第1章 微机的基础知识

本章内容

学习目标

学习二进制、八进制和十六进制；各种进位制的运算规则以及相互转换方法。学习机器数的表示方法以及计算机中常用的编码。介绍微机的组成，特别是由CPU和内部存储器组成的主机及指令和程序的运行。

计算机内部信息的存储和处理采用二进制数，要在计算机内对以数字、字符、声音或图像等形式表达的信息进行处理，必须将其转换成计算机能识别的形式，即由‘0’和‘1’组成的二进制代码。本章介绍二进制、十六进制以及它们的运算和相互转换的规则，学习数在计算机内的表示方法以及数字和字符的常用编码。

CPU 和内部存储器是组成微机的主要部分，称为微机的主机，学习它们的工作原理和工作过程对理解整机至关重要。本章的后一部分介绍微机的逻辑组成、CPU 和内部存储器的组成原理以及 CPU 对内存读写的过程。

长期以来人们习惯用数字表示事物数量的特性。数量可能是无限的，而表示数量的数码是有限的，因此产生了进位计数制。进制各不相同：每七天为一周——七进制；十二个月为一年——十二进制；六十分为一小时——六十进制等。但使用最多的还是十进制。

1.1.1 十进制

在十进制中，有十个表示数的符号：0、1、2 … 9，称为数码。每计够十个数就要向高位进位，即逢十进一，10 称为十进制的基数。同一个数码在数中不同位上表示的量是不同的。例如，“1”在个位上表示 1，而在十位上则表示 10，百位上表示 100 等。数码‘1’在每一位上所表示的量，称为该位的权。十进制各位的权如表 1.1.1 所示。

表 1.1.1 十进制各位的权

	整数部分						小数部分				
位	n	n－1		3	2	1	1	2		m－1	m
权	10^{n-1}	10^{n-2}	…	10^2	10^1	10^0	10^{-1}	10^{-2}	…	$10^{-(m-1)}$	10^{-m}

各位的权都是基数 10 的整数次幂。每一位数码所表示的量等于该数与权的乘积。一个多位数字的值等于各位数字与它的权的乘积之总和。例如一个十进制数 1234.56 所表示的数值为：

$1\times10^3+2\times10^2+3\times10^1+4\times10^0+5\times10^{-1}+6\times10^{-2}$。

一个有 n 位整数和 m 位小数的十进制数 $a_{n-1}a_{n-2}\cdots a_2a_1a_0.b_1b_2\cdots b_{(m-1)}b_m$，它的数值可按各位数字加权相乘求和的方法计算，其计算公式如下：

$$(a_{n-1}a_{n-2}\cdots a_2a_1a_0.b_1b_2\cdots b_{(m-1)}b_m)_{10}=a_{n-1}\times10^{n-1}+a_{n-2}\times10^{n-2}+\cdots a_2\times10^2+a_1\times10^1+a_0\times10^0+b_1\times10^{-1}+b_2\times10^{-2}+\cdots b_{(m-1)}\times10^{-(m-1)}+b_m\times10^{-m}$$

总之，一种进位制包括它使用的数码和个数、进位的规则、各位的权以及计算数值的方法。

1.1.2 二进制

由于还未找到一种简单且具有十种稳定状态的器件，在计算机中直接表示十进制数是困难的。表示基数为 2 的二进制数却十分容易，许多电子器件都具有两种截然相反的

稳定状态。例如，发光二极管的发光与熄灭，晶体管的导通与截止等。两种稳定状态可以分别表示二进制中的数码“0”和“1”。因此，计算机中使用二进制。

为了区分不同的数制，在计算机的书刊或程序中容易混淆的地方一般在数字后面跟一个英文字母以示区别。二进制用 B(Binary)表示，十进制用 D(Decimal)表示，十六进制用 H(Hexadecimal)表示，而八进制用 Q(Octal)表示，以避免字母“O”与数字“0”混淆。

1. 二进制

二进制有两个数码：“0”和“1”，基数为 2，按“逢二进一”的规则进位。各位的权如表 1.1.2 所示。

表 1.1.2 二进制各位的权

	整数部分						小数部分				
位	n	n-1		3	2	1	1	2		m-1	m
权	2^{n-1}	2^{n-2}	…	2^2	2^1	2^0	2^{-1}	2^{-2}	…	$2^{-(m-1)}$	2^{-m}

若用 a_0、a_1、a_2、…、a_{n-1} 分别表示二进制数整数的第一、第二…和第 n 位，用 b_1、b_2、b_3…b_m 分别表示二进制数小数的第一、第二…和第 m 位，其中的 a、b 只能取 1 和 0，则

$$a_{n-1}a_{n-2}\cdots a_2a_1a_0.b_1b_2b_3\cdots b_m$$

表示的就是一个二进制数。其十进制的数值仍按加权相乘求和法计算，计算公式如下：

$$(a_{n-1}a_{n-2}\cdots a_2a_1a_0.b_1b_2\cdots b_{(m-1)}b_m)_2 = a_{n-1}\times 2^{n-1} + a_{n-2}\times 2^{n-2} + \cdots a_2\times 2^2 + a_1\times 2^1 + a_0\times 2^0 + b_1\times 2^{-1} + b_2\times 2^{-2} + \cdots b_{(m-1)}\times 2^{-(m-1)} + b_m\times 2^{-m}。$$

