HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY PRESS

数字逻辑（第二版）

SHUZI LUOJI

主编　毛法尧

编者　毛法尧　欧阳星明　任宏萍

华中理工大学出版社

数 字 逻 辑

（第二版）

主 编 毛法尧
编 者 毛法尧 欧阳星明 任宏萍

华中理工大学出版社

内 容 简 介

科学技术的迅猛发展和教学改革的不断深入，促使了本教材的及时更新。本书在第一版的基础上，从结构到内容都作了较大的调整、修改和增删，从而更加完善和满足教学的需要。

本书系统地介绍了逻辑设计的基本理论和设计方法，主要讨论如何用中、小规模集成电路进行逻辑设计的问题，对于大规模集成电路在逻辑设计中的应用以及常用的逻辑器件也作了适当的介绍，最后介绍了数字系统的一般设计方法及计算机辅助逻辑设计这一新的设计技术。

本书可作为高等学校计算机及应用、计算机软件、自控、通信等专业的教材，也可供有关工程技术人员参考。

数 字 逻 辑

（第二版）

主编　毛法尧

编者　毛法尧　欧阳星明　任宏萍

责任编辑　　黄以铭　唐元瑜

*

华中理工大学出版社出版发行

（武昌喻家山）

新华书店湖北发行所经销

华中理工大学出版社汚阳印刷厂印刷

*

开本：787×1092　1/16　印张：19　字数：459 000

1992 年 11 月第 2 版　1996 年 1 月第 10 次印刷

印数：48 001—58 000

ISBN 7—5609—0730—X/TP・75

定价：14.80 元

（鄂）新登字第 10 号

前　言

本书是1986年出版的《数字逻辑》教材的第二版。

根据高等学校工科计算机专业和高等学校(工科、综合大学)计算机软件专业“数字逻辑”课程教学大纲的要求，并考虑自控、通信等专业学习“数字逻辑”课程的需要，编者结合多年教学实践修订了该书。

数字逻辑是计算机的基础理论之一。熟练地掌握逻辑设计的理论和方法，对于从事计算机研制和应用的广大科技工作者来说是十分必要的。由于技术的发展和教学内容及教学方法的更新，本书第二版作了大幅度的修订。对原书第一至七章从结构到内容都作了较大的调整、修改和增删，并增写了第八、九章。另外，全书的图形符号和文字符号均采用了国家标准。

全书共分九章。第一章数制与码制，介绍数字系统中常用的数制及转换、码制和编码。第二章逻辑代数基础，介绍逻辑代数、逻辑函数及函数化简。第三章组合逻辑电路，着重介绍组合逻辑电路的分析和设计方法。第四章同步时序逻辑电路，介绍同步时序逻辑电路的一般分析和设计方法。第五章异步时序逻辑电路，讨论脉冲异步时序逻辑电路和电平异步时序逻辑电路的分析和设计。第六章采用中、大规模集成电路的逻辑设计，介绍用中、大规模集成电路进行逻辑设计的特点和一般方法。第七章数字系统设计，仅对数字系统逻辑设计的方法和步骤进行探讨。第八章计算机辅助逻辑设计，简单介绍计算机辅助逻辑设计这一先进设计技术。第九章逻辑器件，介绍目前广泛使用的几种器件的结构和特点。

本书在内容和叙述上力求主次分明、详略适当、叙述清晰、由浅入深。

本书由毛法尧主编，第一、七、八、九章由毛法尧编写，第二、三、四章由欧阳星明编写，第五、六章由任宏萍编写。在编写过程中得到了关心本书再版的兄弟院校同行的热忱关怀，华中理工大学出版社为本书的再版给予了大力的支持。在此一并表示诚挚的谢意。

由于编者水平有限，书中难免有欠妥或错误之处，恳请读者批评指正。

编　者

1991年10月于华中理工大学

目　录

第一章　数制与码制

第二章　逻辑代数基础

第三章　组合逻辑电路

第四章　同步时序逻辑电路

第五章　异步时序逻辑电路

第六章　采用中、大规模集成电路的逻辑设计

第七章　数字系统设计

第八章　计算机辅助逻辑设计

第九章　逻辑器件

第一章　数制与码制

计算技术的发展促进了科学技术和生产的飞跃发展。现在，计算机已广泛地应用于科学计算、数据处理和生产过程控制等领域中。它是数字系统中最常见的、最有代表性的一种设备。

数字系统的特点是它所处理的信息都是离散元素。这些离散元素可以是十进制数字、某种字母、各种算符及标点符号等。各种离散元素按不同方式排列可以表示大量的信息。例如，字母C，O，M，P，U，T，E和R就可以形成单词COMPUTER，数字1984表示一个确定的数，等等。

