第5章

信息的表示

本章先介绍信息和数据的概念；然后介绍信息在计算机中的表示、编码、存储与运算的过程，具体内容包括十进制、二进制、八进制和十六进制四种数制及其转换，位、字节和字长的概念，整数的真值、机器数和在计算机中的补码表示，实数在计算机中的浮点表示，英文字符和汉字等非数值信息在计算机中的编码，多媒体中的声音、图像、视频等复杂信息在计算机中的数字化过程以及编码；最后介绍二进制数的算术运算和布尔逻辑运算，数字逻辑电路中基本的逻辑门电路的知识，以及门电路作为存储器和运算器的应用。通过本章的学习，读者对计算机中的信息处理过程将有一个更清晰的认识，从而在使用计算机求解现实问题的过程中能更好地把握信息的实质和更合理地处理信息。

5.1 数与进制

本节介绍信息和数据的概念，数的十进制、二进制、八进制、十六进制四种常见的进制表示，数制转换以及各种进制数的算术运算。

5.1.1 信息与数据

信息的广义解释是对现实世界中事物的存在方式或运动状态的反映。从信息论的角度理解信息，则是对事物运动状态或存在方式的不确定性的描述。在通信行业，信息又定义为用于消除通信对方知识上的不确定性的东西。

在计算机科学中，信息通常被认为是能够用计算机处理的任何有意义的内容或消息，它们一般以数据的形式表达，如数字、字符、文本、图像、视频、声音等，而且计算机可以完成信息的表示、存储、传输和处理等过程。

与信息对应的另一个概念则是数据。数据是描述现实世界事物的符号，即用物理符号记录下来的可以鉴别的信息。其中，物理符号可以是数字、文字、图形、图像、声音及其他特殊符号等。数据作为信息的载体，其多种表现形式可以经过数字化后存入计算机中。

信息与数据是两个既有联系又有区别的概念。数据是信息的符号表示，或称载体；信息是数据的内涵，是数据的语义解释；数据是信息存在的一种形式，只有通过解释和处理才能成为有用的信息；数据可用不同的形式表示，而信息不会随数据的形式而改变。

对于生活中的信息来说，同一信息可以有不同的表示方法，表达信息的形式还可以相

互转换。而在计算机科学中，信息的表示形式可以是一种符号表示，即数学上的数据；信息可以转换为物理器件的状态表示，例如，电平的高低、磁极的方向、电路的开闭等。由于电信号具有连续形式，但计算机又受到实现条件的制约，只能表示离散形式的信息，这就给现代计算机采用二进制提出了客观需求。

计算机采用二进制的第一个原因是二进位记数制仅有两个数码 0 和 1，所以，任何具有两个不同稳定状态的元件都可用来表示数的一位。现实中这种具有两种截然不同的稳定状态的元件非常多，例如，氖灯的“亮”和“熄”、阀门的“开”和“关”、电压的“高”和“低”以及“正”和“负”；纸带上的“有孔”和“无孔”、电路中的“有信号”和“无信号”以及磁性材料的“南极”和“北极”等。利用两种截然不同的稳定状态来表示数据不但很容易实现，简化设计的复杂性，而且两种截然不同的稳定状态既是量的不同，更是质的差别，对提高机器的抗干扰能力和可靠性很有好处，况且现实中具有两种以上状态的简单而可靠的器件也不易寻找。

计算机采用二进制的第二个原因是二进位记数制的四则运算规则十分简单，而且四则运算最后都可以归结为加法和移位运算，这使得计算机中的运算器电路设计起来变得非常容易。简化的表示也带来运算速度的大幅度提高，而人们平常使用的十进位记数制是无法胜任计算机的高速运算的。

综上所述，现代计算机一般采用二进制的 0、1 来表示物理上的两种状态，用不同的 0、1 序列表示不同的信息，每一个二进制位称为一个比特(bit)，也称为“位”，简写为 b，是信息的最小单位。“位”常用来描述速率(即单位时间内传输的二进制位数)，如常说的网速是 100M，指的是每秒钟传输 100M 位。而根据计算机物理器件的构造特性，在计算机中，信息的基本单位约定为 8b，叫作 1 字节(Byte，B)。“字节”常用来表示存储空间的大小，如常说的内存 8G 指的是 8GB。

