CHAPTER 2

NUMBERS AND CODES

2.1 Numbers

When a number such as 101 is given, it is impossible to determine its numerical value. Some may say it is five. Others may say it is one hundred and one. Could it be sixty-five or two hundred and fifty-seven. Yes. It could be any one of those values, or it may not be any of those values at all. The expected answer may be one hundred and one from a human being. If we were all born with two hands but without any fingers, it probably would have been five. Without knowing the base or radix of a number, there is absolutely no way to determine the numerical value of 101.

The base or radix of a number system is the total number of digits that can be used to make up numbers in that system. For a decimal number, ten digits are used and its base is 10. (Note that all bases are expressed using decimal numbers.) The base is 2 for binary numbers. Table 2.1 lists five number systems with different radices. The smallest digit in a system of base R is 0, and the largest digit is (R-1). Note that A, B, C, D, E, and F are used as the six digits after 9 in hexadecimal, which are equivalent to, respectively, 10, 11, 12, 13, 14, and 15 in decimal.

Table 2.1 Example of five number systems.

System	Radix (base)	Digits
Binary	2	0, 1
Ternary	3	0, 1, 2
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

A number can be expressed in two different forms. They are known as positional representation and polynomial representation. An example is given below for a decimal number.

$$\underbrace{751.36}_{\text{Positional representation}} = \underbrace{7\times10^2 + 5\times10^1 + 1\times10^0 + 3\times10^{-1} + 6\times10^{-2}}_{\text{Polynomial representation}}$$

The point used to separate the integer part from the fractional part in the positional representation is called a radix point. It is also called a decimal point for decimal numbers.

A subscript will be used to specify the base of a number. The subscript can be omitted if there is no ambiguity about the base. In general, the two different representations of a decimal number N can be expressed as follows:

$$
\begin{aligned}
N_{10} &= (a_{n-1}a_{n-2} \;..........\; a_2a_1a_0 \,.\, a_{-1}a_{-2} \;.....\; a_{-p+1}a_{-p})_{10} \\
&= (a_{n-1} \times 10^{n-1} + a_{n-2} \times 10^{n-2} + ... + a_2 \times 10^2 + a_1 \times 10^1 \\
&\qquad + a_0 \times 10^0 + a_{-1} \times 10^{-1} + a_{-2} \times 10^{-2} + ... + a_{-p} \times 10^{-p})_{10} \\
&= (\sum_{i=-p}^{n-1} a_i \times 10^i)_{10} \qquad (2.1)
\end{aligned}
$$

For N in base R and $R \neq 10$,

$$
\begin{aligned}
N_R &= (b_{m-1}b_{m-2} \;..........\; b_2b_1b_0 \,.\, b_{-1}b_{-2} \;.....\; b_{-q+1}b_{-q})_R \\
&= (b_{m-1} \times 10^{m-1} + b_{m-2} \times 10^{m-2} + ... + b_2 \times 10^2 + b_1 \times 10^1 \\
&\qquad + b_0 \times 10^0 + b_{-1} \times 10^{-1} + b_{-2} \times 10^{-2} + ... + b_{-q} \times 10^{-q})_R \\
&= (\sum_{i=-q}^{m-1} b_i \times 10^i)_R \qquad (2.2)
\end{aligned}
$$

Note that m and q, the numbers of digits in the integer and fraction of N in base R, may or may not equal n and p respectively. Also, $0 \leq b_i \leq (R-1)$ and "10" in Equation (2.2) is the integer number right after (R-1). Thus $10_R = R_{10}$.

2.2 Number Conversions

Various methods that can be used to convert a number from one system to another are introduced in this section.

2.2.1 Conversion from Non-Decimal to Decimal

To convert a non-decimal number N_R to N_{10}, N_R is first expressed in polynomial form using Equation (2.2). Then all numbers and digits in the polynomial are converted to decimal. Computation can then be carried out to obtain N_{10}. An example to convert an octal number to decimal is shown below.