在数字系统中，信息的离散元素是以称为信号的物理量来表示的，电压和电流就是最常用的电信号。通常，数字系统的信号都只有两个离散量，因此，称为二进制信号。由于只有导通和截止两种状态的电子器件十分可靠，再加上人类的逻辑思维方式也倾向于二值，因此，数字系统常采用二进制信号。

信号的离散量有的是自然形成的，有的则是将连续过程有意加以量化后而得到的。例如，一份学生成绩单就是天然的离散过程，上面有学生的姓名、课程名称、成绩等。又如，科学工作者在科学研究工作中常常要观察连续的过程，但仅将一些特殊的数值记录下来列成表格，这样就将连续的信息量化了，所以表格中的每一个数字都已成为离散量。

数字系统处理的是离散元素，而这些离散元素通常以二进制数的形式出现，如十进制数字或字母符号都用二进制代码来表示。为此，必须讨论数的代码特征及运算。

本章介绍数制和码制的概念、表示方法、性质及相互转换，以便为进一步学习以后各章打下基础。

1.1　进位计数制

1.1.1　十进制数的表示

在日常生活中，人们通常采用十进制数来计数，它的基数为10，每位数可用下列十个数码之一来表示，即0，1，2，3，4，5，6，7，8，9。

十进制数的计数规律是“逢十进一”，也就是说，每位数累计不能超过10，计满10就应向高位进1。

当我们看到一个十进制数，如632.45时，我们就会立刻想到：这个数的最左位为百位(6代表600)，第二位为十位(3代表30)，第三位为个位(2代表2)，小数点右面第一位为十分位(4代表4/10)，第二位为百分位(5代表5/100)。这里百、十、个、十分之一和百分之一都是10的次幂，它取决于系数所在的位置，称之为“权”。十进制数632.45从左至右各位的权分别是10^2，10^1，10^0，10^{-1}，10^{-2}。这样，632.45按权展开的形式如下：

$$632.45=6\times10^2+3\times10^1+2\times10^0+4\times10^{-1}+5\times10^{-2}$$

等式左边的表示方法称之为位置记数法，等式右边则是其按权展开式。

一般说来，对于任意一个十进制数 N，可用位置记数法表示如下：

$$(N)_{10}=(a_{n-1}a_{n-2}\cdots a_1a_0.a_{-1}a_{-2}\cdots a_{-m})_{10}$$

也可用按权展开式表示如下：

$$\begin{aligned}(N)_{10}&=a_{n-1}\times10^{n-1}+a_{n-2}\times10^{n-2}+\cdots+a_1\times10^1+a_0\times10^0\\&\quad+a_{-1}\times10^{-1}+a_{-2}\times10^{-2}+\cdots+a_{-m}\times10^{-m}\\&=\sum_{i=-m}^{n-1}a_i\times10^i\end{aligned}$$

式中，a_i 为 0～9 这 10 个数码中的任意一个；n 为整数部分的位数；m 为小数部分的位数。

通常，对十进制数的表示，可以在数字的右下角标注 10 或 D。

1.1.2 二进制数的表示

数字系统中使用的进位制并不限于十进制，当进位基数为 2 时，称为二进制。在二进制中，只有 0 和 1 两个数码。二进制的计数规则是“逢二进一”，即每位计满 2 就向高位进一。例如，$(1101)_2$ 就是一个二进制数，不同数位的数码表示的值不同，各位的权值是以 2 为底的连续整数幂，从右向左递增。

对于任意一个二进制数 N，用位置计数法表示为

$$(N)_2=(a_{n-1}a_{n-2}\cdots a_1a_0.a_{-1}a_{-2}\cdots a_{-m})_2$$

用按权展开式表示为

$$\begin{aligned}(N)_2&=a_{n-1}\times2^{n-1}+a_{n-2}\times2^{n-2}+\cdots+a_1\times2^1+a_0\times2^0\\&\quad+a_{-1}\times2^{-1}+a_{-2}\times2^{-2}+\cdots+a_{-m}\times2^{-m}\\&=\sum_{i=-m}^{n-1}a_i\times2^i\end{aligned}$$