除位(b)、字节(B)外，随着数据量的增大，常用的数据单位还有 KB、MB、GB、TB 等。换算关系如下。

1B(Byte，字节)＝8b(bit，比特)

1KB(Kilobyte，千字节)＝1024B＝2^{10} B

1MB(Megabyte，兆字节，百万字节，简称“兆”)＝1024KB＝2^{20} B

1GB(Gigabyte，吉字节，十亿字节，又称“千兆”)＝1024MB＝2^{30} B

1TB(Terabyte，万亿字节，太字节)＝1024GB＝2^{40} B

1PB(Petabyte，千万亿字节，拍字节)＝1024TB＝2^{50} B

1EB(Exabyte，百亿亿字节，艾字节)＝1024PB＝2^{60} B

1ZB(Zettabyte，十万亿亿字节，泽字节)＝1024EB＝2^{70} B

1YB(Yottabyte，一亿亿亿字节，尧字节)＝1024ZB＝2^{80} B

1BB(Brontobyte，一千亿亿亿字节)＝1024YB＝2^{90} B

1NB(NonaByte，一百万亿亿亿字节)＝1024BB＝2^{100} B

1DB(DoggaByte，十亿亿亿亿字节)＝1024NB＝2^{110} B

【例 5-1】 计算一块标称 512GB 的硬盘的实际容量。

【解】 市场上买到的标称 512GB 的硬盘，在计算机上查看其实际容量，发现差很多。

实际上,商家是按 1000 来进行单位换算的,所以,标称 512GB 的硬盘实际容量为

$$512\times1000\times1000\times1000=512\ 000\ 000\ 000(\mathrm{B})$$

按计算机的标准换算,它的容量是

$$512\ 000\ 000\ 000/1024/1024/1024=476.837(\mathrm{GB})$$

所以在计算机上看到的会是 476GB。

【例 5-2】 计算百兆宽带的下载速度,每秒下载多少字节的数据?

【解】 下载速度 100M 指的是二进制位,而这里的兆(M)是按 10^6 换算得到的。所以,

$$100\mathrm{Mb}=100\ 000\ 000\mathrm{b}$$

转换为字节:

$$100\ 000\ 000/8/1024/1024=11.92(\mathrm{MB})$$

这里的 M 是 2^{20},B 是字节。

5.1.2 进位记数制

表示数据时,用多个数位表示,每个数位一个数字,左边的数位高,右边的数位低,当低位的数字达到某一数字时再加 1,则本位计 0,高位加 1,这种记数的方法就是**进位记数制**。目前日常用的十进制就是一种进位记数制。

用进位记数制表示数据,有以下四个要素。

- 数码:一组用来表示数的数字,如十进制的 0,1,2,3,…,9 这 10 个数字。
- 基数:其实就是表示数的数码个数,如十进制的 10。
- 数位:数码在数中的位置。对于整数,一般右边是低位,左边是高位,从右向左,十进制用个位、十位、百位等表示,其他进制用序号 0,1,2,3,…表示。对小数部分,从左向右编号为−1,−2,−3,…。
- 权:不同位置上的数字的份量,就是这个数位的 1 表示的数值。如对于十进制,个位的权是 1,十位的权是 10,百位的权是 100,序号是 k 的位上的权是 10^k。

进位记数制的进位规则是“逢 N 进一,借一当 N”。对于十进制,$N=10$。

数的表示可以采用不同的数制,计算机科学中常用的有十进制、二进制、八进制和十六进制。

任何一个以 N 为基数的记数系统,其数可以表示为以下形式。

$$(x_{n-1},\cdots,x_1,x_0.y_{-1}y_{-2}\cdots y_{-(m-1)}y_{-m})_N=\sum_{i=0}^{n-1}x_iN^i+\sum_{j=1}^{m}y_{-j}N^{-j}$$

其中,n 表示整数部分的位数,m 表示小数部分的位数,x_i 表示每一位整数数字,y_{-j} 表示每一位小数数字,x_i 和 y_{-j} 是属于 0,1,…,$N-1$ 的数字,N^i 和 N^{-j} 就是位权,$i=0$,1,…,$n-1$,$j=1$,…,m。注意,x_0 的右边是小数点符号。等号左边是 N 进制数的表示形式,右边是这个数对应的十进制数量,所以这个公式也是 N 进制数转换为十进制数的公式,这种运算方式称为按权展开。