❖ Example 2.1

To convert $(356.1)_8$ to decimal, express the number using Equation (2.2).

$$(356.1)_8 = (3 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 + 1 \times 10^{-1})_8$$

Since $10_8 = 8_{10}$ and all the octal digits are the same as the decimal digits, the polynomial can be readily converted to

$$(3 \times 8^2 + 5 \times 8^1 + 6 \times 8^0 + 1 \times 8^{-1})_{10} = (238.125)_{10}$$

This example shows that the conversion can be performed directly using the following equation, which is called polynomial substitution.

$$\begin{aligned} N_R &= (b_{m-1}b_{m-2} \,\ldots\ldots\ldots\, b_2b_1b_0 \,.\, b_{-1}b_{-2} \ldots\ldots b_{-q+1}b_{-q})_R \\ &= (b_{m-1} \times R^{m-1} + b_{m-2} \times R^{m-2} + \ldots\ldots\ldots + b_2 \times R^2 + b_1 \times R^1 \\ &\quad + b_0 \times R^0 + b_{-1} \times R^{-1} + b_{-2} \times R^{-2} + \ldots\ldots + b_{-q} \times R^{-q})_{10} \end{aligned} \tag{2.3}$$

Three more examples are given below to show the conversion using Equation (2.3).

❖ Example 2.2

$$(130.2)_4 = (1 \times 4^2 + 3 \times 4^1 + 0 \times 4^0 + 2 \times 4^{-1})_{10} = (28.5)_{10}$$

❖ Example 2.3

$$(1101.01)_2 = (1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2})_{10} = (13.25)_{10}$$

❖ Example 2.4

This example shows the conversion for R > 10.

$$\begin{aligned} (15C.F)_{16} &= (1 \times 10^2 + 5 \times 10^1 + C \times 10^0 + F \times 10^{-1})_{16} \\ &= (1 \times 16^2 + 5 \times 16^1 + 12 \times 16^0 + 15 \times 16^{-1})_{10} = (348.9375)_{10} \end{aligned}$$

It is seen that Equation (2.3) is still applicable, except that when any digit b_i is greater than 10 in the polynomial, it should be replaced with its decimal equivalent. As in Example 2.4, C and F in hexadecimal are converted respectively to 12 and 15 in decimal.

❖ Example 2.5

$$(120.2)_3 = (1 \times 3^2 + 2 \times 3^1 + 0 \times 3^0 + 2 \times 3^{-1})_{10} = (15.666\ldots)_{10}$$

This example shows that the conversion of the fraction may not have a finite number of decimal digits.

2.2.2 Conversion from Decimal to Non-Decimal

The method of polynomial substitution used in converting a non-decimal number to a decimal number as shown in Examples 2.1 to 2.5 can also be applied to the conversion of a number from decimal to a non-decimal base R. The conversion of a decimal number to base 4 is given in Example 2.6.

❖ Example 2.6

To convert $(39)_{10}$ to base 4, the number is first expressed in polynomial form.

$$(3\,9)_{10} = (3 \times 10^1 + 9 \times 10^0)_{10} \tag{2.4}$$

Since the equivalents of the ten decimal digits and the integer after decimal 9 in base 4 are

0	1	2	3	4	5	6	7	8	9	10	(for R = 10)
0	1	2	3	10	11	12	13	20	21	22	(for R = 4)

3, 9, and 10 within the parentheses on the right-hand-side of Equation (2.4) can be replaced with their equivalents in base 4, which are 3, 21, and 22 respectively.

$$(3\,9)_{10} = (3 \times 10^1 + 9 \times 10^0)_{10} = (3 \times 22^1 + 21 \times 22^0)_4 = (3 \times 22 + 21)_4$$

The multiplication of 3 × 22 and the addition to 21 are shown below using base 4 arithmetic.

$$\begin{array}{rrr} & 2 & 2 \\ \times & & 3 \\ \hline & 1 & 2 \\ +\,1 & 2 & \\ \hline 1 & 3 & 2 \end{array} \qquad\qquad \begin{array}{rrr} 1 & 3 & 2 \\ + & 2 & 1 \\ \hline 2 & 1 & 3 \end{array}$$

For 2 × 3, the product is 6_{10}, which is 12_4. Also, $3 + 2 = 5_{10} = 11_4$. Thus $(39)_{10} = (213)_4$. The answer can be verified as follows:

$$(2\,1\,3)_4 = (2 \times 4^2 + 1 \times 4^1 + 3 \times 4^0)_{10} = (3\,9)_{10}$$

From the above example, it is seen that in converting a decimal number to base R, computations have to be carried out using base R arithmetic. To avoid all the inconveniences and errors arisen from computations using base R arithmetic, two algorithms that employ decimal arithmetic are developed for the conversion. The integer part and the fractional part of N are treated separately.

