第 1 章　数字逻辑基础

学习要求：

- 熟悉模拟量与数字量及其表示法，了解数字技术的优点和缺点
- 熟练掌握数制及其相互的转换，掌握数字系统中数与编码的表示和运算
- 熟练掌握逻辑代数的相关概念、基本运算、基本公理、基本定理及基本规则
- 熟练掌握逻辑函数的基本表达式、标准形式和化简方法
- 熟悉数字逻辑门电路、晶体管开关特性与集成电路类型及其使用特性

1.1　数字技术的相关概念

在人们的日常生活中，“数字”这一术语的使用已相当普遍，数字技术已得到了广泛的应用。本课程将学习数字技术的基本概念、基本原理和基本方法，熟悉从简单开关电路到复杂计算机的所有数字系统。通过本课程的学习，读者将会深入了解数字系统是如何工作的，并能够把所学到的知识应用于数字系统的分析、设计以及故障检查及排除之中。本章首先介绍一些数字技术的基本概念，然后介绍数制与编码、逻辑代数基础，最后介绍数字逻辑门电路、晶体管的开关特性与集成电路类型及其使用特性。

1.1.1　物理量的表示

在人类的活动中，常常将物理学中度量物体属性或描述物体运动状态及其变化过程的量，如电压、湿度、水位等称为物理量。在各行、各业大多数场合，也经常会涉及数量的概念，数量也称为数值，即数的大小。在大多数物理系统中，物理量以数量的形式被观察、记录、变换与运算，或以其他形式被利用。在处理不同的物理量时，有效且准确地表示其数量十分重要。表示其数量有两种方法，即模拟表示法和数字表示法。

1．模拟表示法

模拟表示法是用电压、电流或与所反映的数量成比例的表头的移动来表示数量的。例如老式电压表，其指针的偏转与电压成正比，指针偏转角度反映了电压的大小，当电压增加或减小时，电压表的指针随之偏转。电压可以取 0～400V 之间的任意值，例如，220V。也就是说，电压表指针的偏转度代表了电压的值。**由此可见，模拟量所具有的重要特征是：其数值可在一定的范围内连续变化。**

2．数字表示法

在数字表示法中，数量是用数字符号而不是用可连续变化的指示仪表来表示的。例如数字式钟表，它用代表小时、分，还有秒的十进制数来反映一天的时间。一天的时间变化是连续的，但数字钟表的读数是不连续的，而是每次以一分或一秒的步长变化的。所以，与连续变化的模拟式壁挂钟的表盘读数所反映的时间相比较，数字表示法是用离散步长的

方法表示其一天时间的变化的。**由此可见，数字量所具有的重要特征是：其数值在一定的范围内是离散变化的。**

3．模拟量与数字量的主要区别

模拟量<=>连续 ；数字量<=>离散（步进）

由于模拟表示具有连续性，因此当阅读模拟量时，数值常有解释的余地；相反，由于数字表示具有离散性，因此当阅读数字量的数值时，不存在模棱两可的情况。实际上，当对模拟量进行测量时，通常会截断到一个适当的精度，即把量值数字化，也就是说，数字表示法是将一个可连续变化的量赋值等于一个有限精度数字的结果。

例如，当用温度计测量体温时，水银柱的液面常常界于两条刻度线之间，而测量者总是基于与液面最近的一条刻度线给出一个数值，比如说 36.5℃。

1.1.2 数字系统与数字技术

数字系统是用来处理逻辑信息或以数字形式表示的物理量的器件组合，其数值仅能取离散值。这些器件可以是机械的、磁性的或气动的，但目前大多数是电子器件。最常见的数字系统包括计算机、手机、数字音像设备及通信网络系统等。

数字逻辑电路是数字系统的硬件部分。数字逻辑电路的特性是接收输入信号且产生与输入信号有确定关系的正确且稳定的输出信号。在数字系统中用到的许多技术被称为数字技术。

模拟系统所包含的装置能处理以模拟形式表示的物理量。例如，收音机中送到扬声器的输出信号的幅值可以取小于最大值之间的任意值。常见的模拟系统有老式带指针的万用电表、磁带式录放机、照明台灯的调光开关等。

