Part 1 Preliminaries

CHAPTER 1 Number systems and codes

This chapter deals with the representation of numerical data, with emphasis on those representations that use only two symbols, 0 and 1. Described are special methods of representing numerical data that afford protection against various transmission errors and component failures.

1.1 Number systems

Convenient as the decimal number system generally is, its usefulness in machine computation is limited because of the nature of practical electronic devices. In most present digital machines, the numbers are represented, and the arithmetic operations performed, in a different number system called the binary number system. This section is concerned with the representation of numbers in various systems and with methods of conversion from one system to another.

Number representation

An ordinary decimal number actually represents a polynomial in powers of 10. For example, the number 123.45 represents the polynomial

$$123.45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}.$$

This method of representing decimal numbers is known as the *decimal number system*, and the number 10 is referred to as the *base* (or *radix*) of the system. In a system whose base is b, a positive number N represents the polynomial

$$\begin{aligned} N &= a_{q-1}b^{q-1} + \cdots + a_0 b^0 + \cdots + a_{-p}b^{-p} \\ &= \sum_{i=-p}^{q-1} a_i b^i, \end{aligned}$$

where the base b is an integer greater than 1 and the a's are integers in the range $0 \leq a_i \leq b-1$. The sequence of digits $a_{q-1}a_{q-2}\cdots a_0$ constitutes the *integer*

Table 1.1 Representation of integers

Base				
2	4	8	10	12
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	10	4	4	4
0101	11	5	5	5
0110	12	6	6	6
0111	13	7	7	7
1000	20	10	8	8
1001	21	11	9	9
1010	22	12	10	α
1011	23	13	11	β
1100	30	14	12	10
1101	31	15	13	11
1110	32	16	14	12
1111	33	17	15	13

part of N, while the sequence $a_{-1}a_{-2}\cdots a_{-p}$ constitutes the *fractional part* of N. Thus, p and q designate the number of digits in the fractional and integer parts, respectively. The integer and fractional parts are usually separated by a *radix point*. The digit a_{-p} is referred to as the *least significant digit* while a_{q-1} is called the *most significant digit*.

When the base b equals 2, the number representation is referred to as the *binary number system*. For example, the binary number 1101.01 represents the polynomial

$$1101.01 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2},$$

that is,

$$1101.01 = \sum_{i=-2}^{3} a_i 2^i,$$

where $a_{-2} = a_0 = a_2 = a_3 = 1$ and $a_{-1} = a_1 = 0$.

A number N in base b is usually denoted $(N)_b$. Whenever the base is not specified, base 10 is implicit. Table 1.1 shows the representations of integers 0 through 15 in several number systems.

The *complement* of a digit a, denoted a', in base b is defined as

$$a' = (b - 1) - a.$$

That is, the complement a' is the difference between the largest digit in base b and digit a. In the binary number system, since $b = 2$, $0' = 1$ and $1' = 0$.

In the decimal number system, the largest digit is 9. Thus, for example, the complement[1] of 3 is $9 - 3 = 6$.

Conversion of bases

Suppose that some number N, which we wish to express in base b_2, is presently expressed in base b_1. In converting a number from base b_1 to base b_2, it is convenient to distinguish between two cases. In the first case $b_1 < b_2$, and consequently base-b_2 arithmetic can be used in the conversion process. The conversion technique involves expressing number $(N)_{b_1}$ as a polynomial in powers of b_1 and evaluating the polynomial using base-b_2 arithmetic.

Example We wish to express the numbers $(432.2)_8$ and $(1101.01)_2$ in base 10. Thus

$$\begin{gathered}(432.2)_8 = 4 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 + 2 \times 8^{-1} = (282.25)_{10}, \\ (1101.01)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \\ \times 2^{-1} + 1 \times 2^{-2} = (13.25)_{10}.\end{gathered}$$

In both cases, the arithmetic operations are done in base 10.