Division Method for Converting Integers

If a given decimal number $a_{n-1}a_{n-2}$ .... $a_2a_1a_0$ is divided by its radix, 10, the result is

$$a_{n-1}a_{n-2} \text{ .... } a_2a_1a_0 \,. /10 = a_{n-1}a_{n-2} \text{ .... } a_2a_1 \,.\, a_0$$

This result is equivalent to placing a decimal point after the least significant digit and shifting the decimal point one place to the left. When expressed in terms of quotient and remainder, the quotient is $a_{n-1}a_{n-2}$ .... a_2a_1 and the remainder is a_0.

The equivalence between dividing a number by its radix and shifting the radix point of the number to its left one place is applicable to any number systems. For instance

$$(213 / 10)_4 = (21.3)_4$$

In general

$$(b_{m-1}b_{m-2} \text{ .... } b_2b_1b_0 /10)_R = (b_{m-1}b_{m-2} \text{ .... } b_2b_1 \,.\, b_0)_R$$

If a decimal integer is to be converted to radix R such that

$$(a_{n-1}a_{n-2} \text{ .... } a_2a_1a_0)_{10} = (b_{m-1}b_{m-2} \text{ .... } b_2b_1b_0)_R$$

the conversion is to find all the unknown digits b_i in base R from the given decimal number.

If the number in base R is divided by its base,

$$(b_{m-1}b_{m-2} \text{ .... } b_2b_1b_0/10)_R = (b_{m-1}b_{m-2} \text{ .... } b_2b_1 \,.\, b_0)_R$$

$(b_{m-1}b_{m-2}$ .... $b_2b_1)_R$ and $(b_0)_R$ are the quotient and remainder of division. The division, as shown below, can be carried out using the given decimal number

$$(b_{m-1}b_{m-2} \text{ .... } b_2b_1b_0/10)_R = (a_{n-1}a_{n-2} \text{ .... } a_2a_1a_0 /R)_{10}$$

Thus b_0 can be obtained by dividing the decimal number by R. Similarly, the quotient $(b_{m-1}b_{m-2}$ .... $b_2b_1)_R$ can be divided by R to obtain b_1. The division process is repeated until all the digits b_i are found. In other words, the conversion is complete when a quotient of zero is reached.

❖ Example 2.6 (revisit)

The conversion of a decimal number to base 4 in Example 2.6 is revisited using the division method.

	Quotient	Remainder
$39 \div 4 =$	9	3
$9 \div 4 =$	2	1
$2 \div 4 =$	0	2

$$(39)_{10} = (2\ 1\ 3)_4$$

Note that the first remainder is the least significant digit in base R and the last remainder (with a quotient of zero) is the most significant digit.

❖ Example 2.7

Find the binary equivalent of $(153)_{10}$.

	Quotient	Remainder
$153 \div 2 =$	76	1
$76 \div 2 =$	38	0
$38 \div 2 =$	19	0
$19 \div 2 =$	9	1
$9 \div 2 =$	4	1
$4 \div 2 =$	2	0
$2 \div 2 =$	1	0
$1 \div 2 =$	0	1

$$(1\,5\,3)_{10} = (1\ 0\ 0\ 1\ 1\ 0\ 0\ 1)_2$$

❖ Example 2.8

Convert $(1905)_{10}$ to an octal number.

Quotient   Remainder

$1905 \div 8 = 238$ 1

$238 \div 8 = 29$ 6

$29 \div 8 = 3$ 5

$3 \div 8 = 0$ 3

$$(1905)_{10} = (3561)_8$$

❖ Example 2.9

Convert $(47)_{10}$ to a ternary number.

Quotient   Remainder

$47 \div 3 = 15$ 2

$15 \div 3 = 5$ 0

$5 \div 3 = 1$ 2

$1 \div 3 = 0$ 1

$$(47)_{10} = (1202)_3$$

Note that every remainder is a single digit in base R. Since division is performed in base 10, the remainder may not necessarily be a single decimal digit. It can be greater than 9 if R is greater than 10. Under such circumstances, the remainder must be converted to its equivalent in base R.

❖ Example 2.10

This example illustrates the conversion of a decimal integer to a base that is greater than 10. The decimal number 5583 is to be converted to a hexadecimal number.