1．数字技术的优点

在各项技术中，曾是用模拟方法实现的功能，如今越来越多地用数字技术替代。数字技术成功应用的主要原因是：

（1）信息存储方便。信息存储由特定器件构成的电路实现，该电路能在相对较小的物理空间上存储大量信息并可以长期保存。但模拟存储能力却相当有限。

（2）操作可编程。很容易设计一个由存储器指令程序控制其操作的数字系统。模拟系统也可被编程，但其操作的复杂性和可变性受到很大限制。

（3）抗干扰能力强。在数字系统中，电压的准确值并不重要，只要噪声信号不至于影响区分高低电平，则电压噪声的影响就可忽略不计。

（4）集成度更高。模拟电路受益于快速发展的 IC 工艺，但是模拟电路相对复杂，模拟系统无法达到与数字电路同样的集成度。相反，数字系统比较容易设计，因为数字系统所使用的电路是开关电路，开关电路中电压或电流的精度并不重要，重要的是其所处的范围。

（5）系统准确度及精度容易保持一致。数字化的信号在处理过程中不会降低精度。在模拟系统中，电压和电流信号会由于电路中元器件参数的改变或温度、湿度的影响产生失真。

2．数字技术的缺点

数字技术缺点比较少，最大的问题是：现实世界中感知的主要是模拟量，而人们习惯于使用数字量。处理数字化信号增加了系统的复杂性、费用和处理时间。

自然界中大多数物理量是模拟量，系统中被监测、处理和控制的输入、输出信号经常是

模拟量，如速度、电压和电流等。但是，大家习惯于用数字表示这些量。比如说，实际体温是 36.9℃，但是，读体温表时通常会说体温是 37℃。所以，人们常常是使用一个数字量来近似模拟量。

为了更好理解模拟量处理的复杂过程，举例如下。

一个典型的温度控制系统的框图如图 1.1 所示，所检测的是模拟温度，然后通过模/数转换器（ADC）把测量值转换为数字量，由数字电路处理数字量。数字量输出通过数/模转换器（DAC）变换为模拟量，再将模拟量输出到控制器，以便采取某种措施调节温度。所以，当处理模拟量输入、输出时，为了利用数字技术的优点，必须采取下述三个步骤：

①把模拟输入转换为数字形式；

②进行数字信号处理；

③将数字输出变换为模拟输出。

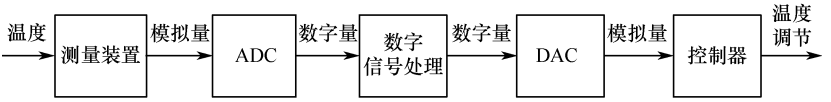

图 1.1　温度控制系统框图

因此，对于数字系统来说，由于信息必须在模拟形式与数字形式之间进行转换，从而增加了系统的复杂性和费用。并且，处理数字信号需要时间。事实上，所需要的数据越精确，处理过程花费的时间越长。但是，在大多数应用中，这些不足已被数字技术的优点所抵消。因此，模/数转换在当今技术领域已相当普遍。然而，在有些情况下，采用模拟技术则比较简单和经济。

在一个系统中经常会同时采用模拟技术与数字技术。为了充分利用各自的优点，在这种混合系统中，设计工作中最重要的是确定系统中哪一部分采用模拟形式，哪一部分采用数字形式。从大多数系统的发展趋势来看，随着信号在系统中的流动，应该在输入通道中尽可能早地使信号数字化，在输出通道中尽可能晚地把数字信号变换成模拟信号，这不失为一种好的方法。

3．世界将不断实现数字化

在日常生活中，人们每天的生活用品已逐渐从模拟形式变换为数字形式。在过去的几十年里，数字技术的发展速度十分惊人，并且有理由相信，未来的发展速度会更快。

计算机、多功能手机、数码相机和数字摄像机等，这些仅是数字化革命所带来的一小部分应用。数字领域将继续强劲增长，汽车可以配备个人电脑或使用云服务，电脑可以把你的仪表盘变为无线通信设备、导航仪等，你可以使用声音命令管理车辆、收发电子邮件、查询交通状况信息。在此时，你的手不用离开方向盘，或者你的视线也不用离开路面。

在不久的将来，借助移动技术、低轨卫星技术，可以通过耳机、手表与他人进行通信，也可以使用云系统将你刚刚观看 2 小时的电视节目内容在几秒钟内即刻传送并存储在你家里的存储器中，以便随时回放。你在网上可以实时观察到 1 万千米以外的地方，仍具有身临其境的感觉。这些所列举的内容也仅仅是沧海一粟。