When $b_1 > b_2$ it is more convenient to use base-b_1 arithmetic. The conversion procedure will be obtained by considering separately the integer and fractional parts of N. Let $(N)_{b_1}$ be an integer whose value in *base* b_2 is given by

$$(N)_{b_1} = a_{q-1}b_2^{q-1} + a_{q-2}b_2^{q-2} + \cdots + a_1 b_2^1 + a_0 b_2^0.$$

To find the values of the a's, let us divide the above polynomial by b_2.

$$\frac{(N)_{b_1}}{b_2} = \underbrace{a_{q-1}b_2^{q-2} + a_{q-2}b_2^{q-3} + \cdots + a_1}_{Q_0} + \frac{a_0}{b_2}.$$

Thus, the least significant digit of $(N)_{b_2}$, i.e., a_0, is equal to the first remainder. The next most significant digit, a_1, is obtained by dividing the quotient Q_0 by b_2, i.e.,

$$\left(\frac{Q_0}{b_2}\right)_{b_1} = \underbrace{a_{q-1}b_2^{q-3} + a_{q-2}b_2^{q-4} + \cdots}_{Q_1} + \frac{a_1}{b_2}.$$

The remaining a's are evaluated by repeated divisions of the quotients until Q_{q-1} is equal to zero. If N is finite, the process must terminate.

[1] In the decimal system, the complement is also referred to as the 9's complement. In the binary system, it is also known as the 1's complement.

Example The above conversion procedure is now applied to convert $(548)_{10}$ to base 8. The r_i in the table below denote the remainders. The first entries in the table are 68 and 4, corresponding, respectively, to the quotient Q_0 and the first remainder from the division $(548/8)_{10}$. The remaining entries are found by successive division.

Q_i	r_i
68	$4 = a_0$
8	$4 = a_1$
1	$0 = a_2$
	$1 = a_3$

Thus, $(548)_{10} = (1044)_8$. In a similar manner we can obtain the conversion of $(345)_{10}$ to $(1333)_6$, as illustrated in the table below.

Q_i	r_i
57	$3 = a_0$
9	$3 = a_1$
1	$3 = a_2$
	$1 = a_3$

Indeed, $(1333)_6$ can be reconverted to base 10, i.e.,

$$(1333)_6 = 1 \times 6^3 + 3 \times 6^2 + 3 \times 6^1 + 3 \times 6^0 = 345$$

If $(N)_{b_1}$ is a fraction, a dual procedure is employed. It can be expressed in base b_2 as follows:

$$(N)_{b_1} = a_{-1}b_2^{-1} + a_{-2}b_2^{-2} + \cdots + a_{-p}b_2^{-p}.$$

The most significant digit, a_{-1}, can be obtained by multiplying the polynomial by b_2:

$$b_2 \cdot (N)_{b_1} = a_{-1} + a_{-2}b_2^{-1} + \cdots + a_{-p}b_2^{-p+1}.$$

If the above product is less than 1 then a_{-1} equals 0; if the product is greater than or equal to 1 then a_{-1} is equal to the integer part of the product. The next most significant digit, a_{-2}, is found by multiplying the fractional part of the above product part by b_2 and determining its integer part; and so on. This process does not necessarily terminate since it may not be possible to represent the fraction in base b_2 with a finite number of digits.

Example To convert $(0.3125)_{10}$ to base 8, find the digits as follows:

$$\begin{aligned} 0.3125 \times 8 &= 2.5000, &\text{hence}\quad a_{-1} &= 2;\\ 0.5000 \times 8 &= 4.0000, &\text{hence}\quad a_{-2} &= 4. \end{aligned}$$

Thus $(0.3125)_{10} = (0.24)_8$.
Similarly, the computation below proves that $(0.375)_{10} = (0.011)_2$:

$$\begin{aligned} 0.375 \times 2 &= 0.750, &\text{hence}\quad a_{-1} &= 0;\\ 0.750 \times 2 &= 1.500, &\text{hence}\quad a_{-2} &= 1;\\ 0.500 \times 2 &= 1.000, &\text{hence}\quad a_{-3} &= 1. \end{aligned}$$

Example To convert $(432.354)_{10}$ to binary, we first convert the integer part and then the fractional part. For the integer part we have

Q_i	r_i
216	$0 = a_0$
108	$0 = a_1$
54	$0 = a_2$
27	$0 = a_3$
13	$1 = a_4$
6	$1 = a_5$
3	$0 = a_6$
1	$1 = a_7$
	$1 = a_8$