	Quotient	Remainder: Decimal	Remainder: Hexadecimal
$5583 \div 16 =$	348	15	F
$348 \div 16 =$	21	12	C
$21 \div 16 =$	1	5	5
$1 \div 16 =$	0	1	1

$$(5583)_{10} = (15CF)_{16}$$

<u>Multiplication Method for Converting Fractions</u>

When a decimal number $.a_{-1}a_{-2} \ldots\ldots\ldots\; a_{-m+1}a_{-m}$ is multiplied by its radix, 10, the result is

$$.a_{-1}a_{-2} \ldots\ldots\ldots\; a_{-m+1}a_{-m} \times 10 = a_{-1} . a_{-2} \ldots\ldots\ldots\; a_{-m+1}a_{-m}$$

This result is equivalent to shifting the decimal point one place to the right, which results in an integer part a_{-1} and a fraction $.a_{-2} \ldots\ldots\ldots\; a_{-m+1}a_{-m}$. The equivalence between multiplying a number by its radix and shifting the radix point of the number to its right one place is applicable to other non-decimal number systems. In general,

$$(.b_{q-1}b_{q-2} \ldots\ldots\ldots\; b_{q-m+1}b_{q-m} \times 10)_R = (b_{q-1} . b_{q-2} \ldots\ldots\ldots\; b_{q-m+1}b_{q-m})_R$$

When a decimal fraction is converted to radix R such that

$$(.a_{-1}a_{-2} \ldots\ldots\ldots\; a_{-m+1}a_{-m})_{10} = (.b_{-1}b_{-2} \ldots\ldots\ldots\; b_{-m+1}b_{-m})_R$$

it requires to find all the unknown digits b_i in base R from the given decimal fraction.

When multiplying the number by R, the first digit b_{-1} will appear in the integer part.

$$(.a_{-1}a_{-2} \ldots\ldots\ldots\; a_{-m+1}a_{-m} \times R)_{10} = (b_{-1} . b_{-2} \ldots\ldots\ldots\; b_{-m+1}b_{-m})_R$$

The fraction $.b_{-2} \ldots\ldots\ldots\; b_{-m+1}b_{-m}$ will then be used to find the digit b_{-2} by multiplying the fraction by R. The multiplication process is repeated until all of the digits b_i are found. That is, when the fraction resulting from multiplication is zero.

❖ Example 2.11

A decimal fraction, 0.4375, is to be converted to its octal equivalent.

$$(0.4375)_{10} = (0.34)_8$$

			Fraction	Integer
0.4375×8	= 3.5	=	.5	3
0.5×8	= 4.0	=	.0	4

❖ Example 2.12

The conversion of a decimal fraction to binary in this example illustrates that the integer part from multiplication can be 0, which is a digit and should not be discarded.

$$(0.625)_{10} = (0.101)_2$$

			Fraction	Integer
0.625×2	= 1.25	=	.25	1
0.25×2	= 0.5	=	.5	0
0.5×2	= 1.0	=	.0	1

❖ Example 2.13

This example is to find the equivalent of a decimal fraction in base 5.

$$(0.32)_{10} = (0.13)_5$$

			Fraction	Integer
0.32×5	= 1.6	=	.6	1
0.6×5	= 3.0	=	.0	3

❖ Example 2.14

This example shows that the multiplication process may continue indefinitely and has to be ended at certain point.

$$(0.75)_{10} = (0.33333\ldots\ldots)_5$$

$0.75 \times 5 = 3.75 =$	.75	3	
$0.75 \times 5 = 3.75 =$	.75	3	
	Fraction	Integer	

The result will be $(0.34)_5$ if it is to be rounded to two significant digits.

In the multiplication method, the integer part from multiplication may not necessarily be a single decimal digit if R is greater than 10. Under such circumstances, it must be converted to an equivalent in base R. This is illustrated in the next example.

❖ Example 2.15

This example illustrates the conversion of a decimal number to hexadecimal.

$$(0.90625)_{10} = (0.E8)_{16}$$

$0.90625 \times 16 = 14.5 =$	.5	14 $\rightarrow$	E
$0.5 \times 16 = 8.0 =$	.0	8 $\rightarrow$	8
		Decimal	Hexadecimal
	Fraction	Integer	

2.2.3 Conversion between Binary and Decimal

Although the polynomial substitution method and the division method can be used to perform conversions between binary and decimal integer, better approaches that use only additions and subtractions are introduced in this section.

When using the polynomial substitution method to find the decimal equivalent of a binary number $(a_{n-1}a_{n-2} \text{ .... } a_2a_1a_0)_2$, each binary digit (bit) a_i is multiplied by 2^i which is listed as a weight under each digit as shown below.

	a_{n-1}	a_{n-2}	a_{n-3}	....	a_3	a_2	a_1	a_0
Weight	2^{n-1}	2^{n-2}	2^{n-3}	....	2^3	2^2	2^1	2^0
	2^{n-1}	2^{n-2}	2^{n-3}	....	8	4	2	1

The weight starts from a value of 1 for the least significant bit and is doubled every time the bit position moves one place towards the left. Since a_i is either 0 or 1, multiplying a 0 by its weight produces a product of 0 and multiplying a 1 by its weight is equal to the weight. The decimal equivalent thus is the sum of all the weights of the 1-bits in a binary number.