总之，数字技术将持续高速进入人类的生活，开创一个前所未有的新天地。如今用以实现这些复杂数字系统的技术和工具已准备就绪。现在大家所能做的是：坚持不懈，努力学习，乐在其中。本书将介绍数字技术的有关概念和方法，以帮助读者逐步建立对数字技术的兴趣并能够很好地掌握它。

1.2 数制与编码

数制也称为计数制，是指用一组固定的符号和统一的规则来表示数值的方法。编码是指用代码来表示各组数据资料，使其成为可利用数字系统进行处理和分析的信息。代码是用来表示事物的记号，它可以用数字、字母、特殊的符号或它们之间的组合来表示。编码与代码之间有时也并不严格区分。

1.2.1 数制及其相互转换

数制是一种表示数值的方法，数制转换就是将一种数值表示的方法转换为另一种数值表示的方法。

1．进位计数制

按进位的方法进行计数，称为进位计数制。在各种进位计数制中，“十进制数”是人们最常用、最熟悉的，所以也常作为其他数制的参考数制。

十进制数中固定的符号为：0，1，2，3，4，5，6，7，8，9 和符号位符号“+”或“−”，以及小数点符号“.”，计数规则为“逢十进一”。其中十进制数的“十”是进位计数制的基数。

例如，人们常将十进制数 678.63 表示为：$678.63=6\times10^{2}+7\times10^{1}+8\times10^{0}+6\times10^{-1}+3\times10^{-2}$

上式中左边的形式是将若干符号并列在一起的方法，称为并列表示法，通常也称为十进制的位置计数法；右边的形式叫按权展开式，通常称为十进制数的多项式表示法。678.63 共有 5 位数字，第一位百位“6”代表 600，第二位为十位“7”代表 70，第三位为个位“8”代表 8，小数点右边的第一位为十分位“6”代表十分之六，小数点右边第二位为百分位“3”代表一百分之三。可见，数字符号在不同的位置代表着不同的数值，称之为不同的“权”。

任何一个十进制数，都可以用位置计数法表示为：$(A_{n-1}A_{n-2}\cdots A_1A_0\,.\,A_{-1}A_{-2}\cdots A_{-m})_{10}$

其中，n 表示整数位数，m 表示小数位数，A_i 是十进制中十个数字中的任何一个，即

$$0\leqslant A_i\leqslant 9,$$

括号外的下角标为进位计数制的基数，即“10”是基数标志，代表十进制数，在本书中十进制数的下角标可以省略。同样，任何一个十进制数也可用多项式表示法写为：

$$A_{n-1}(10)^{n-1}+A_{n-2}(10)^{n-2}+\cdots+A_1(10)^{1}+A_0(10)^{0}+A_{-1}(10)^{-1}+$$

$$A_{-2}(10)^{-2}+\cdots+A_{-m}(10)^{-m}=\sum_{i=-m}^{n-1}A_i\cdot 10^{i}$$

对其他进位计数制来说，也具有上述特点。如基数为 R，R 是一个十进制数，即以十进制数为参考数制的进位计数制，也一定有 R 个有序的数字符号：0，1，2，…，$R-1$ 和符号位符号“+”或“−”以及小数点符号“.”，计数规则为“逢 R 进一”。

对 R 进制中的数，可用位置计数法表示为：$(B_{n-1}B_{n-2}\cdots B_1B_0\,.\,B_{-1}B_{-2}\cdots B_{-m})_R$

也可用多项式表示法写为：

$$(B_{n-1}10^{n-1}+B_{n-2}10^{n-2}+\cdots+B_1 10^{1}+B_0 10^{0}+B_{-1}10^{-1}+B_{-2}10^{-2}+\cdots+B_{-m}10^{-m})_R$$

其中，n 表示整数位数，m 表示小数位数，“10”是 R 进制中的 1 和 0，B_i 是 R 进制中的数字符号之一，即

$$0\leqslant B_i\leqslant R-1$$

R 是计数制的基数，用十进制数表示。如 R=8，代表八进制数，R=10，代表十进制数。

几种进位计数制整数数列中的一部分见表 1.1。由表可知几种进位计数制数的对应等值关系。例如，$15 = (15)_{10} = (1111)_2 = (120)_3 = (33)_4 = (17)_8 = (F)_{16}$，即在不同的进位制中，同一个数之间表示的形式不同，但值是相同的，即等值的。