1. 十进制

十进制数的数码是 0,1,2,3,4,5,6,7,8,9 共 10 个符号。

基数是 10。

位权是 10^k，k 是数位的序号。

进位规则是“逢十进一，借一当十”。

当多种数制的数同时使用时，同样的数代表的量可能是不同的，如 11，如果是十进制，它代表的数量就是“十一”，如果是二进制数，它代表的数量是“三”。为了区别不同数制表示的数，常在数的后面加上后缀，如用 D、B、Q、H 分别表示十进制、二进制、八进制和十六进制数，如 11B 是二进制数，11Q 是八进制数，AFH 是十六进制数。如果省略，默认是十进制数，如 1011，不写后缀，被认为是十进制数。也常用下标表示，如$(11)_{10}$表示十进制数，$(11)_2$ 表示二进制数，$(11)_{16}$表示十六进制数等。

2. 二进制

二进制数的数码是 0，1 共两个符号。

基数是 2。

位权是 2^k，k 是数位的序号。

进位规则是“逢二进一，借一当二”。

例如，$(1101.1011)_2$ 按权展开：

$$\begin{aligned}(1101.1011)_2 &= 1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+\\ &\quad 0\times2^{-2}+1\times2^{-3}+1\times2^{-4}\\ &=(13.6875)_{10}\end{aligned}$$

习惯上，为了清晰，写二进制数时每 4 位加一个空格隔开，如 1001 0010 1111B。

3. 八进制

八进制数的数码是 0，1，2，3，4，5，6，7 共 8 个符号。

基数是 8。

位权是 8^k，k 是数位的序号。

进位规则是“逢八进一，借一当八”。

例如，$(1101.1011)_8$ 按权展开：

$$\begin{aligned}(1101.1011)_8 &= 1\times8^3+1\times8^2+0\times8^1+1\times8^0+1\times8^{-1}+\\ &\quad 0\times8^{-2}+1\times8^{-3}+1\times8^{-4}\\ &=(577.127197265625)_{10}\end{aligned}$$

4. 十六进制

十六进制数的数码是 0～9 共 10 个数字符号和 A～F 共 6 个字母符号（大小写均可，习惯大写），共 16 个符号。A～F 依次分别相当于十进制的 10～15。

基数是 16。

位权是 16^k，k 是数位的序号。

进位规则是“逢十六进一，借一当十六”。

例如，$(2AE.11)_{16}$按权展开：

$$(2AE.11)_{16}=2\times16^2+10\times16^1+14\times16^0+1\times16^{-1}+1\times16^{-2}$$
$$=(686.06640625)_{10}$$

表 5-1 列出了不同进制的数和十进制数的对应关系。

表 5-1 不同进制的基本数值的对应关系

进制	数值															
十进制	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
二进制	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111
八进制	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
十六进制	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

十六进制中的字母符号,大小写均可,通常用大写。

5. 不同进制数的转换

在不同的场合会用不同进制的数,所以常常需要相互转换。其他进制转十进制的方法就是前述的按权展开,下面介绍十进制转其他进制。

1) 十进制转二进制

十进制转成二进制包括两个部分:十进制整数部分转二进制和十进制小数部分转二进制。

整数部分,除二取余,直到商为零,对得到的余数按先后依次从右向左排列。

小数部分,乘二取整,直到小数为零,按先后依次从左向右排列得到的整数部分。出现循环时,可以根据精度需要取若干位的小数即可。

【例 5-3】 将十进制数 25.3125 转换为二进制数。

【解】 对整数部分,采用"除二取余"的方法。

```
2 | 25      余数
2 | 12      1 ——— 最低位
2 |  6      0
2 |  3      0
2 |  1      1
     0      1 ——— 最高位
```

将得到的余数依次从右向左排列,得到$(25)_{10}=(11001)_2$。

对小数部分,采用"乘二取整"的方法。

0.3125×2=0.625……0

0.625×2=0.25 ……1

0.25×2=0.5 ……0

0.5×2=0.0 ……1

将得到的整数从左到右依次排列,得到$(0.3125)_{10}=(0.0101)_2$

十进制转任意的 N 进制,均可以采用"除 N 取余"和"乘 N 取整"的方法。

2）二进制和八进制相互转换

从表5-1了解到，二进制的000，001，010，011，100，101，110，111依次分别对应八进制的0，1，2，3，4，5，6，7共8个数字。也可以说使用三位的不同的0、1序列刚好可以表示八进制的8个数字符号。所以，一个三位二进制数刚好对应一位八进制数。这就得到了二进制数转八进制数的方法。