式中，a_i 为数码 0 或 1；n 为整数部分的位数；m 为小数部分的位数。

通常，对二进制数的表示，可以在数字右下角标注 2 或 B。

1.1.3 任意进制数的表示

由于二进制数简单、容易实现，所以它是数字系统中广泛采用的一种数制。但用二进制表示一个数时，所用的位数比用十进制数表示的位数多，人们读写很不方便，因此，常采用八进制或十六进制。

八进制数的基数是 8，采用的数码是 0，1，2，3，4，5，6，7。计数规则是“逢八进一”，相邻两位高位的权值是低位权值的 8 倍。例如，数$(47.6)_8$就表示一个八进制数。由于八进制的数码和十进制前 8 个数码相同，所以为了便于区分，通常在数字的右下角标注 8 或 O。

十六进制数的基数为 16，采用的数码是 0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F。其中，A，B，C，D，E，F 分别代表十进制数字 10，11，12，13，14，15。十六进制的计数规则是“逢十六进一”，相邻高位的权值是低位权值的 16 倍。例如，数$(51AF.8B)_{16}$就是一个十六进制数。通常，在数字的右下角标注 16 或 H。

与二进制数一样，任意一个八进制数和十六进制数均可用位置计数法的形式和按权展开式的形式表示。

一般说来，对于任意的数 N，都能表示成以 r 为基数的 r 进制数，数 N 的表示方法也有

两种形式，即

位置记数法：

$$(N)_r=(a_{n-1}a_{n-2}\cdots a_1a_0.a_{-1}a_{-2}\cdots a_{-m})_r$$

按权展开式：

$$\begin{aligned}(N)_r&=a_{n-1}\times r^{n-1}+a_{n-2}\times r^{n-2}+\cdots+a_1r^1+a_0r^0\\&\quad+a_{-1}\times r^{-1}+a_{-2}\times r^{-2}+\cdots+a_{-m}\times r^{-m}\\&=\sum_{i=-m}^{n-1}a_ir^i\end{aligned}$$

式中，a_i 为 0～$r-1$ 数码中的任意一个；r 为进位制的基数；n 为整数部分的位数；m 为小数部分的位数。

r 进制的计数规则是“逢 r 进一”。

不同数制的各种数码见表 1.1，该表列出了当 r 为 10，2，8，16 时，各种进位计数制中开头的 16 个自然数。

表 1.1　不同进位计数制的各种数码

十进制数 ($r=10$)	二进制数 ($r=2$)	八进制数 ($r=8$)	十六进制数 ($r=16$)
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	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

1.1.4　二进制的特点

选择什么样的进位计数制来表示数，对数字系统的成本和性能影响很大。在数字系统中，常用二进制来表示数字和进行运算。这是由于它具有如下特点：

(1) 二进制数只有 0 和 1 两个数码，任何具有两个不同稳定状态的元件都可用来表示 1 位二进制数，例如晶体管的导通和截止，脉冲信号的有和无等。

(2) 二进制运算规则简单，其运算规则是：

加法规则

0+0=0　　0+1=1

1+0=1　　1+1=0（同时向相邻高位进 1）

减法规则

0−0=0　　0−1=1（同时向相邻高位借 1）

1−0=1　　1−1=0

乘法规则

0×0=0　　0×1=0

1×0=0　　1×1=1

除法规则

0÷1=0　　1÷1=1

下面举几个二进制数运算的例子。

例 1.1　进行 1101+1011 运算。

解

$$\begin{array}{r} 1101 \\ +)\ 1011 \\ \hline 11000 \end{array}$$

两个二进制数的加法运算和十进制数的加法运算相似，但采用“逢二进一”的法则，每位数累计到 2 时，本位就记为 0，且向相邻高位进 1。

例 1.2　进行 11101−10011 运算。

解

$$\begin{array}{r} 11101 \\ -)\ 10011 \\ \hline 1010 \end{array}$$

二进制减法采用“借一当二”的法则，减法运算从低位起按位进行，在遇到 0 减 1 时，就要向相邻高位借 1，也就是从那一位减去 1。

例 1.3　进行 1101×1011 运算。

解

$$\begin{array}{r} 1101 \\ \times)\ 1011 \\ \hline 1101 \\ 1101 \\ 0000 \\ 1101 \\ \hline 10001111 \end{array}$$

