$$D = \sum k_i 16^i \tag{1.2.5}$$

并由此式计算出它所表示的十进制数值。例如

$$(2A.7F)_{16} = 2 \times 16^1 + 10 \times 16^0 + 7 \times 16^{-1} + 15 \times 16^{-2}$$
$$= (42.4960937)_{10}$$

式中的下脚注 16 表示括号里的数是十六进制数，有时也用 H(Hexadecimal)代替这个脚注。

由于目前在微型计算机中普遍采用 8 位、16 位和 32 位二进制并行运算，而 8 位、16 位和 32 位的二进制数可以用 2 位、4 位和 8 位的十六进制数表示，因而用十六进制符号书写程序十分简便。

表 1.2.1 是十进制数 0～15 与等值二进制、八进制、十六进制数的对照表。

表 1.2.1 不同进制数的对照表

十进制(Decimal)	二进制(Binary)	八进制(Octal)	十六进制(Hexadecimal)
00	**0000**	00	0
01	**0001**	01	1
02	**0010**	02	2
03	**0011**	03	3
04	**0100**	04	4
05	**0101**	05	5
06	**0110**	06	6
07	**0111**	07	7
08	**1000**	10	8
09	**1001**	11	9
10	**1010**	12	A
11	**1011**	13	B
12	**1100**	14	C
13	**1101**	15	D
14	**1110**	16	E
15	**1111**	17	F

复习思考题

R1.2.1 写出 4 位二进制数、4 位八进制数和 4 位十六进制数的最大数。

R1.2.2 与 4 位二进制数、4 位八进制数、4 位十六进制数的最大值等值的十进制数各为多少？

1.3 不同数制间的转换

一、二－十转换

将二进制数转换为等值的十进制数称为二－十转换。转换时只要将二进制数按式(1.2.3)展开，然后将所有各项的数值按十进制数相加，就可以得到等值的十进制数了。例如

$$\begin{aligned}(\mathbf{1011.01})_2 &= \mathbf{1}\times 2^3+\mathbf{0}\times 2^2+\mathbf{1}\times 2^1+\mathbf{1}\times 2^0+\mathbf{0}\times 2^{-1}+\mathbf{1}\times 2^{-2}\\ &=(11.25)_{10}\end{aligned}$$

二、十－二转换

所谓十－二转换，就是将十进制数转换为等值的二进制数。

首先讨论整数的转换。

假定十进制整数为$(S)_{10}$，等值的二进制数为$(k_nk_{n-1}\cdots k_0)_2$，则依式(1.2.3)可知

$$\begin{aligned}(S)_{10} &= k_n2^n+k_{n-1}2^{n-1}+\cdots+k_12^1+k_02^0\\ &=2(k_n2^{n-1}+k_{n-1}2^{n-2}+\cdots+k_1)+k_0 \qquad (1.3.1)\end{aligned}$$

上式表明，若将$(S)_{10}$除以2，则得到的商为$k_n2^{n-1}+k_{n-1}2^{n-2}+\cdots+k_1$，而余数即$k_0$。

同理，可将式(1.3.1)除以2得到的商写成

$$k_n2^{n-1}+k_{n-1}2^{n-2}+\cdots+k_1=2(k_n2^{n-2}+k_{n-1}2^{n-3}+\cdots+k_2)+k_1 \qquad (1.3.2)$$

由式(1.3.2)不难看出，若将$(S)_{10}$除以2所得的商再次除以2，则所得余数即k_1。

依此类推，反复将每次得到的商再除以2，就可求得二进制数的每一位了。

例如，将$(173)_{10}$化为二进制数可如下进行

2	173	余数 $=\mathbf{1}=k_0$
2	86	余数 $=\mathbf{0}=k_1$
2	43	余数 $=\mathbf{1}=k_2$
2	21	余数 $=\mathbf{1}=k_3$
2	10	余数 $=\mathbf{0}=k_4$
2	5	余数 $=\mathbf{1}=k_5$
2	2	余数 $=\mathbf{0}=k_6$
2	1	余数 $=\mathbf{1}=k_7$
	0	