二进制转八进制，整数部分，从右向左每三位一组，不够三位时补0，将每三位二进制转换为一位八进制数。小数部分，从左向右每三位一组，不够三位时补0，将每三位二进制转换为一位八进制数。

【例5-4】 将$(10011.11001)_2$转换为八进制。

【解】 按照规则，整数部分，从右向左每三位一组，不够三位时补0；小数部分，从左向右每三位一组，不够三位时补0。

$$(10011.11001)_2 \rightarrow 010\ 011.110\ 010$$

将每三位二进制转换为一位八进制数。

$$010\ 011.110\ 010 \rightarrow 23.62$$

所以：$(10011.11001)_2=(23.62)_8$。

注意：不够三位一定要补0，忘掉补0会得到不同的数，结果就会错误。

反过来，八进制转二进制，将每位八进制数转换为三位的二进制数即可。

【例5-5】 将$(121.61)_8$转换为二进制。

【解】 按照规则，将每位八进制数转换为三位的二进制数。

$$(121.61)_8=(001\ 010\ 001.110\ 001)_2$$

注意：一定要将一位八进制数写成三位二进制数，例如，$(1)_8$要写成$(001)_2$，否则，得到的结果就会是错误的。

3）二进制和十六进制

从表5-1还可以了解到，二进制的0000，0001，0010，…，1111依次分别对应十六进制的0，1，2，…，E，F共16个数码。也可以说使用四位的不同的0、1序列刚好可以表示十六进制的16个数字符号。所以，一个四位二进制数刚好对应一位十六进制数。这就得到了二进制数转十六进制数的方法。

二进制转十六进制，整数部分，从右向左每4位一组，不够4位时补0，将每4位二进制转换为一位十六进制数。小数部分，从左向右每4位一组，不够4位时补0，将每4位二进制转换为一位十六进制数。

【例5-6】 将二进制数1010 1000 1100.1转换为十六进制数。

【解】 首先进行补位和每4位分隔后得到：1010　1000　1100.　1000，每4位转换为一位十六进制数，十六进制结果为A8C.8H。

十六进制转二进制，将每位十六进制数转换为4位二进制数。

【例5-7】 将十六进制数A2.B3转换为二进制数。

【解】 首先从小数点开始分别向左右查表扩写如下：1010　0010.　1011　0011，最后写出紧凑的二进制结果10100010.10110011B。

十进制到八进制、十六进制的转换可以借助于二进制来进行。

6. 二进制的算术运算

二进制数的算术运算与十进制类似,也有加减乘除四则运算,对于二进制数字 0 和 1 来说,其算术运算规则如表 5-2 所示。

表 5-2　二进制的算术运算规则

加法	0+0=0	0+1=1	1+0=1	1+1=10
乘法	0×0=0	0×1=0	1×0=0	1×1=1
减法	0−0=0	0−1=−1	1−0=1	1−1=0
除法		0÷1=0		1÷1=1

知道了二进制算术运算规则之后,就可以进行任意位的二进制整数和实数的算术运算了。

【例 5-8】 二进制整数的四则运算。计算下式的值(均为二进制,结果也是二进制)。

10101010+00101010=?

01000000−00001010=?

1001110×1011101=?

10111010÷110=?

【解】

10101010+00101010=11010100

01000000−00001010=00110110

1001110×1011101=1110001010110

10111010÷110=11111

【例 5-9】 二进制实数的四则运算。计算下式的值(均为二进制,结果也是二进制)。

0.11001+1.01010=?

1.1000011−0.0101101=?

0.11011×0.10011=?

0.1010÷0.1011=?