上式也是二进制数转换为十进制数的基本公式。应当注意，整数位的权等于基数的位数减 1 的幂。例如，第一位整数的权为 $2^{1-1}=2^0=1$，第 i 位整数的权为 2^{i-1}。而小数位的幂就等于基数位数的幂。例如第一位小数位的权为 $2^{-1}=0.5$，第 j 位小数位的权为 2^{-j}。

将二进制整数位和小数位的权列在表 1.1.3 中，以便记忆。

表 1.1.3 二进制整数位和小数位的权

整数位数	1	2	3	4	5	6	7	8
权	1	2	4	8	16	32	64	128
小数位数	1	2	3	4	5	6		
权	0.5	0.25	0.125	0.0625	0.03125	0.015625		

2. 十进制数转换为二进制数

可用多种方法把一个数由十进制转换成二进制，降幂法和逐次相除／相乘法就是其中的两种。

(1)降幂法

降幂法的基本方法是：对一个给定的十进制数 a，在二进制各位的权中找一个比 a 小

且为最大的权 b，若 a－b＞0，则与该权对应的位取 1，否则，该位取 0。然后将余数 a－b 按此法类推，确定下一位，直到余数为 0 时，前次的余数 1 即为最低位。

例 1 用降幂法将 117D 转换为二进制数。

$$
\begin{aligned}
117-64=53 \quad & (64=2^6, \quad a_6=1)\\
53-32=21 \quad & (32=2^5, \quad a_5=1)\\
21-16=5 \quad & (16=2^4, \quad a_4=1)\\
5-8<0 \quad & (8=2^3, \quad a_3=0)\\
5-4=1 \quad & (4=2^2, \quad a_2=1)\\
1-2<0 \quad & (2=2^1, \quad a_1=0)\\
1-1=0 \quad & (1=2^0, \quad a_0=1)
\end{aligned}
$$

于是 117D＝1110101B。

(2)逐次相除／相乘法

这种方法对整数和小数分别进转换，整数采用逐次相除法，小数用逐次相乘法。将转换的十进制整数除以 2，其余数为二进制的最低位，即若能被整除，最低位为 0，否则为 1。此后，将前次所得的商再除以 2，其余数即为低位的二进制数字。此法不断进行，直到余数为 1 时，即为最高位。

例 2 用逐次相除法将 117D 转换为二进制数。

		余数	
2	117	…… 1	……最低位
2	58	…… 0	
2	29	…… 1	
2	14	…… 0	
2	7	…… 1	
2	3	…… 1	
	1	…… 1	……最高位

于是 117D＝1110101B。

一个二进制小数转换成十进制小数使用逐次相乘法，即给小数部分乘 2，其乘积的整数位就是的二进制数的第一个小数位。若整数位为 1，则第一小数位为 1，否则为 0。然后去掉整数，再乘以 2，其乘积的整数位为第二小数位。依次进行，每乘一次 2，二进制小数位向右移动一位，直至十进制的小数部分为零。

例 3 用逐次相乘法将 0.6875D 转换为二进制小数。

$$
\begin{aligned}
0.6875\times 2=1.375 \quad & \text{整数部分为 } 1\\
0.375\times 2=0.75 \quad & \text{整数部分为 } 0\\
0.75\times 2=1.5 \quad & \text{整数部分为 } 1\\
0.5\times 2=1.0 \quad & \text{整数部分为 } 1
\end{aligned}
$$

于是 0.6875D＝0.1011B。

例 4 将 79.125D 转换为二进制数。

整数：

```
2 | 79   ……  1   ……最低位
2 | 39   ……  1
2 | 19   ……  1
2 |  9   ……  1
2 |  4   ……  0
2 |  2   ……  0
    1    ……  1   ……最高位
```

小数：$0.125 \times 2 = 0.25$　　整数部分为 0

$0.25 \times 2 = 0.5$　　整数部分为 0

$0.5 \times 2 = 1.0$　　整数部分为 1

于是 $79.125D = 1001111.001B$。

思考：十进制的 0～15 与相应的二进制数要经常使用，请把相应的二进制数填入表 1.1.4 中，并熟悉记忆，以备以后使用。

表 1.1.4　二进制数与十进制数的对应关系

十进制	0	1	2	3	4	5	6	7
二进制								
十进制	8	9	10	11	12	13	14	15
二进制								

1.1.3　十六进制

1. 十六进制

二进制数只是 0 和 1 的序列，数字的构成比较单调。同一数值用二进制表示数字较长，而且与十进制之间没有直接的对应关系，给阅读、书写和记忆带来不便。采用十六进制可以克服上述的缺点。

十六进制应有 16 个数码，除了使用十进制的 0～9 十个数码外，增加了字母字符 A、B、C、D、E 和 F 六个数码。它们与十进制和二进制数对应关系如表 1.1.5 所示：

表 1.1.5　十六进制数与十进制数的对应关系

十六进制	0	1	2	3	4	5	6	7
十进制	0	1	2	3	4	5	6	7
二进制	0000	0001	0010	0011	0100	0101	0110	0111
十六进制	8	9	A	B	C	D	E	F
十进制	8	9	10	11	12	13	14	15
二进制	1000	1001	1010	1011	1100	1101	1110	1111