故 $(173)_{10}=(\mathbf{10101101})_2$。

其次讨论小数的转换。

若$(S)_{10}$是一个十进制的小数,对应的二进制小数为$(0.k_{-1}k_{-2}\cdots k_{-m})_2$,则据式(1.2.3)可知

$$(S)_{10}=k_{-1}2^{-1}+k_{-2}2^{-2}+\cdots+k_{-m}2^{-m}$$

将上式两边同乘以2得到

$$2(S)_{10}=k_{-1}+(k_{-2}2^{-1}+k_{-3}2^{-2}+\cdots+k_{-m}2^{-m+1}) \tag{1.3.3}$$

式(1.3.3)说明,将小数$(S)_{10}$乘以2所得乘积的整数部分即k_{-1}。

同理,将乘积的小数部分再乘以2又可得到

$$2(k_{-2}2^{-1}+k_{-3}2^{-2}+\cdots+k_{-m}2^{-m+1})=k_{-2}+(k_{-3}2^{-1}+\cdots+k_{-m}2^{-m+2}) \tag{1.3.4}$$

亦即乘积的整数部分就是k_{-2}。

依此类推,将每次乘2后所得乘积的小数部分再乘以2,便可求出二进制小数的每一位了。

例如,将$(0.8125)_{10}$化为二进制小数时可如下进行

0.8125
× 2
1.6250 ……整数部分 = **1** = k_{-1}

0.6250
× 2
1.2500 ……整数部分 = **1** = k_{-2}

0.2500
× 2
0.5000 ……整数部分 = **0** = k_{-3}

0.5000
× 2
1.0000 ……整数部分 = **1** = k_{-4}

故$(0.8125)_{10}=(\mathbf{0.1101})_2$。

三、二-十六转换

将二进制数转换为等值的十六进制数称为二-十六转换。

由于4位二进制数恰好有16个状态,而把这4位二进制数看作一个整体时,它的进位输出又正好是逢十六进一,所以只要从低位到高位将整数部分每4位二进制数分为一组并代之以等值的十六进制数,同时从高位到低位将小数部

分的每4位数分为一组并代之以等值的十六进制数,即可得到对应的十六进制数。

例如,将$(01011110.10110010)_2$化为十六进制数时可得

$$\begin{array}{cccccc} (0101 & 1110. & 1011 & 0010)_2 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ =(\ 5 & E. & B & 2\ )_{16} \end{array}$$

四、十六-二转换

十六-二转换是指将十六进制数转换为等值的二进制数。转换时只需将十六进制数的每一位用等值的4位二进制数代替就行了。

例如,将$(8FA.C6)_{16}$化为二进制数时得到

$$\begin{array}{cccccc} (\ 8 & F & A. & C & 6\ )_{16} \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ =(1000 & 1111 & 1010. & 1100 & 0110)_2 \end{array}$$

五、八进制数与二进制数的转换

将二进制数转换为八进制数的二-八转换和将八进制数转换为二进制数的八-二转换,在方法上与二-十六转换和十六-二转换的方法基本相同。

在将二进制数转换为八进制数时,只要将二进制数的整数部分从低位到高位每3位分为一组并代之以等值的八进制数,同时将小数部分从高位到低位每3位分为一组并代之以等值的八进制数就可以了。

例如,若将$(011110.010111)_2$化为八进制数,则得到

$$\begin{array}{cccc} (011 & 110. & 010 & 111)_2 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ (\ 3 & 6. & 2 & 7\ )_8 \end{array}$$

反之,若将八进制数转换为二进制数,则只要将八进制数的每一位代之以等值的二进制数即可。例如,将$(52.43)_8$转换为二进制数时,得到

$$\begin{array}{cccc} (5 & 2. & 4 & 3)_8 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ (101 & 010. & 100 & 011)_2 \end{array}$$

六、十六进制数与十进制数的转换

在将十六进制数转换为十进制数时,可根据式(1.2.5)将各位按权展开后相加求得。在将十进制数转换为十六进制数时,可以先转换为二进制数,然后再将得到的二进制数转换为等值的十六进制数。这两种转换方法上面已经讲过了。