【解】

0.11001+1.01010=10.00011

1.1000011−0.0101101=1.0010110

0.11011×0.10011=0.1001000001

0.1010÷0.1011=0.111…

5.2　数的编码

本节主要介绍计算机中整数和实数的表示方法。由于它们在形式上与数学上的表示方式不相同,而更重要的是一种代号的概念,所以称为编码。编码的含义是用某个符号代

表另一个事物。

5.2.1　整数的编码

首先介绍几个概念，即字、字长、真值和机器数，然后介绍机器数的编码。

1. 字与字长

字是指计算机中作为一个整体来处理或运算的一串数码。这串数码的长度称为**字长**。如现在说的 64 位计算机或 64 位操作系统，它们一次可以处理 64 位的二进制数据，64 位是一个字，64 就是字长。

2. 机器数和真值

前面讨论的数未涉及符号，称为**无符号数**。实际中的数有正有负，带有正负号的数称为**有符号数**或**带符号数**。数学中的正负号用“＋”“－”表示，而计算机中所有的信息都要转换为 0、1 序列。通常规定，用 0 表示正号，1 表示负号，放在一个数的最高位。例如，设机器的字长为 8 位，则第 7 位（即最高位）为符号位，$(0000\ 1001)_2$ 表示＋9，$(1000\ 0101)_2$ 表示－5。

符号被数值化了的（二进制）数称为**机器数**。而把原来带“＋”“－”号的数称为**真值**，如$(+0001001)_2$，$(-0000101)_2$。

3. 机器数的编码

一个数可以使用不同的形式表示，计算机中的数有原码、反码和补码三种形式。

1）**原码**

原码就是机器数，即真值的绝对值加上符号位，即用最高位表示符号，其余位表示值。例如，对于字长为 8 位的机器来讲，＋1 的原码为 0000 0001，－1 的原码为 1000 0001，记为

$$[+1]_{原}=0000\ 0001,\quad [-1]_{原}=1000\ 0001$$

＋0 的原码为 0000 0000，－0 的原码为 1000 0000，说明 0 的原码不唯一。

对于 8 位字长，由于最高位即第 7 位为符号位，所以 8 位二进制机器数的取值范围为 1111 1111～0111 1111，即十进制－127～＋127。

2）**反码**

规定反码的表示方法为：正数的反码是其原码；负数的反码是在其原码的基础上，保持符号位不变，其余数据位的各个位取反，即 0 变为 1，1 变为 0。例如，字长 8 位，＋1 和－1 的原码为 0000 0001 和 1000 0001，其反码分别为 00000001 和 11111110。记为

$$[+1]_{反}=0000\ 0001,\quad [-1]_{反}=1111\ 1110$$

＋0 的反码为 0000 0000，－0 的反码为 1111 1111，说明 0 的反码不唯一。

3）补码

规定补码的表示方法为：正数的补码和其原码相同，负数的补码是在其反码的基础上加 1。例如，字长为 8 位，＋1 和－1 的反码分别为 00000001 和 11111110，其补码分别

为 00000001 和 11111111,记为

$$[+1]_{补}=0000\ 0001,\quad [-1]_{补}=11111111$$

计算机中采用补码的优点是,可以将加法、减法运算统一简化为加法运算。例如:对于式子 5−5=0,可以改写为(5)+(−5)=(0),而 5 的补码为 0000 0101,−5 的补码为 1111 1011,这里假定机器字长为 8 位。由于 0000 0101+1111 1011=1 0000 0000,去掉溢出的第 8 位,00000000 正好为 0 的补码。

再如对于式子 3−5=−2,可以改写为(3)+(−5)=(−2),而 3 的补码为 0000 0011,−5 的补码为 1111 1011,这里假定机器字长为 8 位。由于 0000 0011+1111 1011=1111 1110,1111 1110 正好为−2 的补码。

事实上,计算机中的整数是用补码表示的,而运算也是补码运算,将减法转为加法。

4) 特殊数补码

假定字长为 8 位。

0 有+0 和−0,+0 的原码、反码和补码均为 0000 0000;−0 的原码、反码和补码分别为 1000 0000、11111 1111、1 0000 0000,去掉溢出最高位第 8 位 1,得到−0 的补码也为 0000 0000。这样 0 的补码是唯一的。

+127 的原码、反码和补码均为 0111 1111;−127 的原码、反码和补码分别为 1111 1111、1000 0000 和 1000 0001。

由于补码 1000 0000 未被占用,所以规定它代表−128 的补码形式,而−128 没有原码和反码。8 位补码的表示范围是−128～127。

对于 n 位二进制数,原码的表示范围为$-(2^{n-1}-1)\sim+(2^{n-1}-1)$,补码的表示范围为$-2^{n-1}\sim+(2^{n-1}-1)$。补码可以定义为

$$[X]_{补}=\begin{cases}X & 0\leqslant X<2^{n-1}\\ 2^n+X=2^n-|X| & -2^{n-1}\leqslant X<0\end{cases}$$