表 1.1　数制表（整数）

R=10	*R*=2	*R*=3	*R*=4	*R*=8	*R*=16
0	0	0	0	0	0
1	1	1	1	1	1
2	10	2	2	2	2
3	11	10	3	3	3
4	100	11	10	4	4
5	101	12	11	5	5
6	110	20	12	6	6
7	111	21	13	7	7
8	1 000	22	20	10	8
9	1 001	100	21	11	9
10	1 010	101	22	12	A
11	1 011	102	23	13	B
12	1 100	110	30	14	C
13	1 101	111	31	15	D
14	1 110	112	32	16	E
15	1 111	120	33	17	F
16	10 000	121	100	20	10
17	10 001	122	101	21	11
⋮	⋮	⋮	⋮	⋮	⋮

二进制数的权是 2 的幂，因此必须像熟悉 10 的幂那样熟悉 2 的幂。部分常用的二进制数的权见表 1.2。

表 1.2　二进制(R=2)各位的权(R^i)

i	R^i	*i*	R^i	*i*	R^i
−7	0.007 812 5	0	1	7	128
−6	0.015 625	1	2	8	256
−5	0.031 25	2	4	9	512
−4	0.062 5	3	8	10	1024
−3	0.125	4	16	11	2048
−2	0.25	5	32	12	4096
−1	0.5	6	64	13	8192

当基数 R=2 时，表示二进制数。由于二进制数中仅有两个数字符号 0 和 1，所以很容易用具有开关特性的电子元件来表示，因而被作为数字系统运算的基础。在二进制中，相应的运算规则有：

加法	乘法
0+0=0	0×0=0
0+1=1+0=1	0×1=1×0=0
1+1=10	1×1=1

这种运算的简便性带来了完成运算的电路及其控制方法的简单化。

2．进位计数制的相互转换

进位计数制的相互转换就是将一个数从一种进位计数制的表示法转换成另外一种进位计数制的表示法，记为 $N_\alpha \rightarrow N_\beta$，其中 N、α 和 β 均为十进制数，该式表示 N 在 α 进制中的表示转换为 N 在 β 进制中的表示。当然，这种转换一定是等值的，即 $N=N_\alpha=N_\beta$。进位计数制之间均可采用数学计算的方法来转换。通常用于数制转换的方法有两种：一种是多项式替代法，另一种是基数乘除法，这两种方法适用范围不同。

1）多项式替代法

基于计数制的计数本质，将基数为 α 进制的数转换成基数为 β 进制的数，转换一定是等值的，而且对于各项的运算并不会影响数值的大小。

可以将该过程描述如下：$N_\alpha \rightarrow N_\beta$。对于 α 进制中的 N，可表示为：

$$N_\alpha=(A_{n-1}\cdots A_1A_0.A_{-1}A_{-2}\cdots A_{-m})_\alpha$$
$$=(A_{n-1}\,10^{n-1}+\cdots+A_1\,10^1+A_0\,10^0+A_{-1}\,10^{-1}+A_{-2}\,10^{-2}+\cdots+A_{-m}\,10^{-m})_\alpha$$

将 α 进制下的 A_i、10 转换成 β 进制下的数，假设$(A_i)_\alpha=(B_i)_\beta$，$(10)_\alpha=(\gamma)_\beta$，则$(10)_\alpha=\alpha$，$\gamma=\alpha_\beta$

因此，$N_\beta=(B_{n-1}\gamma^{n-1}+\cdots+B_1\gamma^1+B_0\gamma^0+B_{-1}\gamma^{-1}+B_{-2}\gamma^{-2}+\cdots+B_{-m}\gamma^{-m})_\beta$

【例 1-1】 将$(2CE9)_{16}$转换为十进制数。

解：

$$(2CE9)_{16}=(2\times10^3+C\times10^2+E\times10^1+9\times10^0)_{16}$$
$$=(2\times16^3+12\times16^2+14\times16^1+9\times16^0)_{10}$$
$$=(11497)_{10}$$
$$=11497$$