二进制数的乘法运算和十进制数的乘法运算相似，所不同的是对部分积进行累加时要按“逢二进一”的原则。

例 1.4　进行 10010001÷1011 运算。

解

$$\begin{array}{r|l} & 1101\cdots\text{商} \\ 1011 & 10010001 \\ & \underline{1011} \\ & 1110 \\ & \underline{1011} \\ & 1101 \\ & \underline{1011} \\ & 10\cdots\text{余数} \end{array}$$

二进制数的除法运算同十进制数的除法运算类似，但采用二进制数的运算规则。

(3) 二进制数的数码 0 和 1，可与逻辑代数中逻辑变量的“假”和“真”对应起来。也就是说，可用一个逻辑变量来表示一个二进制数码。这样，在逻辑运算中可以使用逻辑代数这一数学工具。

二进制数的缺点是书写长，不便记忆和阅读。因此，人们常采用八进制和十六进制数，这两种数制容易书写和阅读，便于记忆，且具有二进制数的特点，十分容易将它们转换成二进制数。

1.2 数制转换

在计算机和其它数字系统中，普遍使用二进制数，而人们习惯于使用十进制数，所以，在信息处理中首先必须把十进制数转换成计算机能加工和处理的二进制数，然后再将二进制数的计算结果转换成人们习惯的十进制数。这里就存在一个不同数制的相互转换问题。

1.2.1 二进制数和十进制数的转换

二进制数转换成十进制数是很方便的，只要将二进制数写成按权展开式，并将式中各乘积项的积算出来，然后各项相加，即可得到与该二进制数相对应的十进制数。例如

$$\begin{aligned}(11010.101)_2 &= 1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0\\ &\quad +1\times2^{-1}+0\times2^{-2}+1\times2^{-3}\\ &=16+8+2+0.5+0.125\\ &=(26.625)_{10}\end{aligned}$$

十进制数转换成二进制数时，需将待转换的数分成整数部分和小数部分，并分别加以转换。当一个十进制数写成

$$(N)_{10}=\langle\text{整数部分}\rangle_{10}.\langle\text{小数部分}\rangle_{10}$$

转换时，首先将 $\langle$整数部分$\rangle_{10}$转换成 $\langle$整数部分$\rangle_2$，然后再将 $\langle$小数部分$\rangle_{10}$转换成 $\langle$小数部分$\rangle_2$。待整数部分和小数部分确定后，就可写成

$$(N)_2=\langle\text{整数部分}\rangle_2.\langle\text{小数部分}\rangle_2$$

一、整数转换

十进制数的整数部分采用“除 2 取余”法进行转换，即把十进制整数除以 2，取出余数 1 或 0 作为相应二进制数的最低位，把得到的商再除以 2，再取余数 1 或 0 作为二进制数的次低位，依次类推，继续上述过程，直至商为 0，所得余数为最高位。

例如，要将十进制整数 58 转换为二进制整数，就要把它写成如下形式：

$$\begin{aligned}(58)_{10}&=(a_{n-1}a_{n-2}\cdots a_1a_0)_{10}\\ &=a_{n-1}\times2^{n-1}+a_{n-2}\times2^{n-2}+\cdots+a_1\times2^1+a_0\times2^0\\ &=2(a_{n-1}\times2^{n-2}+a_{n-2}\times2^{n-3}+\cdots+a_1)+a_0\end{aligned}$$

只要求出等式中的各个系数 a_{n-1}，a_{n-2}，…，a_1，a_0，便得到二进制数。

将上式两边除以 2，得

$$(29)_{10}=a_{n-1}\times2^{n-2}+a_{n-2}\times2^{n-3}+\cdots+a_1+\frac{a_0}{2}$$

两数相等，整数部分和小数部分必须对应相等，等式左边余数为0，则取 a_0 为0。因而得

$$(29)_{10}=2\ (a_{n-1}\times 2^{n-3}+a_{n-2}\times 2^{n-4}+\cdots+a_2)\ +a_1$$

将等式两边再除以2，得

$$\left(14+\frac{1}{2}\right)_{10}=a_{n-1}\times 2^{n-3}+a_{n-2}\times 2^{n-4}+\cdots+a_2+\frac{a_1}{2}$$

比较等式两边，等式左边余数为1，则取 a_1 为1。

依次类推，可得系数 a_2，a_3，…，a_{n-2}，a_{n-1}。

根据上面讨论的方法，可用下列形式很方便地将十进制整数转换成二进制数。

```
2 |5 8
2 |2 9        余数 0 (a0) 最低位
2 |1 4        余数 1 (a1)
2 | 7         余数 0 (a2)
2 | 3         余数 1 (a3)
2 | 1         余数 1 (a4)
  0           余数 1 (a5) 最高位
```

