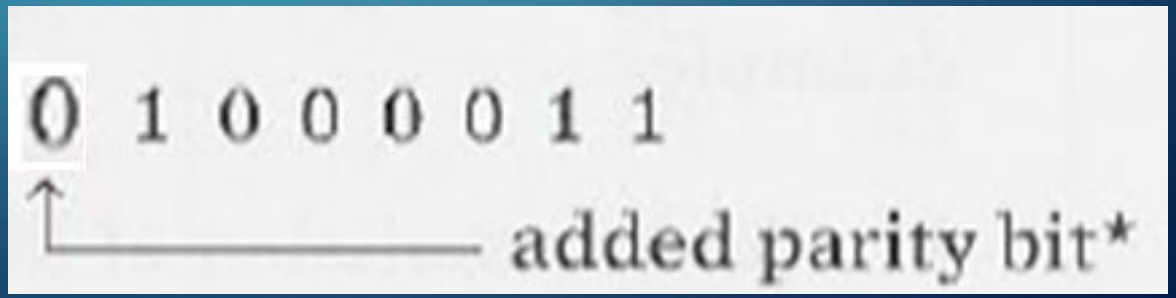
If number of 1s in the code group even then add a parity bit of 1 .

For example:



If number of 1s in the code group odd then add a parity bit of 0 .

For example:



Suppose that the character A is being transmitted and odd parity is being used .
The transmission code would be

The receiver will check that the code contain an odd number of 1s (including the parity)
If so, receiver assume that the code has been correctly receive.
Supposed Some noise change code as

The receiver will find that this code has an even number of 1s and error is detected.

There is no way that the receiver can tell which bit is in error because it does not know what the code is supposed to be .

The parity method would not work of two bits were in error .

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