在运算中，γ=16，十六进制中的 10 就是十进制中的 16。

其中的计算是按十进制计算规则进行的。

【例 1-2】 将$(211.2)_3$转换为二进制数。

解：

$$(211.2)_3=(2\times10^2+1\times10^1+1\times10^0+2\times10^{-1})_3$$
$$=(10\times11^2+1\times11^1+1\times11^0+10\times11^{-1})_2$$
$$=(10010+11+1+0.101010\cdots)_2$$
$$=(10110.101010\cdots)_2$$

在运算中，γ=11，即三进制中的 10 就是二进制中的 11。

其中的计算是按二进制计算规则进行的，如：

$$10\times11^{-1}=10\div11=0.101010\cdots$$

是循环小数，故$(211.2)_3=(10110.101010\cdots)_2$

由此可见，用多项式替代法实现 α 进制的数转换为 β 进制的数时，计算是在 β 进制中进行的，因此必须熟悉 β 进制的运算。也就是说，把其他进制的数转换为十进制数时，常采用多项式替代法。如要把十进制数转换为其他进制数时，可以采用基数乘除法。

2）基数乘除法

不同于多项式替代法，用基数乘除法将 α 进制的数转换成 β 进制的数时，计算是在 α

进制中进行的。整数的转换和小数的转换方法不同，整数转换要用基数除法，而小数的转换要用基数乘法。如果一个数包括整数和小数两部分，就可以将它们分别转换，然后合并起来。

（1）整数转换方法：基数除法

可以将整数转换过程描述如下：$N_\alpha \rightarrow N_\beta$。假设，对于 β 进制中的整数 N，可表示为：

$$N_\beta=(B_{n-1}\cdots B_1B_0)_\beta=(B_{n-1}\ 10^{n-1}+\cdots+B_1\ 10^1+B_0\ 10^0)_\beta$$

将 β 进制下的 B_i、10 转换成 α 进制下的数，假设 $(A_i)_\alpha=(B_i)_\beta$，$(\gamma)_\alpha=(10)_\beta$，则 $\gamma=\beta_\alpha$

因此，如果 $N_\alpha=(A_{n-1}\gamma^{n-1}+\cdots+A_1\gamma^1+A_0\gamma^0)_\alpha=((\cdots(A_{n-1}\gamma+A_{n-2})\gamma+\cdots)\gamma+A_1)\gamma+A_0$

将上式除以 γ，就可得 $(\cdots(A_{n-1}\gamma+A_{n-2})\gamma+\cdots)\gamma+A_1$ 和余数 A_0，如将所得的商再除以 γ，又可得到余数 A_1，重复以上过程，直至求得最后一个余数 A_{n-1}。然后将 α 进制中 $A_{n-1}\cdots A_1A_0$ 转换为 β 进制中的 $B_{n-1}\cdots B_1B_0$，这就是 N 在 β 进制中的表示。

【例 1-3】 将十进制的 3519 转换成十六进制数。

解：计算过程也可用下列算式表示：

16|3519　　余 15，个位数 A_0，要转换为十六进制中的数是 B_0，即 F

16|219　　余 11，十位数 A_1，要转换为十六进制中的数是 B_1，即 B

16|13　　余 13，百位数 A_2，要转换为十六进制中的数是 B_2，即 D

0

注意，这里 β=16，此时 γ=16，即十六进制中的 10 就是十进制中的 16。上述运算是在十进制中进行的，所以余数是十进制数，因此还要转换为十六进制中的数。

即 $(3519)_{10}=(DBF)_{16}$。

（2）小数转换方法：基数乘法

可以将小数转换过程描述如下：$N_\alpha \rightarrow N_\beta$。对于 β 进制中的小数 N，可表示为：

$$N_\beta=(0.B_{-1}\cdots B_{1-m}B_{-m})_\beta=(B_{-1}10^{-1}+\cdots+B_{1-m}10^{1-m}+B_{-m}10^{-m})_\beta$$

将 β 进制下的 B_i、10 转换成 α 进制下的数，假设 $(A_i)_\alpha=(B_i)_\beta$，$(\gamma)_\alpha=(10)_\beta$，则 $\gamma=\beta_\alpha$

因此，如果　　$N_\alpha=(A_{-1}\gamma^{-1}+\cdots+A_{1-m}\gamma^{1-m}+A_{-m}\gamma^{-m})_\alpha$