十六进制的前十个数码与十进制相同，后六个数码分别用 A、B、C、D、E 和 F 表示，相当于十进制的 10～15。因此，十六进制与十进制有明显的对应关系。

十六进制采用逢十六进一的规则进位，它各位的权如表 1.1.6 所示：

表 1.1.6 十六进制各位的权

位	1	2	3	4	5	…
权	1	16	256	4096	65536	…

2. 十六进制数的转换

(1)十六进制数转换为十进制数

十六进制数转换为十进制数也用加权相乘求和法计算。

例 5 将十六进制数 9EH、3DA5H 转换成十进制数。

$9EH = 9 \times 16 + 14 = 158D$。

$$\begin{aligned} 3DA5H &= 3 \times 16^3 + 13 \times 16^2 + 10 \times 16^1 + 5 \times 16^0 \\ &= 3 \times 4096 + 13 \times 256 + 10 \times 16 + 5 \times 1 \\ &= 15781D。 \end{aligned}$$

(2)十进制数转换为十六进制数

与十进制数转换为二进制数的方法相同，使用降幂法或逐次相除法可将十进制转换为十六进制。

例 6 分别用降幂法和逐次相除法将 107D 转换为十六进制。

①降幂法

$$\begin{aligned} 107D &= 6 \times 16 + 11 \\ &= 6BH。 \end{aligned}$$

②逐次相除法

$$16 \overline{)\ 107} \quad -96 \qquad 11D(BH)$$
$$\quad\ 6 \qquad\qquad\ 6$$

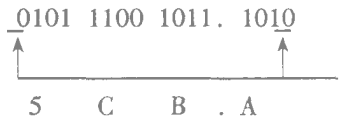
于是 $107D = 6BH$。

(3)二进制与十六进制的转换

与十进制和十六进制间的转换相比，二进制和十六进制间的转换更容易和方便，这是因为二进制与十六进制之间有更简单的对应关系。从表 1.1.5 可以看出：一位十六进制数总是与四位二进制数相对应。因此，可将四位二进制数“压缩”成一位十六进制数。反之，可将一位十六进制数‘扩张’为四位二进制数。

二进制数转换为十六进制数的步骤是：首先，从二进制数的小数点起整数部分向左、小数部分向右，每四位划为一组，整数部分的最左一组或小数部分的最右一组不足四位时，用 0 补足四位。然后，从表 1.1.5 的十六进制与二进制的对应关系中找出四位二进制数对应的十六进制数，按照原来的分组对应顺序把十六进制数排列起来，即为转换的十六进制数。

例 7 将二进制数 101 1100 1011.101B 转换为十六进制数。



于是 101 1100 1011.101B = 5CB.AH。

用与上述相反的过程可将十六进制转换为二进制，即将一位十六进制数扩展为四位二进制数。

例 8 将十六进制数 3E9.4CH转换为二进制数。

3	E	9	.	4	C
11	1110	1001	.	0100	1100

于是 3E9.4CH = 0011 1110 1001.0100 11B。

存储器存储信息的最小单元是字节，可存放八位二进制数，八位二进制数恰好用两位十六进制数表示。因此，两个十六进制数可以表示一个字节的数据。用十六进制表示计算机内的信息不仅与其存储方式有简单的对应关系，而且易读易记，因而得到了广泛的使用。

1.1.4 八进制

八进制也是计算机中常用的一种数制。八进制有 8 个数码 0～7，基数为 8，采用逢八进一的进位规则。八进制整数位的权分别为：1、8、64、512…。使用降幂法或逐次相除法可将十进制数转换为八进制数。八进制与二进制之间也有简单的对应关系：即三位二进制数对应一位八进制数。因此，很容易实现八进制与二进制之间的转换。下面通过例子说明各种数制间的转换方法。

例 9 将十进制数 53D 转换成八进制数。

53D = 6 × 8 + 5
= 65Q。

例 10 将二进制数 1111101B 转换为八进制数。

001	111	101

于是 1 111 101B = 175Q。

例 11 将八进制数 275Q转化为十六进制；把十六进制数 CAH 转换为八进制。

一般不直接进行八进制和十六进制间的转换，而是通过二进制间接进行，即先将八进制或十六进制都转换成二进制，然后再按三位或四位一组划分，最后转换成所需的数制。

275Q = 010 111 101B = 1011 1101B = BDH。

CAH = 1100 1010B = 011 001 010B = 312Q。

1.1.5 二进制数的运算

1. 二进制的运算规则

二进制运算包括算术运算和逻辑运算。在算术运算中，加法和乘法是基本运算。加法的运算规则是：

$0+0=0$　　$0+1=1$

$1+0=1$　　$1+1=1$（产生进位）。

乘法的运算规则是：

$0\times0=0$　　$0\times1=0$

$1\times0=0$　　$1\times1=1$。

二进制的逻辑运算包括与运算、或运算和非运算。

逻辑与运算的规则是：两个一位二进制数中只要有一个为 0，其与运算的结果为 0；只有两者均为 1 时，与运算的结果才为 1。与运算的运算符是 and 或 ∧，例如，0 and 1 或 0∧1 均表示 0 和 1 的与运算。上述规则可用下面一组表达式表示：