因此，$(58)_{10}=(111010)_2$。

二、纯小数转换

十进制数的小数部分采用“乘2取整”法进行转换，即先将十进制小数乘以2，取其整数1或0，作为二进制小数的最高位；然后将乘积的小数部分再乘以2，并再取整数，作为次高位。重复上述过程，直到小数部分为0或达到所要求的精度。

例如，要将十进制小数0.625转换为二进制小数，需把它写成如下形式：

$$\begin{aligned}(0.625)_{10}&=(0.a_{-1}a_{-2}\cdots a_{-m})_2\\&=a_{-1}\times 2^{-1}+a_{-2}\times 2^{-2}+\cdots+a_{-m}\times 2^{-m}\\&=\frac{a_{-1}}{2}+\frac{1}{2}(a_{-2}\times 2^{-1}+\cdots+a_{-m}\times 2^{-m+1})\end{aligned}$$

只要求出各系数 a_{-1}，a_{-2}，…，a_{-m}，便得到二进制小数。

将上式两边乘2，得

$$(1.25)_{10}=a_{-1}+(a_{-2}\times 2^{-1}+\cdots+a_{-m}\times 2^{-m+1})$$

根据两个数相等，必须是整数部分和小数部分分别相等的道理，a_{-1}必须等于左边的整数，则 a_{-1}为1。

等式右边括号内的数仍为小数，因而

$$(0.25)_{10}=\frac{a_{-2}}{2}+\frac{1}{2}(a_{-3}\times 2^{-1}+\cdots+a_{-m}\times 2^{-m+2})$$

再将等式两边乘2，得

$$(0.5)_{10}=a_{-2}+(a_{-3}\times 2^{-1}+\cdots+a_{-m}\times 2^{-m+2})$$

比较等式两边的整数，又取 a_{-2}为0。如此连续乘2，直到小数部分等于0，即可求得系数 a_{-1}，a_{-2}，…，a_{-m}。

根据上面讨论的方法，可用下列形式很方便地将十进制小数转换成二进制数。

$$
\begin{array}{rl}
0.625 & \\
\times)\quad 2 & \\
\hline
\boxed{1}.250 & \text{整数 1 } (a_{-1}) \text{ 最高小数位} \\
\times)\quad 2 & \\
\hline
\boxed{0}.500 & \text{整数 0 } (a_{-2}) \\
\times)\quad 2 & \\
\hline
\boxed{1}.000 & \text{整数 1 } (a_{-3}) \text{ 最低小数位}
\end{array}
$$

因此，$(0.625)_{10}=(0.101)_2$。

必须指出：式中的整数不参加连乘。

在十进制的小数部分转换中，有时连续乘 2 不一定能使小数部分等于 0，这说明该十进制小数不能用有限位二进制小数表示。这时，只要取足够多的位数，使其误差达到所要求的精度就可以了。

例 1.5 将十进制数 0.18 转换成二进制数，精确到小数点后 4 位。

解

$$
\begin{array}{rl}
0.18 & \\
\times)\quad 2 & \\
\hline
\boxed{0}.36 & \text{整数 } 0(a_{-1}) \\
\times)\quad 2 & \\
\hline
\boxed{0}.72 & \text{整数 } 0(a_{-2}) \\
\times)\quad 2 & \\
\hline
\boxed{1}.44 & \text{整数 } 1(a_{-3}) \\
\times)\quad 2 & \\
\hline
\boxed{0}.88 & \text{整数 } 0(a_{-4}) \\
\times)\quad 2 & \\
\hline
\boxed{1}.76 & \text{整数 } 1(a_{-5})
\end{array}
$$

十进制数 0.18 连续四次乘 2 后，其小数部分等于 0.88，仍不为 0。由于要求精确到小数点后 4 位，因此将 0.88 再乘一次 2，小数点后第 5 位四舍五入后得

$$(0.18)_{10}\approx(0.0011)_2$$

如果一个十进制数既有整数部分又有小数部分，转换时，整数部分采用“除 2 取余”法，小数部分采用“乘 2 取整”法，然后再把转换的结果合并起来。

例 1.6 将$(58.625)_{10}$转换成二进制数。

解 $(58.625)_{10}=(58)_{10}+(0.625)_{10}$

$=(111010)_2+(0.101)_2$

$=(111010.101)_2$

1.2.2 八进制数、十六进制数与二进制数的转换

八进制数的基数是 8($8=2^3$)，十六进制数的基数为 16($16=2^4$)。二进制数、八进制数和十六进制数之间具有 2 的整指数倍的关系，因而可直接进行转换。