将上式乘以 γ，可得 $A_{-2}\gamma^{-1}+A_{-3}\gamma^{-2}+\cdots+A_{-m}\gamma^{1-m}$ 和整数 A_{-1}，如将所得小数再乘以 γ，又可得到整数 A_{-2}。重复以上过程，直至求得最后一个整数 A_{-m}。然后将 α 进制中 $A_{-1}\cdots A_{1-m}A_{-m}$ 转换为 β 进制中的 $B_{-1}\cdots B_{1-m}B_{-m}$，这就是 N 在 β 进制中的表示。

【例 1-4】 将 $(0.235)_{10}$ 转换成十六进制数。

解：这里 β=16，此时 γ=16，即十六进制中的 10 就是十进制中的 16。

0　.235

× 16

3　.760

× 16

12 .160

× 16

2　.560

上述运算是在十进制中进行的，所得整数是十进制中的数，因此还要转换为二进制中的数，即 $(0.235)_{10}\approx(0.3C2)_2$。

但是，小数的转换并不是都能将小数部分转换至零后结束。如果还未求到 A_{-m}，但是小数部分已为零，则表示此数已精确转换，余下位数均为零；如求到 A_{-m}，小数部分还不是零，则表明 N_β 是 N_α 的近似值，即转换有误差，需要考虑转换精度。

为了至少保持相同精度，可以使用下列方法确定。

假设 α 进制的小数有 i 位，转换成 β 进制后维持相同的精度至少需要 j 位，这时应有：

$$(0.1)^i_\alpha \geqslant (0.1)^j_\beta$$

在十进制中可表示为：

$$(1/\alpha)^i \geqslant (1/\beta)^j，即\ \beta^j \geqslant \alpha^i$$

两边取对数，得

$$\lg\beta^j \geqslant \lg\alpha^i，即\ j \cdot \lg\beta \geqslant i \cdot \lg\alpha，所以\ j \geqslant i \cdot \lg\alpha/\lg\beta$$

因为 j 是整数，所以 j 取大于等于 $i \cdot \dfrac{\lg\alpha}{\lg\beta}$ 的整数，并取最小值。

【例 1-5】 将（0.534 1）$_{10}$ 转换为十六进制时，小数位数应取多少？

解：由于原数精度为（0.1）$^4_{10}$，j 应满足下列不等式：$j \geqslant 3.320$，

所以，将（0.432 1）$_{10}$ 转换为十六进制时，小数位数应取 4 位。

3．任意两种进制之间的转换

现在已经知道：将 α 进制的数转换成 β 进制的数时，如果熟悉 α 进制的运算规则就采用基数乘除法；如果熟悉 β 进制的运算规则就采用多项式替代法。

如果对 α 进制和 β 进制的运算规则均不熟悉，则可利用十进制作为桥梁，即可先用多项式替代法将 α 进制的数转换成十进制的数，再用基数乘除法将十进制的数转换成 β 进制的数。这样，所有的计算均在十进制中进行。

4．直接转换法

在将 α 进制的数转换成 β 进制的数时，如果基数 α、β 都是 2^k（k 为正整数），则可以使用将二进制数作为桥梁的直接转换法。基数为 2^k 的进位计数制是将若干位二进制的字符串用一个数字字符来表示，这样使用就很方便。如十六进制的一个数字字符可以表示四位二进制字符串，而八进制的一个数字字符可以表示三位二进制字符串。

因此，基数为 2^k 的进位计数制的数与二进制数之间的转换，可以用划分相应字符串的方法直接转换。

如十六进制数转换为二进制数时，可将每位十六进制数字符号展开为相应的四位二进制字符串，再舍去多余的 0，即舍去整数部分最高位的 0 和小数部分最低位的 0 来求解。

当二进制数转换为十六进制数时，整数部分应从小数点开始向左数，每四位对应十六进制数的一位，最高有效位不足四位时，可在高位前加 0 补足四位。小数部分则应从小数点向右数，每四位对应十六进制数的一位，最低有效位不足四位时，应在低位后加 0 补足。

【例 1-6】 将（BE.17C）$_{16}$ 转换为八进制。

解：

十六进制：	B			E		.	1			7			C		
二进制：	0	1	0	1	1	1	1	1	0	.	0	0	0	1	0	1	1	1	1	0	0
八进制：	2			7			6			.	0			5			7			4	