复习思考题

R1.3.1 在十－二转换中，整数部分的转换方法和小数部分的转换方法有何不同？

R1.3.2 怎样将八进制数转换为十六进制数和将十六进制数转换为八进制数？

R1.3.3 怎样才能将十进制数转换为八进制数？

1.4 二进制算数运算

1.4.1 二进制算数运算的特点

当两个二进制数码表示两个数量大小时，它们之间可以进行数值运算，这种运算称为算术运算。二进制算术运算和十进制算术运算的规则基本相同，唯一的区别在于二进制数是“逢二进一”而不是十进制数的“逢十进一”。

例如，两个二进制数 **1001** 和 **0101** 的算术运算有

加法运算

```
  1001
+ 0101
------
  1110
```

减法运算

```
  1001
- 0101
------
  0100
```

乘法运算

```
  1001
× 0101
------
  1001
 0000
1001
0000
-------
0101101
```

除法运算

```
        1.11…
     ------
0101 ) 1001
       0101
       ----
       1000
       0101
       ----
       0110
       0101
       ----
       0010
```

从上面的例子中可以看到二进制算数运算的两个特点，即二进制数的乘法运算可以通过若干次的“被乘数（或零）左移 1 位”和“被乘数（或零）与部分积相加”这两种操作完成；而二进制数的除法运算能通过若干次的“除数右移 1 位”和“从被除数或余数中减去除数”这两种操作完成。

如果我们再能设法将减法操作转化为某种形式的加法操作,那么加、减、乘、除运算就全部可以用“移位”和“相加”两种操作实现了。利用上述特点能使运算电路的结构大为简化。这也是数字电路中普遍采用二进制算数运算的重要原因之一。

1.4.2 反码、补码和补码运算

我们已经知道,在数字电路中是用逻辑电路输出的高、低电平表示二进制数的 **1** 和 **0** 的。那么数的正、负又如何表示呢?通常采用的方法是在二进制数的前面增加一位符号位。符号位为 **0** 表示这个数是正数,符号位为 **1** 表示这个数是负数。这种形式的数称为原码。

在做减法运算时,如果两个数是用原码表示的,则首先需要比较两数绝对值的大小,然后以绝对值大的一个作为被减数、绝对值小的一个作为减数,求出差值,并以绝对值大的一个数的符号作为差值的符号。不难看出,这个操作过程比较麻烦,而且需要使用数值比较电路和减法运算电路。如果能用两数的补码相加代替上述的减法运算,那么计算过程中就无需使用数值比较电路和减法运算电路了,从而使运算器的电路结构大为简化。

为了说明补码运算的原理,我们先来讨论一个生活中常见的事例。例如,你在 5 点钟的时候发现自己的手表停在 10 点上了,因而必须把表针拨回到 5 点。由图 1.4.1 可以看出,这时有两种拨法:第一种拨法是往回拨 5 格,10 − 5 = 5,拨回到了 5 点;另一种拨法是往前拨 7 格,10 + 7 = 17。由于表盘的最大数只有 12,超过 12 以后的“进位”将自动消失,于是就只剩下减去 12 以后的余数了,即 17 − 12 = 5,也将表针拨回到了 5 点。这个例子说明,10 − 5 的减法运算可以用 10 + 7的加法运算代替。因为 5 和 7 相加正好等于产生进位的模数 12,所以我们称 7 为 −5 对模 12 的补数,也称为补码(Complement)。