❖ Example 2.16

This example illustrates the addition method for the conversion of an 8-bit binary number 10011010 to decimal, which is $(154)_{10}$.

Binary number	1	0	0	1	1	0	1	0	
Weight	128	64	32	16	8	4	2	1	
Decimal number	128 +			16 +	8 +		2		= 154

❖ Example 2.17

Convert $(111111111)_2$ to decimal.

The addition method can also be applied to the conversion of a binary number to a decimal number in this example. However,

$$111111111 = 1000000000 - 1$$

A better approach is to subtract 1 from the decimal equivalent of $(1000000000)_2$. The weight of the 1-bit in 1000000000 is $2^9 = 512$. Thus

$$(111111111)_2 = (511)_{10}$$

From this example, it is seen that the range of an n-bit binary number N is

$$0 \le N \le (2^n - 1)$$

As addition is used to convert a binary integer to decimal, subtraction can be employed to find the binary equivalent of a decimal integer. The approach attempts to find the weights of all the 1-bits and subtract them from the largest possible weight until a difference of 0 is reached.

In the subtraction method, the largest weight that is smaller than the given decimal number is first determined and subtracted from the decimal integer. Subtraction continues for the next largest weight that is smaller than the difference from the first subtraction. The process of subtracting the largest weight from the difference of the most recent subtraction repeats until a difference of zero is reached. If a weight of 2^i is subtracted, a_i = 1. Otherwise $a_i = 0$.

❖ Example 2.18

The subtraction method is used to find the binary equivalent of $(1658)_{10}$.

Weight	1658	decimal number N
$2^{10} = 1024$	- 1024	
	634	difference
$2^9 = 512$	- 512	
	122	difference
$2^6 = 64$	- 64	
	58	difference
$2^5 = 32$	- 32	
	26	difference
$2^4 = 16$	- 16	
	10	difference
$2^3 = 8$	- 8	
	2	difference
$2^1 = 2$	- 2	
	0	stop

The weights that are subtracted from the decimal number are $2^{10}, 2^9, 2^6, 2^5, 2^4, 2^3, 2^1$. Therefore $a_{10} = a_9 = a_6 = a_5 = a_4 = a_3 = a_1 = 1$, $a_8 = a_7 = a_2 = a_0 = 0$, and

$$(1658)_{10} = (a_{10}\, a_9\, a_8\, a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0)_2 = (11001111010)_2$$

The addition method and subtraction method can also be extended to fractions. The individual bits of a binary fraction and their respective weights are listed below for a binary fraction

	a_{-1}	a_{-2}	a_{-3}	....	a_{-q+3}	a_{-q+2}	a_{-q+1}	a_{-q}
Weight	2^{-1}	2^{-2}	2^{-3}	....	2^{-q+3}	2^{-q+2}	2^{-q+1}	2^{-q}
	.5	.25	.125	....	2^{-q+3}	2^{-q+2}	2^{-q+1}	2^{-q}

Because the weights are fractions, the addition method may not be much more effective than the polynomial substitution method when converting a binary fraction to decimal. The subtraction method for converting a decimal fraction to binary also does not have advantage over the multiplication method.

2.2.4 Conversion without Computations

When two bases R_1 and R_2 satisfy the relationship $R_1 = (R_2)^n$ in which n is an integer and $n \geq 2$, no computations are required in the conversions between those systems. A digit in base R_1 is equivalent to n digits in R_2. The following procedure can be used in number conversions.

(i) Construct a table that lists all the digits in R_1 and their equivalents in R_2.
(ii) If the conversion is from R_1 to R_2, replace each digit in R_1 with n digits in R_2 by looking up the table. Leave the radix point intact.
(iii) If the conversion is from R_2 to R_1, arrange the digits of the number in R_2 in groups of n digits from the radix point and towards both directions. If there are not enough digits in the leftmost group of the integer part, add leading zeros to the integer part so that the leftmost group consists of n digits in R_2. If the rightmost group in the fractional part of the number does not have n digits, add zeros to the right of the last significant digit in the fractional part so that there are n digits in the rightmost group of the fraction. Replace each group of n digits in base R_2 with a single digit in R_1 by looking up the table. Leave the radix point intact.

The equivalents between three bits and one octal digit, and between four bits and one hexadecimal digit, are listed in Table 2.2.