$0\wedge0=0,\quad 0\wedge1=0,\quad 1\wedge0=0,\quad 1\wedge1=1$。

逻辑或运算的规则是：两个一位二进制数中只要有一个为 1，其或运算的结果为 1；只有两者均为 0 时，或运算的结果才为 0。或运算的运算符是 or 或 ∨。例如，0 or 1 或 0∨1 均表示 0 和 1 的或运算。上述规则可用下面一组表达式表示：

$0\vee0=0,\quad 0\vee1=1,\quad 1\vee0=1,\quad 1\vee1=1$。

二进制数中的 0 和 1 是互为相反数，逻辑非运算就是取相反数的运算。逻辑非运算的运算符是 not 或在数字上加“-”，例如：not 1 或 $\bar{1}$，表示对 1 作非运算或取反。非运算的规则如下：

not $0=1$,　　not $1=0$。或 $\bar{1}=0$,　　$\bar{0}=1$。

2. 二进制数的运算

根据以上二进制的运算规则，就可以对二进制数进行算术或逻辑的运算。通过下面的例题说明各种运算的方法。

例 12 求 $1001011+100101=1110000$。

$$\begin{array}{r} 1001011 \\ +\ 100101 \\ \hline 1110000 \end{array}$$

例 13 求 $1011\times101=110111$。

$$\begin{array}{r} 1011 \\ \times\ 101 \\ \hline 1011 \\ 0000 \\ +\ 1011 \\ \hline 110111 \end{array}$$

例 14 求 11010 and 10111 = 10010。

$$\begin{array}{ccccc} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 0 & 0 & 1 & 0 \end{array}$$

例 15 求 10010 or 10111 = 10111。

1 0 0 1 0
1 0 1 1 1
↓ ↓ ↓ ↓ ↓
1 0 1 1 1。

1.1.6 机器数的表示

以上所讨论的数均为不带符号的正数，而一般的数不仅有小数，也可能是带符号的正负数。在计算机中如何表示小数点位，如何表示数的符号是这里重点讨论的问题。

1. 机器数

由于受计算机内部结构的限制，计算机中的数有以下几个特点：

(1)计算机是用电路元件的状态来表示数的，基本的电路元件只有两种截然相反的状态,由于它只能表示 0 和 1,因此,计算机中只能使用二进制数。

(2）机器中数的长度或它的位数是确定的。机器中存储一个 0 或 1,称为位(b,bit),它是计算机所能表示的数的最小单位。计算机的微处理器一次处理或运算的二进制数称为字(W,Word)。一个字所包含的位数，称为字长。计算机的字长受微处理器结构的限制,在八位机中,字长是 8 位，即一个字节;16 位机，字长是两字节;32 位机,字长是四字节。

(3)由于可以事先约定机器中数的小数点的位置，比如可以设定它位于最低位之后(该数为整数),也可设定它在最高位之前(即该数为一小数),因此,小数点无须用特定的符号表示。

(4)机器中数的符号应能同数本身一起参与运算，即要求符号数值化。通常用“ + ”或“ - ”表示数的正、负,例如 + 110、- 101 等。当对这两个数求和时，要根据符号决定对它们进行加法还是减法运算，而符号本身并不加减。

在计算机中,所有的信息都必须用二进制数表示,符号也不例外,例如用 0 表示“ + ”,而用 1 表示“ - ”。这样一来,符号代码与数值代码并无区别,若符号位也能同数值部分一起参与运算，并得到正确的结果，运算过程将得以简化。具有上述特点的符号，称为符号的数值化。

在计算机中，经过数值化处理的数，称为机器数。八位的机器数在计算机中存储的映像如图 1.1.1 所示。其中最高位作符号位,其余 7 位为数值部分。

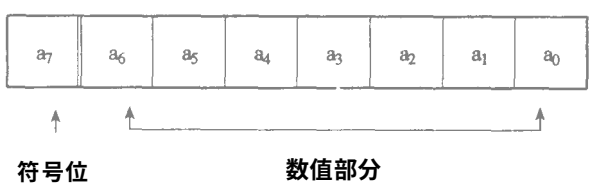


图 1.1.1 机器数在计算机中存储的映像

2. 真值、原码、反码和补码

为使机器数的符号数值化，必须对带符号数进行编码，常用的编码有原码、反码和补码。下面以 8 位字长的机器数为例，说明三种编码的编码规则。对如图 1.1.1 所示的 8

位机器数,规定其最高位 a_7 为符号位,其低 7位 $a_6 \sim a_0$ 表示数值。

(1)真值

为了理解机器数的编码规则，首先介绍真值的概念。机器数原来的实际值，称为真值。真值的数值一般用十进制或二进制表示,而符号仍用'+'、'-'表示。因其符号还没有数值化，真值还不是机器数。

例如,两个变量 $X_1 = +91D = +101\ 1011B$ 和 $X_2 = -91D = -101\ 1011B$。X_1, X_2 就是用真值表示的。