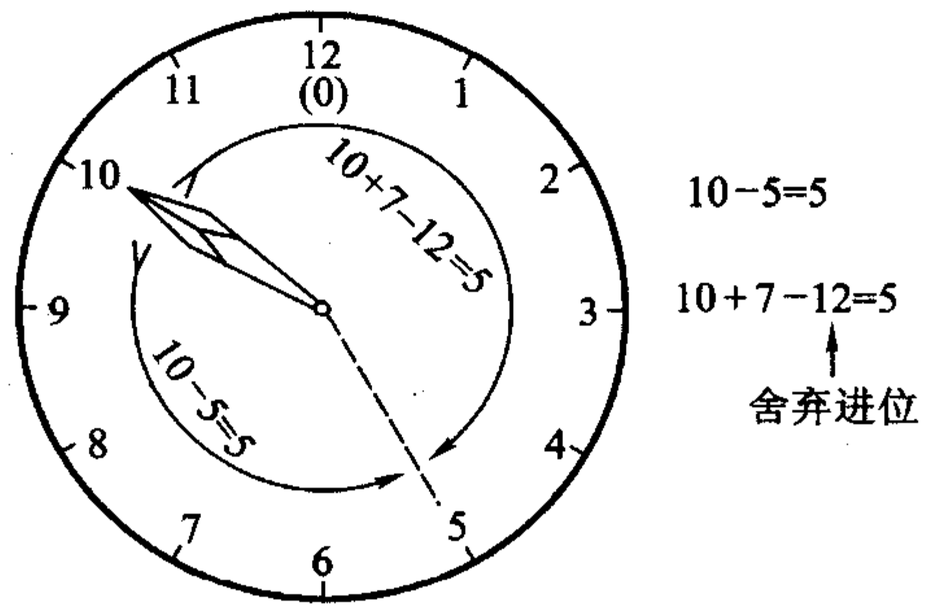

图 1.4.1 说明补码运算原理的例子

从这个例子中可以得出一个结论，就是在舍弃进位的条件下，减去某个数可以用加上它的补码来代替。这个结论同样适用于二进制数的运算。

图 1.4.2 给出了 4 位二进制数补码运算的一个例子。由图可见，**1011 − 0111 = 0100** 的减法运算，在舍弃进位的条件下，可以用 **1011 + 1001 = 0100** 的加法运算代替。因为 4 位二进制数的进位基数是 16（**10000**），所以 **1001**(9)恰好是 **0111**(7)对模 16 的补码。