❖ Example 2.19

This example shows how a number can be converted without computations. It also illustrates the conversion between two systems which do not satisfy the relationship $R_1 = R_2^n$ such as octal and hexadecimal, but can be converted via binary.

(a)
$$
\begin{aligned}
& (F8.A7)_{16} \\
= & (1111\ \ 1000.1010\ \ 0111)_2 \\
= & (011\ \ 111\ \ 000.101\ \ 001\ \ 110)_2 \\
= & (370.516)_8
\end{aligned}
$$

(b) $(20000)_{16}$

$= (0010\ 0000\ 0000\ 0000\ 0000)_2$

$= (100\ 000\ 000\ 000\ 000\ 000)_2$

$= (400000)_8$

(c) $(320.71)_8$

$= (011\ 010\ 000.111\ 001)_2$

$= (1101\ 0000.1110\ 0100)_2$

$= (D0.E4)_{16}$

Table 2.2 (a) Conversion between binary and octal. (b) Conversion between binary and hexadecimal.

(a)

R =2	R = 8
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

(b)

R =2	R = 16
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

2.3 Binary Codes

Conversion between decimal numbers and binary numbers is a means for communication between human and computers. The conversion process is analogous to the translation between two different languages. However, a decimal digit sometimes can be represented by a sequence of bits assigned to it based on certain specific rules. The assigned bit sequence is called a binary code or computer code. The bit sequence for the binary code of a decimal digit is not necessarily the same as the binary equivalent of a decimal digit. To represent the ten decimal digits by binary codes, it requires a minimum of four bits. A sequence of three bits has only eight different combinations that are not sufficient to represent the ten decimal digits. With a sequence of four bits, ten of the sixteen combinations can be used to represent the ten decimal digits whereas the other six combinations should never be used and are considered as invalid codes.

The BCD (binary-coded-decimal) code given in Table 2.3 selects the first ten combinations of a 4-bit sequence to represent the ten decimal digits. It is also called a weighted-code. Each of the four bits in a BCD code is given a weight. The weights are 8, 4, 2, and 1 from left to right. The equivalent decimal digit of a BCD code is the sum of the weights of all the 1-bits. A (6, 3, 1,1) weighted-code is also given in Table 2.3. Some of the (6, 3, 1,1) codes are not unique. For example, decimal digit 4 can be encoded as 0101 and 0110. When there is more than one combination, only one is selected and all the others are considered as invalid. Two other codes are also listed in Table 2.4. The excess-3 code discards the first three and the last three of the sixteen combinations. For the reflected (gray) code, only one of the four bits will change from 0 to 1 or from 1 to 0 when the decimal digit is changed by 1. This is also true when the change is from 9 to 0. The gray code is useful when it is used to control mechanical parts and in analog-digital conversion.

Table 2.3 Weighted codes for decimal digits.

Decimal digit	BCD code (8, 4, 2, 1) weighted code	(6, 3, 1, 1) weighted code	(6, 3, 1, 1) weighted code
0	0 0 0 0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1	0 0 1 0
2	0 0 1 0	0 0 1 1	0 0 1 1
3	0 0 1 1	0 1 0 0	0 1 0 0
4	0 1 0 0	0 1 0 1	0 1 1 0
5	0 1 0 1	0 1 1 1	0 1 1 1
6	0 1 1 0	1 0 0 0	1 0 0 0
7	0 1 1 1	1 0 0 1	1 0 1 0
8	1 0 0 0	1 0 1 1	1 0 1 1
9	1 0 0 1	1 1 0 0	1 1 0 0
Unused (invalid) codes	1 0 1 0	0 0 1 0	0 0 0 1
	1 0 1 1	0 1 1 0	0 1 0 1
	1 1 0 0	1 0 1 0	1 0 0 1
	1 1 0 1	1 1 0 1	1 1 0 1
	1 1 1 0	1 1 1 0	1 1 1 0
	1 1 1 1	1 1 1 1	1 1 1 1

❖ Example 2.20

This example shows the conversions between numbers and binary codes.