将二进制数转换成八进制或十六进制数的方法为：从小数点开始，分别向左、右按 3 位(转换成八进制)或 4 位(转换成十六进制)分组，最后不满 3 位或 4 位的，则需加 0。将每组以

对应的八进制数或十六进制数代替，即为等值的八进制数和十六进制数。例如

八进制	2		5		7	.	0		5	5	1
二进制	010		101		111	.	000		101	101	100
十六进制		A		F		.		1		6	C

则 $(257.0554)_8=(10101111.0001011011)_2=(AF.16C)_{16}$

将八进制数或十六进制数转换成二进制数时，可按上述方法的逆过程进行。

1.3 带符号数的代码表示

1.3.1 真值与机器数

前面讨论的数都没有考虑符号，一般认为是正数，但在算术运算中总会出现负数。不带符号的数是数的绝对值，在绝对值前加上表示正负的符号就成了带符号数。一个带符号的数由两部分组成：一部分表示数的符号，另一部分表示数的数值。数的符号是一个具有正、负两种值的离散信息，它可以用一位二进制数来表示。习惯上以0表示正数，而以1表示负数。对于一个 n 位二进制数，如果数的第一位为符号位，则剩下的 $n-1$ 位就表示数的数值部分。一般，直接用正号“+”和负号“−”来表示符号的二进制数，叫做符号数的真值。数的真值形式是一种原始形式，不能直接用于数字计算机中。但是，当使符号数值化以后，就可在计算机中使用它。计算机中使用的符号数叫做机器数。

例如，二进制正数+0.1011在机器中的表示如图1.1（a）所示，二进制负数−0.1011在机器中的表示如图1.1（b）所示。

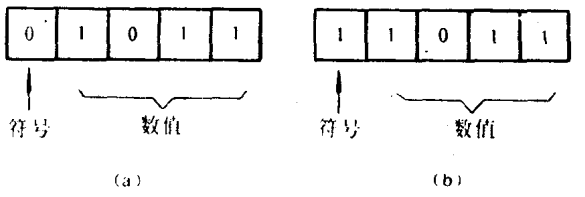

图1.1 二进制数在机器中的表示

由前面介绍的二进制数的加、减、乘、除四种运算可知，乘法运算实际上是做移位加法运算，而除法运算则是做移位减法运算。这就是说，在机器中只需要做加、减两种运算。但做减法运算时，必须先比较两个数绝对值的大小，将绝对值大的数减绝对值小的数，最后在相减结果的前面加上正确的符号。虽然逻辑电路可以实现减法运算，但所需的电路复杂，运算时间较长。为了能使减法运算变成加法运算，人们提出了三种机器数的表示形式，即原码、反码和补码。

1.3.2 原码

原码又称为“符号-数值表示”。在以原码形式表示的正数和负数中，第1位表示符号位，对于正数，符号位记作0，对于负数，符号位记作1，其余各位表示数值部分。

假如两个带符号的二进制数分别为 N_1 和 N_2，其真值形式为

$$N_1=+10011 \qquad N_2=-01010$$

则 N_1 和 N_2 的原码表示形式为

$$[N_1]_{原}=010011 \qquad [N_2]_{原}=101010$$

根据上述原码形成规则，一个 n 位的整数 N（包括一位符号位）的原码一般表示式为

$$[N]_{原}=\begin{cases} N & 0\leqslant N<2^{n-1} \\ 2^{n-1}-N & -2^{n-1}<N\leqslant 0 \end{cases}$$

对于定点小数，通常小数点定在最高位的左边，这时，数值小于 1。定点小数原码一般表示式为

$$[N]_{原}=\begin{cases} N & 0\leqslant N<1 \\ 1-N & -1<N\leqslant 0 \end{cases}$$

从原码的一般表示式中可以看出：

(1) 当 N 为正数时，$[N]_{原}$ 和 N 的区别只是增加一位用 0 表示的符号位。由于在数的左边增加一位 0 对该数的数值并无影响，所以 $[N]_{原}$ 就是 N 本身。