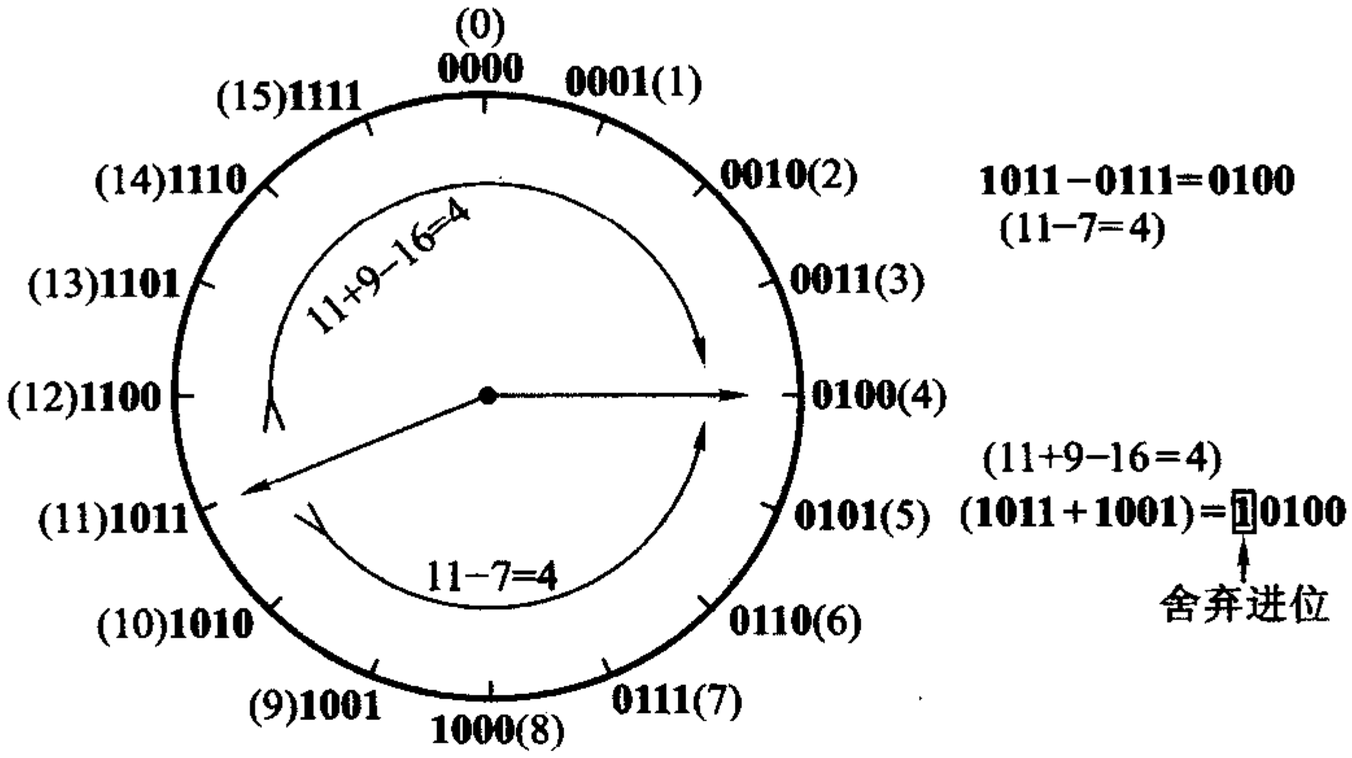

图 1.4.2　4 位二进制数补码运算的例子

基于上述原理，对于有效数字（不包括符号位）为 n 位的二进制数 N，它的补码 $(N)_{\text{COMP}}$ 表示方法为

$$(N)_{\text{COMP}} = \begin{cases} N & (\text{当 } N \text{ 为正数}) \\ 2^n - N & (\text{当 } N \text{ 为负数}) \end{cases} \tag{1.4.1}$$

即正数（当符号位为 **0** 时）的补码与原码相同，负数（当符号位为 **1** 时）的补码等于 $2^n - N$。符号位保持不变。

在一些国外的教材中，也将式(1.4.1)定义的补码称为“2 的补码”(2's Complement)。

为了避免在求补码的过程中做减法运算，通常是先求出 N 的反码 $(N)_{\text{INV}}$，然后在负数的反码上加 1 而得到补码。二进制 N 的反码 $(N)_{\text{INV}}$ 是这样定义的

$$(N)_{\text{INV}} = \begin{cases} N & (\text{当 } N \text{ 为正数}) \\ (2^n - 1) - N & (\text{当 } N \text{ 为负数}) \end{cases} \tag{1.4.2}$$

由上式可知，当 N 为负数时，$N + (N)_{\text{INV}} = 2^n - 1$，而 $2^n - 1$ 是 n 位全为 **1** 的二进制数，所以只要将 N 中每一位的 **1** 改为 **0**、**0** 改为 **1**，就得到了 $(N)_{\text{INV}}$。以后我们将会看到，将二进制数的每一位求反，在电路上是很容易实现的。国外的有些教材中又将式(1.4.2)定义的反码称为“1 的补码”(1's Complement)。

由式(1.4.2)又可得到,当 N 为负数时,$(N)_{INV}+1=2^n-N$,而由式(1.4.1)又知,当 N 为负数时,$(N)_{COMP}=2^n-N$,由此得到

$$(N)_{COMP}=(N)_{INV}+1 \tag{1.4.3}$$

即二进制负数的补码等于它的反码加 1。

【例 1.4.1】 写出带符号位二进制数 **00011010**(+26)、**10011010**(-26)、**00101101**(+45)和 **10101101**(-45)的反码和补码。

解: 根据式(1.4.2)和式(1.4.3)得到

原 码	反 码	补 码
00011010	**00011010**	**00011010**
10011010	**11100101**	**11100110**
00101101	**00101101**	**00101101**
10101101	**11010010**	**11010011**

表 1.4.1 是带符号位的 3 位二进制数原码、反码和补码的对照表。其中规定用 **1000** 作为 -8 的补码,而不用来表示 -0。

下面再来讨论两个用补码表示的二进制数相加时,和的符号位如何得到。为此,我们在例 1.4.2 中列举出了两数相加时的四种情况。

表 1.4.1 原码、反码、补码对照表

十进制数	二进制数		
	原码(带符号数)	反 码	补 码
+7	**0111**	**0111**	**0111**
+6	**0110**	**0110**	**0110**
+5	**0101**	**0101**	**0101**
+4	**0100**	**0100**	**0100**
+3	**0011**	**0011**	**0011**
+2	**0010**	**0010**	**0010**
+1	**0001**	**0001**	**0001**
+0	**0000**	**0000**	**0000**
-1	**1001**	**1110**	**1111**
-2	**1010**	**1101**	**1110**
-3	**1011**	**1100**	**1101**
-4	**1100**	**1011**	**1100**
-5	**1101**	**1010**	**1011**
-6	**1110**	**1001**	**1010**
-7	**1111**	**1000**	**1001**
-8	**1000**	**1111**	**1000**