(2)原码

原码的编码规则是：保持其真值的数值部分代码不变，在其最高位加符号位。正数加 0 负数加 1。

例 16 设 $X_1 = +91D = +101\ 1011B$ 和 $X_2 = -91D = -101\ 1011B$。求 X_1和 X_2 的原码。

若用$[X_1]_{原}$和$[X_2]_{原}$分别表示 X_1 和 X_2 的原码,则

$$[X_1]_{原} = 0101\ 1011B,$$

$$[X_2]_{原} = 1101\ 1011B。$$

数 0可以有 +0、-0。若 令 $X_3 = +000\ 0000B, X_4 = -000\ 0000B$,虽然 $X_3 = X_4$,但它们的原码却不相同。

$$[X_3]_{原} = 0000\ 0000B,$$

$$[X_4]_{原} = 1000\ 0000B。$$

因此,0 的原码有两个。

八位的机器数有 7位表示数值，其最大和最小值的真值分别为 $+111\ 1111B$ 和 $-111\ 1111B$; 对应的原码分别为 0111 1111B 和 1111 1111B。由此可见，八位的原码表示的数值范围是 $-127 \sim -0, +0 \sim +127$,包括两个 0,共 256 个数。

原码的编码规则简单，与真值有直接的对应关系。但运算时还必须对符号位进行判断，然后才能对数值部分进行运算，不便于计算机操作。

(3)反码

在原码的基础上很容易得到其反码。正数和负数反码的编码规则是不同的。正数的反码与其原码相同。负数的反码是保持其原码的符号位不变，仍为 1;其原码的数值部分按位取反。所谓按位取反是将各位的 1 变为 0,而将 0变为 1 的操作。负数反码就是将其原码除符号位外按位取反。

例 17 设 $X_1 = +91D = +101\ 1011B$ 和 $X_2 = -91D = -101\ 1011B$。求 X_1 和 X_2 的反码。

用$[X_1]_{反}$和$[X_2]_{反}$分别表示 X_1 和 X_1的反码,则

$$[X_1]_{反} = 0101\ 1011B。$$

$$[X_2]_{原} = 1101\ 1011B,$$

$$[X_2]_{反} = 1010\ 0100B。$$

若令 $X_3 = +000\ 0000B, X_4 = -000\ 0000B$,则其反码分别为：

$$[X_3]_{反} = [X_3]_{原} = 0000\ 0000B。$$

$$[X_4]_{原} = 1000\ 0000B,$$

$$[X_4]_{反} = 1111\ 1111B。$$

因此,0 的反码也有两个。

+127D(+111 1111B)的反码为 0111 1111B，-127(-111 1111B)的反码 1000 0000B。八位机器码所表示的数值范围是 -127～-0，+0～+127，与原码相同。

(4)补码

机器数在计算机中存储时大多都采用补码。因为补码的符号位能和数值位一起直接参与运算，同时能把对两个数的减法运算转换成对其补码的加法运算。从而简化机器的设计。

① 补码的概念

首先讨论十进制数的"补码"。观察 27、17 和 7 等几个数对模数 10 的模运算。所谓 a 对模数 b 的模运算就是求 a 除以 b 所得的余数的运算，并用表达式 a MOD b 表示。于是有：

7 MOD 10 = 7，

17 MOD 10 = 7，

27 MOD 10 = 7。

可以看出，一个数加上模数的整倍数的模运算其值不变，即 (a+nb)MOD b = a MOD b (n 为整数)。由于 27、17 和 7 对模数 10 有相同的余数，称它们是互余的。若(a+nb)MOD b 式中的 a<0，例如 a=-3，那么，-3 MOD 10 =(-3+10)MOD 10 = 7 MOD 10 = 7。因此，-3 与 7 也是互余的。

对于模数 10，一个数，例如 8，分别与几个同余数相加，其结果是相同的，即：

8+7=5，　(对于 MOD 10)

8+(-3)=5。

比较以上两式可看出：8-3=8+7。由此可得如下的结论：8 与 3 的减法运算可用 8 与 7 的求和运算替代，于是减法运算转换为加法运算了，这正是我们所期待的。由于 7=-3+10，因此称 7 是 -3 的补数。

设 b 为模数，对于模数 b，数 a 的补数等于 a+b。由以上的讨论可知，当 a>0 时，a=a+b 其补数就是自己当 a<0 时，其补数为 a+b。还以 10 位模数，8 的补数就是 8，-4 的补数等于 -4+10=6，因此，6 就是 -4 的补数。