即 $(BE.17C)_{16}=(276.0574)_8$ 或 $(BE.17C)_H=(276.0574)_O$。

式中下角标 H 和 O 分别表示十六进制和十进制数。

1.2.2　数的表示及其运算

1．数的小数点的表示

在数字系统中，小数是必定会出现的。在计算机中，表示小数点位置的方法通常有两种：一种是定点表示法；另一种是浮点表示法。定点表示法又分为定点小数表示法和定点整数表示法。一般使用定点小数表示法。

1）定点表示法

实际上，小数点在机器中并不存在，它只是人们约定的一个假想位置。所谓定点表示法，就是在计算机中数的小数点位置是固定的，一般固定在数的最高位之前或数的最低位之后。当小数点约定在数的最高位之前，而一般在符号位之后时，计算机表示的是绝对值小于 1 的纯小数。这种定点表示法称为**定点小数表示法**。当小数点约定在数的最低位之后时，计算机表示的是纯整数。这种定点表示法称为**定点整数表示法**。

通常情况下，实际处理的数可能既有整数部分，又有小数部分。这就需要先对数进行处理，选取一个合适的比例因子使它们变成纯整数或纯小数。这样之后，才能对它们直接进行运算。

【例 1-7】 假设某台计算机字长 8 位，规定最高位用来表示数的正、负号，其余 7 位表示数值。现有一组数：110.101，−10.0111 和 111.01，如何用定点表示法表示？

解：若用定点整数表示，则需选取比例因子 2^4，并用此比例因子与上述各数相乘，即得：

$$110.101\times2^4=1101010$$
$$-10.0111\times2^4=-0100111$$
$$111.01\times2^4=1110100$$

它们在机器中如果以原码的形式来表示，则可表示为：

0	1	1	0	1	0	1	0
1	0	1	0	0	1	1	1
0	1	1	1	0	1	0	0

若要用定点小数表示，则需提取比例因子 2^{-3}，并用此比例因子与上述各数相乘，即得：

$$110.101\times2^{-3}=0.110101$$
$$-10.0111\times2^{-3}=-0.0100111$$
$$111.01\times2^{-3}=0.11101$$

同样，它们在机器中如果以原码的形式来表示，则可表示为：

0	1	1	0	1	0	1	0
1	0	1	0	0	1	1	1
0	1	1	1	0	1	0	0

从上述表示中可以知道，这两种表示除选取的比例因子不一样外，其他都是一样的。对于选取相同比例因子的数，是可以直接进行计算的。

2）数的浮点表示法

所谓浮点表示法，就是计算机中数的小数点位置不是固定的，而是浮动的。因而，计算机必须能够表示小数点位置的浮动情况。

在数学中，数的表示有“记阶表示法”，也称为“科学表示法”。

例如，十进制数 N 可表示为：$N=10^J\times S$

其中，J 是一个正或负的整数，称为阶码；S 为一个正或负的小数，称为尾数。显然，只要有 J 和 S，即可表示出 N 的值。

例如，十进制数 3.548 可表示为：

$$3.548=10^{1}\times 0.354\ 8$$
$$=10^{2}\times 0.035\ 48$$

而 0.003 548 也可表示为：

$$0.003\ 548=10^{-1}\times 0.035\ 48$$
$$=10^{-2}\times 0.354\ 8$$

同理，二进制数也可用这种方法表示，仅是其中的基数“10”是十进制中的 2，即$(10)_2=(2)_{10}$。

例如，二进制数 101.11 和 10.111 可表示为：

$$101.11=(10)^{11}\times 0.10111$$
$$10.111=(10)^{10}\times 0.10111$$

注意，这里的阶码是一个二进制数。因此，这两个二进制数可用阶码和尾数表示为：

$$101.11\rightarrow 11，0.10111$$
$$10.111\rightarrow 10，0.10111$$

由此可见，这两个数的有效数字完全相同，仅小数点位置不同，其尾数完全相同，仅阶码不同。注意到，这两个数是不能使用一般的方法直接进行计算的，而是需要使用浮点运算方法。这种用阶码和尾数表示数的方法就是**数的浮点表示法**。