(2) 当 N 为负数时，$[N]_{原}$ 和 N 的区别是增加一位用 1 表示的符号位。

(3) 在原码表示中，有两种不同形式的 0，即

$$[+0]_{原}=0.00\cdots0$$
$$[-0]_{原}=1.00\cdots0$$

1.3.3 反码

反码又称为“对 1 的补数”。用反码表示时，左边第 1 位也为符号位，符号位为 0 代表正数，符号位为 1 代表负数。对于负数，反码的数值是将原码数值按位求反，即原码的某位为 1，反码的相应位就为 0，或者原码的某位为 0，反码的相应位就为 1。而对于正数，反码和原码相同。所以，反码数值的形成与它的符号位有关。

假如两个带符号的二进制数分别为 N_1 和 N_2，其真值形式为

$$N_1=+10011 \qquad N_2=-01010$$

则 N_1 和 N_2 的反码表示形式为

$$[N_1]_{反}=010011 \qquad [N_2]_{反}=110101$$

根据上述的反码形成规则，一个 n 位的整数 N（包括一位符号位）的反码一般表示式为

$$[N]_{反}=\begin{cases} N & 0\leqslant N<2^{n-1} \\ (2^n-1)+N & -2^{n-1}<N\leqslant 0 \end{cases}$$

同样，对于定点小数，若小数部分的位数为 m，则它的反码一般表示式为

$$[N]_{反}=\begin{cases} N & 0\leqslant N<1 \\ (2-2^{-m})+N & -1<N\leqslant 0 \end{cases}$$

从反码的一般表示式可以看出：

(1) 正数 N 的反码 $[N]_{反}$ 与原码 $[N]_{原}$ 相同。

(2) 对于负数 N，其反码 $[N]_{反}$ 的符号位为 1，数值部分是将原码数值按位变反。

(3) 在反码表示中，0 的表示有两种不同的形式，即

$$[+0]_{反}=0.00\cdots0$$
$$[-0]_{反}=1.11\cdots1$$

1.3.4 补码

补码又称为“对2的补数”。在补码表示法中，正数的表示同原码和反码的表示是一样的，而负数的表示却不同。对于负数，从原码转换到补码的规则是：符号位不变，仍为1，数值部分变反加1，即按位变反，最低位加1。

假如两个带符号的二进制数分别为 N_1 和 N_2，其真值形式为

$$N_1=+10011 \qquad N_2=-01010$$

则 N_1 和 N_2 的补码表示形式为

$$[N_1]_{补}=010011 \qquad [N_2]_{补}=110110$$

根据上述补码形成规则，一个 n 位的整数 N（包括一位符号位）的补码一般表示式为

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

同样，对于定点小数，补码一般表示式可写成

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

由补码的一般表示式可以看出：

(1) 正数 N 的补码 $[N]_{补}$、反码 $[N]_{反}$ 和原码 $[N]_{原}$ 是相同的。

(2) 对于负数，补码 $[N]_{补}$ 的符号位为1，其数值部分为反码数值加1。

(3) 在补码表示法中，0的表示形式是唯一的，即

$$[+0]_{补}=0.00\cdots0$$

$$[-0]_{补}=0.00\cdots0$$

1.3.5 机器数的加、减运算

上面介绍了带符号数的三种表示法，它们的形成规则不同，加、减运算的规律也不相同。下面分别介绍。

一、原码运算

原码中的符号位仅用来表示数的正、负，不参加运算，进行运算的只是数值部分。原码运算时，应首先比较两个数的符号，若两数的符号相同，则就将两个数的数值相加，最后给结果附上相应的符号；若两数的符号不同，于是就得进一步比较两数的数值相对大小，然后将数值较大的数减去数值较小的数，并将数值较大的数的符号作为最后结果的符号。下面举例说明。

例1.7 已知 $N_1=-0.0011$，$N_2=0.1011$，求 $[N_1+N_2]_{原}$ 和 $[N_1-N_2]_{原}$。

解 $[N_1+N_2]_{原}=[(-0.0011)+0.1011]_{原}$

由于 N_1 和 N_2 的符号不同，并且 N_2 的绝对值大于 N_1 的绝对值，因此要进行 N_2 减 N_1 的运算，其结果为正。

$$\begin{array}{r} 0.1011 \\ -)\ 0.0011 \\ \hline 0.1000 \end{array}$$

运算结果为原码，即

$$[N_1+N_2]_{原}=0.1000$$

故其真值为