模数不同，一个数的补数是不同的。例如对于模数 10，-4 的补数是 6。若模数为 100 时，-4 的模数是 96 了。

思考 以 256 为模，+53 和 -53 的补数分别是多少？

把上面的概念应用到二进制中，就引出了二进制数的补码。n 位二进制数能表示数的个数为 2^n，2^n 就是它的模数。对一个真值为 X 的 n 位二进制数：

若 X>0 则 $[X]_{补}=[X]_{原}$。若 X<0，则 $[X]_{补}=X+2^n$。

仍以 8 位二进制数为例，其模为 2^8 = 1 0000 0000B。若 X=53=+011 0101B，其补码：

$[X]_{补}=[X]_{原}$ = 0011 0101B。

若 X=-53=-011 0101B，其补码：

$[X]_{补}$ = 1 0000 0000 - 011 0101 = 1100 1011B。

思考：若将以上两个补码直接转换成十进制数，它们的值与上面思考题的结果相同吗？

②二进制补码的编码规则

二进制数的补码可按如下的简便方法得出:对正数,其补码与其原码相同;对负数,保持其原码的符号位 1 不变,其余各位按位取反后加 1。

例 18:设 $X_1 = +91D = +101\ 1011B$ 和 $X_2 = -91D = -101\ 1011B$。求 X_1 和 X_2 的补码。

$[X_1]_{补} = [X_1]_{原} = 0101\ 1011B$。

$[X_2]_{原} = 1101\ 1011B$,

$[X_2]_{反} = 1010\ 0100B$,

$[X_2]_{补} = 1010\ 0101B$。

例 19:求 +0 和 −0 的补码。

设 $X_3 = +0000000B$,

$[X_3]_{补} = [X_3]_{原} = 0000\ 0000B$。

设 $X_4 = -000\ 0000B$,

$[X_4]_{原} = 1000\ 0000B$,

$[X_4]_{反} = 1111\ 1111B$,

$[X_4]_{补} = [X_4]_{反} + 1 = 1111\ 1111B + 1 = \boxed{1}\ 0000\ 0000B$。

由于受八位数据长度的限制,带框的 1 被舍去。因此,0 的补码只有一种形式。

③八位二进制补码表示数的范围

+127D 的补码与原码相同为:0111 1111B。

−1D 的补码:$[-1D]_{原} = 1000\ 0001B$,$[-1D]_{反} = 1111\ 1110B$,

$[-1D]_{补} = 1111\ 1111B$。

−127D 的补码:$[-127D]_{原} = 1111\ 1111B$,$[-127D]_{反} = 1000\ 0000B$,

$[-127D]_{补} = 1000\ 0001B$。

因为,$-128D = (-127D) - 1$,$[-128D]_{补} = [-127D]_{补} - 1 = 1000\ 0000B$。所以,$[-128D]_{补} = 1000\ 0000B$。

八位二进制数总共可以表示 256 个数,而 0 的补码只有一个,因此,八位二进制补码所表示数的范围为 $-128 \sim 0 \sim +127$。现将八位机器数的各种编码与其数值范围列于表 1.1.7 中,以供参考。

思考:为什么 −128 没有相应的原码和反码却有补码呢?

④补码的运算法则

设 A、B 均为正数,两数的补码运算遵循以下法则:

a. $[A+B]_{补} = [A]_{补} + [B]_{补}$.

b. $[A-B]_{补} = [A]_{补} + [-B]_{补}$.

c. $[[A]_{补}]_{补} = [A]_{原}$.

例 20:利用补码的计算验证以下各十进制表达式的正确性。

a. $47 + 69 = 116$。

b. $47 - 89 = -42$。

c. $-87 + 39 = -48$。

表 1.1.7 八位机器数的编码表

十进制数	二进制数	原码	反码	补码
+0	+000 0000	0000 0000	0000 0000	0000 0000
+1	+000 0001	0000 0001	0000 0001	0000 0001
+2	+000 0010	0000 0010	0000 0010	0000 0010
…				
+126	+111 1110	0111 1110	0111 1110	0111 1110
+127	+111 1111	0111 1111	0111 1111	0111 1111
-0	-000 0000	1000 0000	1111 1111	0000 0000
-1	-000 0001	1000 0001	1111 1110	1111 1111
-2	-000 0010	1000 0010	1111 1101	1111 1110
…				
-126	-111 1110	1111 1110	1000 0001	1000 0010
-127	-111 1111	1111 1111	1000 0000	1000 0001
-128	-1000 0000	×	×	1000 0000

解：

a. $[47]_{补}=0010\ 1111B$，$[69]_{补}=0100\ 0101B$。

$$[47+69]_{补}=[47]_{补}+[69]_{补}$$
$$=0010\ 1111B+0111\ 0101B$$
$$=0111\ 0100B。$$

$[116]_{补}=0111\ 0100B$。

a**式成立。**

b. $[47]_{补}=0010\ 1111B$，$[-89]_{补}=1010\ 0111B$。

$$[47-89]_{补}=[47]_{补}+[-89]_{补}$$
$$=0010\ 1111B+1010\ 0111B$$
$$=1101\ 0110B。$$

$[-42]_{补}=1101\ 0110B$。

b**式成立。**

c. $[-87]_{补}=1010\ 1001B$，$[39]_{补}=0010\ 0111B$

$$[-87+39]_{补}=[-87]_{补}+[39]_{补}$$
$$=1010\ 1001B+0010\ 0111B$$
$$=1101\ 0000B。$$