Hence $(432)_{10} = (110110000)_2$. For the fractional part we have

$$\begin{aligned} 0.354 \times 2 &= 0.708, &\text{hence}\quad a_{-1} &= 0,\\ 0.708 \times 2 &= 1.416, &\text{hence}\quad a_{-2} &= 1,\\ 0.416 \times 2 &= 0.832, &\text{hence}\quad a_{-3} &= 0,\\ 0.832 \times 2 &= 1.664, &\text{hence}\quad a_{-4} &= 1,\\ 0.664 \times 2 &= 1.328, &\text{hence}\quad a_{-5} &= 1,\\ 0.328 \times 2 &= 0.656, &\text{hence}\quad a_{-6} &= 0,\\ && a_{-7} &= 1,\\ && &\text{etc.} \end{aligned}$$

Consequently $(0.354)_{10} = (0.0101101\cdots)_2$. The conversion is usually carried up to the desired accuracy. In our example, reconversion to base 10 shows that

$$(110110000.0101101)_2 = (432.3515)_{10}$$

Table 1.2 Elementary binary operations

Bits		Sum		Difference		Product
a	b	$a+b$	Carry	$a-b$	Borrow	$a \cdot b$
0	0	0	0	0	0	0
0	1	1	0	1	1	0
1	0	1	0	1	0	0
1	1	0	1	0	0	1

A considerably simpler conversion procedure may be employed in converting octal numbers (i.e., numbers in base 8) to binary and vice versa. Since $8 = 2^3$, each octal digit can be expressed by three binary digits. For example, $(6)_8$ can be expressed as $(110)_2$, etc. The procedure of converting a binary number into an octal number consists of partitioning the binary number into groups of three digits, starting from the binary point, and to determine the octal digit corresponding to each group.

Example

$$(123.4)_8 = (001\ 010\ 011.100)_2,$$
$$(1010110.0101)_2 = (001\ 010\ 110.010\ 100) = (126.24)_8.$$

A similar procedure may be employed in conversions from binary to hexadecimal (base 16), except that four binary digits are needed to represent a single hexadecimal digit. In fact, whenever a number is converted from base b_1 to base b_2, where $b_2 = b_1^k$, k digits of that number when grouped may be represented by a single digit from base b_2.

Binary arithmetic

The binary number system is widely used in digital systems. Although a detailed study of digital arithmetic is beyond the scope of this book, we shall present the elementary techniques of binary arithmetic. The basic arithmetic operations are summarized in Table 1.2, where the sum and carry, difference and borrow, and product are computed for every combination of binary digits (abbreviated *bits*) 0 and 1. For a more comprehensive discussion of computer arithmetic, the reader may consult [2].

Binary addition is performed in a manner similar to that of decimal addition. Corresponding bits are added and if a carry 1 is produced then it is added to the binary digits at the left.

Example The addition of $(15.25)_{10}$ and $(7.50)_{10}$ in binary proceeds as follows:

$$\begin{array}{rl}
1111 & \text{carries of } 1 \\
1111.01 & = (15.25)_{10} \\
+ & \\
0111.10 & = (\ 7.50)_{10} \\
\hline
10110.11 & = (22.75)_{10}
\end{array}$$

In subtraction, if a borrow of 1 occurs and the next left digit of the minuend (the number from which a subtraction is being made) is 1 then the latter is changed to 0 and subtraction is continued in the usual manner. If, however, the next left digit of the minuend is 0 then it is changed to 1, as is each successive minuend digit to the left which is equal to 0. The first minuend digit to the left, which is equal to 1, is changed to 0, and subtraction is continued.

Example The subtraction of $(12.50)_{10}$ from $(18.75)_{10}$ in binary proceeds as follows:

$$\begin{array}{rl}
1 & \text{borrows of } 1 \\
10010.11 & = (18.75)_{10} \\
- & \\
01100.10 & = (12.50)_{10} \\
\hline
00110.01 & = (\ 6.25)_{10}
\end{array}$$

Just as with decimal numbers, the multiplication of binary numbers is performed by successive addition while division is performed by successive subtraction.