(a) $(\,2\,5\,7\,0\,1\,)_{10}$

$= (\,0010\ \ 0101\ \ 0111\ \ 0000\ \ 0001\,)_{BCD}$

$= (\,0101\ \ 1000\ \ 1010\ \ 0011\ \ 0100\,)_{\text{Excess-3}}$

(b) $(\,1\,0\,0\,0\,0\,0\,0\,0\,0\,)_{2}$

$= (\,2\,5\,6\,)_{10}$

$= (\,0010\ \ 0101\ \ 0110\,)_{BCD}$

$= (\,0101\ \ 1000\ \ 1001\,)_{\text{Excess-3}}$

(c) $(\,0011\ \ 0100\ \ 0110\ \ 1001\ \ 0001)_{BCD}$

$= (\,3\,4\,6\,9\,1\,)_{10}$

(d) $(\,0011\ \ 0100\ \ 1110\ \ 1001\ \ 0001)_{BCD}$

$= (\,?\,)_{10}$

The binary codes are used for decimal numbers. A non-decimal number, such as the one in Example 2.20(b), should be first converted to decimal before being converted to binary codes. Although the leading zeros of the integer part and the trailing zeros of the fractional part of a binary number can be discarded, under no circumstance should any zero be removed from the binary codes. Every digit in a decimal number is expressed by exactly by four bits. The BCD code in (d) is not valid because the third BCD code, 1110, is not valid. Its decimal equivalent cannot be determined.

Table 2.4 Other binary codes.

Decimal digit	Excess-3 code	Reflected (Gray) code	2-out-of-5 code
0	0 0 1 1	0 0 0 0	0 0 0 1 1
1	0 1 0 0	0 0 0 1	0 0 1 0 1
2	0 1 0 1	0 0 1 1	0 0 1 1 0
3	0 1 1 0	0 0 1 0	0 1 0 0 1
4	0 1 1 1	0 1 1 0	0 1 0 1 0
5	1 0 0 0	1 1 1 0	0 1 1 0 0
6	1 0 0 1	1 0 1 0	1 0 0 0 1
7	1 0 1 0	1 0 1 1	1 0 0 1 0
8	1 0 1 1	1 0 0 1	1 0 1 0 0
9	1 1 0 0	1 0 0 0	1 1 0 0 0
Unused (invalid) codes	0 0 0 0	0 1 1 1	0 0 0 0 0
	0 0 0 1	0 1 0 1	0 0 0 0 1
	0 0 1 0	0 1 0 0	……….
	1 1 0 1	1 1 0 0	……….
	1 1 1 0	1 1 0 1	1 1 1 1 0
	1 1 1 1	1 1 1 1	1 1 1 1 1

2.4 Error Detection

A sequence with more than four bits can also be used to represent the ten decimal digits. Extra bits beyond four are not redundant but serve the purpose of detecting or correcting bit errors. The 2-out-of-5 code in Table 2.4 can be used to detect a single bit error. Since the total number of 1-bits in any 2-out-of-5 code is two, there is only one 1-bit if one of the two 1-bits is wrong and has changed from 1 to 0. On the other hand, if the error bit was originally a 0-bit, then there are three 1-bits instead of two 1-bits.

As an illustration, consider the 2-out-of-5 code word N = $c_4c_3c_2c_1c_0$ =10010. c_1 is contaminated during transmission and changes to 0. $c_4c_3c_2c_1c_0$ then becomes 10000. The occurrence of only one 1-bit indicates that N = 10000 is erroneous. The change of anyone of the four 0-bits to a 1-bit will revert to a 2-out-of-5 code. However, there is no way to tell which one of the four 0-bits is the error bit. The recovery of the original code word N is not possible. Thus the 2-out-of-5 code can be used only to detect a single error. Correction is not possible. Neither can the 2-out-of-5 code be used to detect double errors. Suppose there are errors in c_1 and c_0, N will then become 10001, which is still a 2-out-of-5 code but definitely not the correct code for N.

Table 2.5 Single-error detection codes.

				Even Parity					Odd Parity				
a_3	a_2	a_1	a_0	a_3	a_2	a_1	a_0	P	a_3	a_2	a_1	a_0	P
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	1	0	0	0	1	0
0	0	1	0	0	0	1	0	1	0	0	1	0	0
0	0	1	1	0	0	1	1	0	0	0	1	1	1
0	1	0	0	0	1	0	0	1	0	1	0	0	0
0	1	0	1	0	1	0	1	0	0	1	0	1	1
0	1	1	0	0	1	1	0	0	0	1	1	0	1
0	1	1	1	0	1	1	1	1	0	1	1	1	0
1	0	0	0	1	0	0	0	1	1	0	0	0	0
1	0	0	1	1	0	0	1	0	1	0	0	1	1
1	0	1	0	1	0	1	0	0	1	0	1	0	1
1	0	1	1	1	0	1	1	1	1	0	1	1	0
1	1	0	0	1	1	0	0	0	1	1	0	0	1
1	1	0	1	1	1	0	1	1	1	1	0	1	0
1	1	1	0	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	1	1	0	1	1	1	1	1