$[-48]_{补}=1101\ 0000B$。

c**式成立。**

例 21 验证$[[-54]_{补}]_{补}=[-54]_{原}$。

$[-54]_{原}=1011\ 0110B$，

$[-54]_{补}=1100\ 1010B$,

$[-54]_{补}]_{补}=1011\ 0110B$。

等式成立。

1.1.7 定点数和浮点数

机器数在运算过程中，按照小数点在数字中的位置是否发生移动，可把机器数分为定点数和浮点数。

1. 定点数和浮点数

以一个能显示 8 位数字的计算器为例说明定点数和浮点数的区别。若其最高位显示 +、- 号，则有效数字为七位。如图 1.1.2 所示。

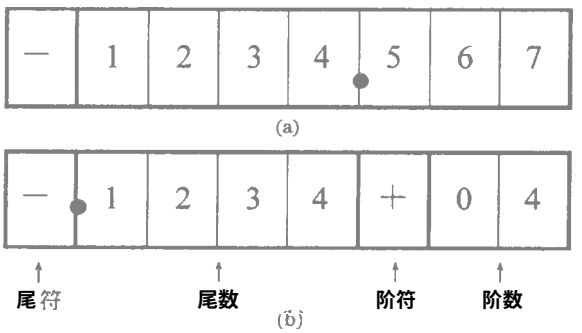


图 1.1.2 一个 8 位数字的计算器显示屏

如前所述，机器数在机器中表示时小数点的位置是事先设定的。若将小数点设在图 1.1.2(a) 中的 4 和 5 之间，那么，它可以表示四位整数和三位小数 -1234.567。计算器所能表示的数的范围在 -9999.999 ~ +9999.999 之间。由于小数点的位置在计算器计算过程中保持不变，这样的数称为定点数。

另一种数的表示方法是科学计数法。例如 -1234.567 用科学记数法可表示为 -0.1234567×10^{4}。其中 0.1234567 称为尾数，10 称为阶底，10 的方次 +4 称为阶码。对应图 1.1.2(b) 的显示格式，所能显示的数值范围在 -0.999×10^{-99} ~ $+0.999\times10^{+99}$ 之间。这样，在总位数不变并保持三位有效数字的前提下，扩大了计数的范围。虽然上述的尾数的小数点位置并不改变，但阶码变化，小数点在数中的实际位置随之改变。因小数点在运算过程中是浮动的，这样的机器数称为浮点数。

运算过程中阶码固定的数为定点数，阶码变化的数为浮点数。

二进制数也可以用科学记数法表示，与十进制的区别只是它以 2 为底。例如：

$+1011011B=0.1011011B\times2^{6D}=0.1011011B\times2^{110B}$。

$-0.00101B=-0.101B\times2^{-2D}=-0.101\times2^{-10B}$。

2. 机器数的浮点表示

用 12 位表示一个浮点机器数的格式如图 1.1.3 所示。其中阶码（包括阶符和阶数）占用高四位。尾符和尾数占用低 8 位。阶符和尾符各占一位，0 表示正，1 表示负。阶数和尾数分别占用三位和七位。

浮点数中的阶数为整数，尾数为小数，即小数点位于整数位 1 和小数位 0.5 之间，如

图 1.1.3 所示。浮点数所表示的数值范围主要由阶码的位数决定，若阶码的位数为 n，其表示数值的范围为：$-(2^n-1)\sim(2^n-1)$。有效数字的位数由尾数（含尾符）的位数决定，若尾数的位数为 m，则有效数字的位数为：INT(mlog2)或 INT(mlog2)+1。对图 1.1.3 所示的浮点数，它所表示数的范围是：$-(2^8-1)\sim(2^8-1)$。其尾数的有效数字为 3 位。若一个机器数由 6 个字节组成，其阶码占 8 位，尾数占 40 位，它所表示的数值范围为：$2^{-128}\sim 2^{127}$，即 $2.939\times10^{-39}\sim1.701\times10^{38}$。其有效数字有 11～12 位。

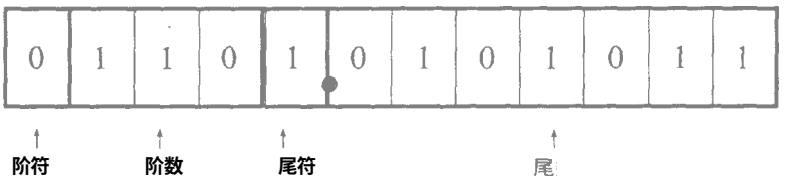


图 1.1.3　一个 12 位浮点数的格式

思考 根据你所学的数学知识，说明为什么 n 位阶码它所表示的数值范围为 $-(2^n-1)\sim(2^n-1)$？

例 22 按图 1.1.3 的格式写出以下四个浮点数的存储映像，其中阶码和尾数均使用补码。

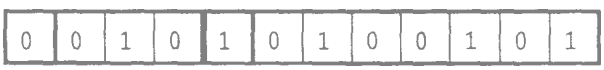
a. $X_1=14.625$，　b. $X_2=-22.75$，　c. $X_3=0.34375$，　d. $X_4=-0.09375$。

a. $X_1=14.625D=1110.101B=0.111\ 0101B\times2^4$，

$[X_1]_{补}=0.111\ 0101B\times2^{0100B}$。其存储映象为：

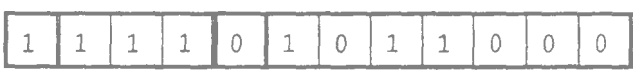
b. $X_2=-22.75D=-10110.11B=-0.101\ 1011B\times2^2$