【例 1.4.2】　用二进制补码运算求出 13 + 10、13 − 10、− 13 + 10 和 − 13 − 10。

解：　由于 13 + 10 和 − 13 − 10 的绝对值为 23，所以必须用有效数字为 5 位的二进制数才能表示，再加上一位符号位，就得到 6 位的二进制补码。

由式(1.4.1)和式(1.4.3)可知，+13 的二进制补码应为 **001101**(最高位为符号位)，−13 的二进制补码为 **110011**，+10 的二进制补码为 **001010**，−10 的二进制补码为 **110110**。计算结果分别为

+13	**0 01101**		+13		**0 01101**
+10	**0 01010**		−10		**1 10110**
+23	**0 10111**		+3	(1)	**0 00011**
−13	**1 10011**		−13		**1 10011**
+10	**0 01010**		−10		**1 10110**
−3	**1 11101**		−23	(1)	**1 01001**

从上面的例子中可以看出，若将两个加数的符号位和来自最高有效数字位的进位相加，得到的结果(舍弃产生的进位)就是和的符号。这个道理仍然可以用图 1.4.2 所示的图形加以说明。

需要强调指出，在两个同符号数相加时，它们的绝对值之和不可超过有效数字位所能表示的最大值，否则会得出错误的计算结果。

复习思考题

R1.4.1　二进制正、负数的原码、反码和补码三者之间是什么关系？

R1.4.2　为什么两个二进制数的补码相加时，和的符号位等于两数的符号位与来自最高有效数字位的进位相加的结果(舍弃产生的进位)？

R1.4.3　如何求二进制数补码对应的原码？

1.5　几种常用的编码

一、十进制代码

为了用二进制代码表示十进制数的 0 ~ 9 这十个状态，二进制代码至少应当有 4 位。4 位二进制代码一共有十六个(**0000 ~ 1111**)，取其中哪十个以及如何

与0~9相对应,有许多种方案。表1.5.1中列出了常见的几种十进制代码,它们的编码规则各不相同。

表1.5.1 几种常见的十进制代码

十进制数 \ 编码种类	8421码(BCD代码)	余3码	2421码	5211码	余3循环码
0	0000	0011	0000	0000	0010
1	0001	0100	0001	0001	0110
2	0010	0101	0010	0100	0111
3	0011	0110	0011	0101	0101
4	0100	0111	0100	0111	0100
5	0101	1000	1011	1000	1100
6	0110	1001	1100	1001	1101
7	0111	1010	1101	1100	1111
8	1000	1011	1110	1101	1110
9	1001	1100	1111	1111	1010
权	8421		2421	5211	

8421码又称BCD(Binary Coded Decimal)码,是十进制代码中最常用的一种。在这种编码方式中,每一位二值代码的1都代表一个固定数值,将每一位的1代表的十进制数加起来,得到的结果就是它所代表的十进制数码。由于代码中从左到右每一位的1分别表示8、4、2、1,所以将这种代码称为8421码。每一位的1代表的十进制数称为这一位的权。8421码中每一位的权是固定不变的,它属于恒权代码。

余3码的编码规则与8421码不同,如果把每一个余3码看作4位二进制数,则它的数值要比它所表示的十进制数码多3,故而将这种代码称为余3码。

如果将两个余3码相加,所得的和将比十进制数和所对应的二进制数多6。因此,在用余3码做十进制加法运算时,若两数之和为10,正好等于二进制数的16,于是便从高位自动产生进位信号。

此外,从表1.5.1中还可以看出,0和9、1和8、2和7、3和6、4和5的余3码互为反码,这对于求取对10的补码是很方便的。

余3码不是恒权代码。如果试图将每个代码视为二进制数,并使它等效的十进制数与所表示的代码相等,那么代码中每一位的1所代表的十进制数在各个代码中不能是固定的。

2421码是一种恒权代码,它的0和9、1和8、2和7、3和6、4和5也互为反码,这个特点和余3码相仿。

5211码是另一种恒权代码。待学了第六章中计数器的分频作用后可以发