The even-parity and odd-parity codes in Table 2.5 are also for single-error detection. They are more general than the 2-out-of-5 code and can be applied to a sequence with any number of bits. The generation of a single error detection code is by adding a parity bit that is either 0 or 1 to the original code such that the total number of 1-bits in the sequence including the parity bit is even for even parity and odd for odd parity. In other words, given a sequence Q = s_{n-1} s_{n-2} .... s_1 s_0, P, the even parity bit for Q, is 0 if (s_{n-1} + s_{n-2} + ... + s_1 + s_0) modulo 2 = 0. P = 1 if (s_{n-1} + s_{n-2} + ... + s_1 + s_0) modulo 2 = 1. The value of P should change from 0 to 1 or 1 to 0 for odd parity. (s_{n-1} + s_{n-2} + ... + s_1 + s_0) modulo 2 can be obtained easily by counting the number of 1-bits in Q. It equals 0 or 1 if the total number of 1-bits is, respectively, even or odd. For example, if Q = 01011110, then 0 + 1 + 0 + 1 + 1 + 1 + 1 + 0 = 5. 5 mod 2 = 1. Thus P = 1 for even parity and P = 0 for odd parity. The complete sequence is 010111101 for even parity or 010111100 for odd parity. Because the total number of 1-bits in a single error detection code is even (odd) for even (odd) parity, the total number of 1-bits will become odd (even) when a single error occurs. The addition of just one parity bit cannot correct a single error or detect more than one error.

PROBLEMS

1. Convert each of the following numbers to decimal by the polynomial substitution method.
   (a) 10010111.111_2 (b) 11101110.001_2
   (c) 125.3_6 (d) $AB2.9_{12}$
   (e) 216.35_8 (f) $10B2.4_{16}$

2. Convert each of the following decimal integers to base R using the division method.
   (a) 1000, R = 9 (b) 202, R = 16
   (c) 278, R = 5 (d) 3467, R = 7

3. Convert each of the following decimal fractions to base R using the multiplication method.
   (a) .46875, R = 8 (b) .46875, R = 16
   (c) .64, R = 5 (d) .1875, R = 2

4. Convert each of the following numbers to base 6.
   (a) 1000_{16} (b) 200_3
   (c) 214_5 (d) 3467_8

5. If 353 is the decimal equivalent of 541 in base R, find R using algebra. (Do not use the trial-and-error method.)

6. Convert each of the following decimal numbers to binary, octal, and hexadecimal.
   (a) 100 (b) 217
   (c) 472.625 (d) 256.03125

7. Convert each of the following decimal number to binary.
   (a) 2047 (b) 2^{16}
   (c) 2^{12} - 1 (d) 111111

8. Convert each of the following binary number to decimal using the addition method.
   (a) 11010111 (b) 10010110
   (c) 1011000010 (d) 1110110011

9. Convert each of the following decimal number to binary using the subtraction method.
   (a) 1567 (b) 3489
   (c) 8204 (d) 9999

10. Perform the following conversions from base 4 to base R using the most efficient method.
    (a) 2132.12_4, R = 16
    (b) 212.32_4, R = 2
    (c) 132.02_4, R = 10
    (d) 123_4, R = 7

11. Convert the following numbers to base 4 using the most efficient method.
    (a) 101101_2
    (b) 101101_{16}
    (c) 101101_8
    (d) 745.125_{10}

12. Perform the following conversions to or from BCD code.
    (a) 2508_{10} to BCD code
    (b) $(11111)_2$ to BCD code
    (c) $(1001001101010111)_{BCD}$ to decimal
    (d) $(000100010001)_{BCD}$ to binary

13. Perform the following number/code conversions:
    (a) 1078_{10} to excess-3 code
    (b) $(010110011011)_{excess\text{-}3}$ to BCD code
    (c) $(10000000)_2$ to excess-3 code
    (d) $12CF_{16}$ to BCD code

14. (a) Construct a (7, 3, 1, -2) weighted decimal code $a_3a_2a_1a_0$.
    (b) Encode the weighted code in (a) as a 5-bit single-error-detection code $a_3a_2a_1a_0P$, where P is an even parity check bit.

15. (a) Explain why it is impossible to construct a (6, 4, 3, 2) weighted code.
    (b) Is a weighted code possible if one of the four weights in (a) is changed to negative? If it is possible, find all the possible weighted codes with three positive weights and one negative weight.