$[X_2]_{补}=1.010\ 0101B\times2^{0010B}$。其存储映象为：



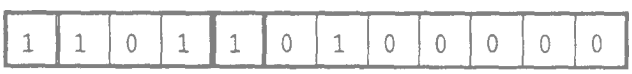
c. $X_3=0.34375D=0.01011B=0.101\ 1000B\times2^{-1}$

$[X_3]_{补}=0.101\ 1000B\times2^{1111B}$。其存储映象为：



d. $X_4=-0.09375D=-0.00011D=-0.110\ 0000B\times2^{-3}$

$[X_4]_{补}=1.010\ 0000B\times2^{1101B}$。其存储映象为：



1.2 编 码

用以交换信息的字符、符号、图形或声音只有以二进制的形式表示才能在计算机中存储和处理，而输出时还必须转换成原有的形式才能被人们识别。为此，我们把按照某种规则用 0 和 1 组成的二进制代码去表示字符、符号或图形的过程，称为编码。下面介绍计算机中常用的数字、字符和汉字的编码。

1.2.1 8421—BCD 码

1.8421—BCD 码的编码规则

数在计算机中的存储和运算都以二进制的形式进行，而输入、输出时人们却常用十进制。因此，需要一种以十进制输入、输出而在机内以二进制形式存储和运算的数字编码。由前节可知，用四位二进制数可以表示十进制的 0～9，BCD(Binary Code Decimal)码或称二－十进制码就是常用的一种编码。表 1.2.1 列出了 BCD 码的编码规则。

表 1.2.1 8421－BCD 码的编码表

BCD 码	0000	0001	0010	0011	0100	0101	0110	0111
十进制	0	1	2	3	4	5	6	7
BCD 码	1000	1001	1010	1011	1100	1101	1110	1111
十进制	8	9	非 法 编 码					

从编码表可以看出，四位二进制代码共十六个，其中前十个分别对应十进制的 0～9，且每一个 BCD 码的二进制数值等于对应的十进制数的值。而 1010～1111 没有与之对应的十进制数，称为非法编码。BCD 码运算按十进制进位，其表达形式和运算方法都与十进制相似，只是以二进制的形式表示，故称为二－十进制编码，BCD 的含义就在于此。在这种编码中，二进制数各位的权分别为 8、4、2、1，因此，这种编码又称为 8421－BCD 码。

用 BCD 码表示十进制数十分方便。例如：

$X_1 = 147D$，　$[X_1]_{BCD} = 0001\ 0100\ 0111$。

$X_2 = -853D$，　$[X_2]_{BCD} = -1000\ 0101\ 0011$。

$X_3 = 0.962D$　　$[X_3]_{BCD} = 0000.1001\ 0110\ 0010$。

应当注意区分十进制数的 BCD 码和与之对应的二进制数的不同。二进制数按位表示十进制数的值的，而 BCD 码是四位二进制数作为一个整体表示一个十进制数的。例如，147D 对应的二进制数为 1001 0011B，而 BCD 码 $[1001\ 0011]_{BCD}$ 表示的十进制数却是 93D。

除了 8421－BCD 码外，2421 码、5421 和余三码也都是 BCD 码。

2.BCD 码的运算

BCD 码的加减运算与十进制相同，即按位运算。不同的是十进制的一位对应 BCD 码却是四位。

例 1：用 BCD 码计算验证 14＋3＝17。

十进制	BCD 码
14	0001 0100
+ 3	+ 0011
17	0001 0111

$17D = [0001\ 0111]_{BCD}$。

例 2：用 BCD 码计算验证 59 − 25 = 34。

十进制	BCD 码
59	0101 1001
− 25	− 0010 0101
34	0011 0100

$34D = [0011\ 0100]_{BCD}$。

例 3：用 BCD 码计算验证 73 − 36 = 37。

十进制

73

− 36

37

第二位被借位变为 0110

BCD 码

0111 0011 ← 0011 不够减，向高位借位。

− 0011 0110

↓ ↓

0110 1101 ← 第一位加 10D(1010B)变为 = 1101

− 0011 0110

0011 0111

$37D = [0011\ 0111]_{BCD}$。

3. 十进制调整

BCD 码运算时按十进制的位加减和进位或借位，而每一个 BCD 码的加减还是按二进制规则运算，如果两个一位的 BCD 码的运算结果大于 9 时，就会出现非法编码。例如，计算 5 + 7，BCD 码相加的结果是(0101 + 0111 =)1100，于是出现计算错误。

BCD 码运算错误出现在：(1) 二数之和大于 9；(2) 二数相加本位和虽不大于 9，但已产生了进位这两种情况。为此，在 BCD 码的运算中当上述情况发生时，采用加 6 调整的方法，即进行十进制调整以避免产生错误。

例 4 二数之和大于 9 时作加 6 调整。请用 BCD 码计算 4 + 8 = ？

十进制	BCD 码	
4	0100	
+ 8	+ 1000	
12	1100	←非法编码
	+ 0001 0110	←加 6 调整
	0001 0010	

$12D = [0001\ 0010]_{BCD}$。

可以看出，BCD 码相加和大于 9 应进位而没有进位，结果出现非法编码，对这个结果作加 6(0110B)调整，即可获得正确的结果。

例 5 二数相加本位和不大于 9，但产生进位。请用 BCD 码计算 28 + 9 = ？