Example Multiply the binary numbers below:

$$\begin{array}{rl}
11001.1 & = (25.5)_{10} \\
\times & \\
110.1 & = (\ 6.5)_{10} \\
\hline
110011 & \\
000000 & \\
110011 & \\
110011 & \\
\hline
10100101.11 & = (165.75)_{10}
\end{array}$$

Example Divide the binary number 1000100110 by 11001.

```
                10110  quotient
      11001|1000100110
            11001
            00100101
                11001
              0011001
                11001
                00000  remainder
```

1.2 Binary codes

Although the binary number system has many practical advantages and is widely used in digital computers, in many cases it is convenient to work with the decimal number system, especially when the communication between human being and machine is extensive, since most numerical data generated by humans is in terms of decimal numbers. To simplify the problem of communication between human and machine, several codes have been devised in which decimal digits are represented by sequences of binary digits.

Weighted codes

In order to represent the 10 decimal digits 0, 1, . . . , 9, it is necessary to use at least four binary digits. Since there are 16 combinations of four binary digits, of which 10 combinations are used, it is possible to form a very large number of distinct codes. Of particular importance is the class of *weighted codes*, whose main characteristic is that each binary digit is assigned a decimal "weight," and, for each group of four bits, the sum of the weights of those binary digits whose value is 1 is equal to the decimal digit which they represent. If w_1, w_2, w_3, and w_4 are the given weights of the binary digits and x_1, x_2, x_3, x_4 the corresponding digit values then the decimal digit $N = w_4x_4 + w_3x_3 + w_2x_2 + w_1x_1$ can be represented by the binary sequence $x_4x_3x_2x_1$. The sequence of binary digits that represents a decimal digit is called a *code word*. Thus, the sequence $x_4x_3x_2x_1$ is the code word for N. Three weighted four-digit binary codes are shown in Table 1.3.

The binary digits in the first code in Table 1.3 are assigned weights 8, 4, 2, 1. As a result of this weight assignment, the code word that corresponds to each decimal digit is the binary equivalent of that digit; e.g., 5 is represented by 0101, and so on. This code is known as the *binary-coded-decimal* (BCD)

Table 1.3 The code words $x_4x_3x_2x_1$ for the decimal digits N in three weighted binary codes

Decimal	$w_4w_3w_2w_1$											
digit N	8	4	2	1	2	4	2	1	6	4	2	−3
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	1	0	1
2	0	0	1	0	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1	1	0	0	1
4	0	1	0	0	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	1	1	1	0	1	1
6	0	1	1	0	1	1	0	0	0	1	1	0
7	0	1	1	1	1	1	0	1	1	1	0	1
8	1	0	0	0	1	1	1	0	1	0	1	0
9	1	0	0	1	1	1	1	1	1	1	1	1

code. For each code in Table 1.3, the decimal digit that corresponds to a given code word is equal to the sum of the weights in those binary positions that are 1's rather than 0's. Thus, in the second code, where the weights are 2, 4, 2, 1, decimal 5 is represented by 1011, corresponding to the sum $2 \times 1 + 4 \times 0 + 2 \times 1 + 1 \times 1 = 5$. The weights assigned to the binary digits may also be negative, as in the code (6, 4, 2, −3). In this code, decimal 5 is represented by 1011, since $6 \times 1 + 4 \times 0 + 2 \times 1 - 3 \times 1 = 5$.

It is apparent that the representations of some decimal numbers in the (2, 4, 2, 1) and (6, 4, 2, −3) codes are not unique. For example, in the (2, 4, 2, 1) code, decimal 7 may be represented by 1101 as well as 0111. Adopting the representations shown in Table 1.3 causes the codes to become self-complementing. A code is said to be *self-complementing* if the code word of the "9's complement of N", i.e., $9 - N$, can be obtained from the code word of N by interchanging all the 1's and 0's. For example, in the (6, 4, 2, −3) code, decimal 3 is represented by 1001 while decimal 6 is represented by 0110. In the (2, 4, 2, 1) code, decimal 2 is represented by 0010 while decimal 7 is represented by 1101. Note that the BCD code (8, 4, 2, 1) is not self-complementing. It can be shown that a necessary condition for a weighted code to be self-complementing is that the sum of the weights must equal 9. There exist only four positively weighted self-complementing codes, namely, (2, 4, 2, 1), (3, 3, 2, 1), (4, 3, 1, 1), and (5, 2, 1, 1). In addition, there exist 13 self-complementing codes with positive and negative weights.

Nonweighted codes

There are many nonweighted binary codes, two of which are shown in Table 1.4. The Excess-3 code is formed by adding 0011 to each BCD code word.