5) 补码的简便计算

设字长为 n 位,则正数的补码是其原码,0 的补码是 n 个 0,−1 的补码是 n 个 1,-2^{n-1}的补码是 1 后面 $n-1$ 个 0,其他负数的补码:先写出其原码,然后从右向左的连续 0 和第一个 1 不变,再向左按位取反,符号位不变。如 8 位字长,$-2^{8-1}=-128$ 的补码是 1000 0000,$[-52]_{原}=1011\ 0100$,$[-52]_{补}=1100\ 1100$。

4. 溢出问题

由于计算机表示数据时用的二进制位是有限的,例如 8 位,那么它能表示的数的范围就是有限的。8 位补码能表示的数的范围是−128～127。这样两个同符号的数相加时就可能超出 8 位二进制能表示的范围,例如 127+1=128,这种现象称为溢出。大于能表示的最大正数称为上溢,小于能表示的最小负数称为下溢。

如何判断溢出呢?方法是:异号相加不会溢出;同号相加,最高位变号则溢出,最高位不变号则不溢出,而不管是否向 D8 进位。因为异号相加,结果的绝对值小于两数的绝对值,一定在能表示的范围内;而两正数相加应该为正数,两负数相加应该为负数,如果符

号位改变，说明结果是不正确的，本质上是产生了溢出。

例如，补码运算 1000 0001＋0100 1101＝1100 1110，异号相加，不溢出。0100 0000＋0100 0000＝1000 0000 两个正数相加，符号位改变（结果为负数），溢出。1100 0000＋1100 0000＝(1)1000 0000，两个负数相加，符号位没变，不溢出，尽管有向 D8 位进 1（括号中的 1），这个 1 会被自然舍去，而 8 位的最高位没变。

注意：D8 表示整数部分序号是 8 的数位。

【例 5-10】 若计算机采用 8 位二进制，判断下列计算是否会溢出，说明原因。

① $(-32)_{10}+(-32)_{10}$

② $(-119)_{10}+(-63)_{10}$

③ $(-63)_{10}+(74)_{10}$

④ $(67)_{10}+(62)_{10}$

⑤ $(79)_{10}+(35)_{10}$

⑥ $(-63)_{10}+(-63)_{10}$

【解】 （本例未标几进制的是二进制补码形式）11100000 代表－32 的补码，其最高位为 1，计算 $(-32)_{10}+(-32)_{10}$＝1110 0000＋1110 0000＝1 1100 0000，出现进位，舍掉进位得到 1100 0000＝$-(64)_{10}$，这是假溢出，新的最高位仍为 1，计算结果正确。

$(-119)_{10}+(-63)_{10}$＝1000 1001＋1100 0001＝1 0100 1010，舍去掉 D8 的进位得到 0100 1010＝$(+74)_{10}$，符号位改变，溢出。

$(-63)_{10}+(74)_{10}$＝1100 0001＋0100 1010＝1 0000 1011，舍掉 D8 的进位得到 0000 1011＝$(+11)_{10}$，不溢出。

$(67)_{10}+(62)_{10}$＝0100 0011＋0011 1110＝1000 0001＝$(-127)_{10}$，无进位，但符号位改变$(-127)_{10}$，溢出。

$(79)_{10}+(35)_{10}$＝0100 1111＋0010 0011＝0111 0010＝$(+114)_{10}$，无进位，符号位不变，不是溢出。

$(-63)_{10}+(-63)_{10}$＝1100 0001＋1100 0001＝1 1000 0010，去掉进位得到 1000 0010＝$(-126)_{10}$，符号位不变，不是溢出。

5.2.2　浮点数的编码

数学中的实数包含整数、有限位小数、无限循环小数和无限不循环小数，计算机中的实数一般有两种表示形式，即定点数和浮点数。

1. 定点数和浮点数

定点数将小数点始终固定于实数全部数字中间的某个位置上。例如，货币值一般采用的就是这种定点方式，假如共有 4 位精确度，小数点固定在第二位数之后，如 87.60 或者 03.52。

定点数表示形式尽管简单，但其缺点在于形式过于死板，当小数点位置固定下来后，也就决定了整数部分和小数部分都具有固定位数，一旦遇到特别大或者特别小的数时就不胜任了。浮点数表示数据更灵活，可以有更大的表示范围，它利用科学记数法来表示实