也就是说，需将一个字长划分为两部分，其中一部分表示阶码，另一部分表示尾数，同时将这两部分的最高位都定为符号位。例如，规定某 16 位字长的前 5 位表示阶码的符号及数值，后 11 位表示尾数的符号及数值，如下所示：

15	14 13 12 11	10	9 8 7 6 5 4 3 2 1 0
J_f	J	S_f	S

其中，J_f为阶符，J 为阶码，S_f为尾数符，S 为尾数。

有了这样的规定，如果尾数使用纯小数表示，则数 101.11 和 10.111 的实际表示形式为

0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	0
0	0	0	1	0	0	1	0	1	1	1	0	0	0	0	0

2．带符号数的代码表示及其运算

在数字系统中，数的表示有些是带符号的，有些是可以不带符号的。比如用来表示地址的数可以不带符号，而在计算机中一般进行运算的数都是带符号的。

在大家的习惯里，在表示带符号数时，通常在数值前面加上符号，正数用“+”，常被省略，负数用“−”表示，如：

$$(+15)_{10}=(+1111)_2$$
$$(-27)_{10}=(-11011)_2$$

以上表示法常被称为带符号数的“真值”表示，所表示的是数的真值。

在数字系统中所谓带符号数的表示是指带符号数的数值部分和符号部分都用统一的 0 或 1 两种数字符表示，常称为代码表示。下面介绍三种代码表示，即原码、反码和补码表示。这里常使用数的定点表示法。

1）*原码*

原码是在数值位左面加上符号位的表示方法。对于正数，符号位记 0；对于负数，符号位记 1。因此，原码表示又叫“符号+数值”表示法。根据这一法则，可以很容易地写出，8 位二进制数表示的原码所对应的十进制数值。

例如：
$x=(+1010111)_2=(+87)_{10}$　$[x]_{原}$=01010111
$x=(-1010111)_2=(-87)_{10}$　$[x]_{原}$=11010111
$x=(-1111111)_2=(-127)_{10}$　$[x]_{原}$=11111111

数值零在原码中有两种表示方法：00000000 和 10000000，即“+0”和“−0”，其真值是一样的。在用原码表示的带符号数的系统即原码系统中所包含的正数和负数的数值范围是一样的。一个 n 位原码系统的数值范围是从$-(2^{n-1}-1)$到$+(2^{n-1}-1)$，其中包括两种零的表示方法。

2）*反码*

反码也是在数值位的左面加上一位符号位，而且反码符号位和原码符号位一样：0 表示正数，1 表示负数。但反码数值位与它的符号位有关：对于正数，反码与原码完全相同；对于负数，该数的反码数值位是原码数值位按位取反。这也是反码名称的由来。它也被称为按位对 1 的补。

例如：
$x=(+19)_{10}=(+0010011)_2$　$[x]_{原}$=00010011　$[x]_{反}$=00010011
$x=(-19)_{10}=(-0010011)_2$　$[x]_{原}$=10010011　$[x]_{反}$=11101100
$x=(-127)_{10}=(-1111111)_2$　$[x]_{原}$=11111111　$[x]_{反}$=10000000

数值零在反码中也有两种表示方法：00000000 和 11111111，即“+0”和“−0”，它们的真值相同。在用反码表示的带符号数的系统即反码系统中所包含的正数和负数的数值范围也是一样的。一个 n 位反码系统的数值范围是从$-(2^{n-1}-1)$到$+(2^{n-1}-1)$，其中包括两种零的表示方法。

3）*补码*

补码和原码、反码相比较，补码最左面一位符号位的规定与原码、反码相同。补码数值位也与符号位有关：对于正数，补码完全等同于原码和反码；对于负数，从原码转换到补码或补码转换到原码的规则是：数值位按位取反，并在最低位加 1，或称为“取反加 1”。

例如：
$[x]_{原}$=01110110　$[x]_{反}$=01110110　$[x]_{补}$=01110110
$[x]_{原}$=11110110　$[x]_{反}$=10001001　$[x]_{补}$=10001010

但是，如果 x=−128，则$[x]_{补}$=10000000。

显然，在使用补码表示的带符号数的系统即补码系统中，一个 n 位补码系统的数值范围是从-2^{n-1} 到$+(2^{n-1}-1)$，对于零，补码只有一种形式，即 00000000。

原码、反码和补码之间的关系可以由图 1.2 说明。

4）*带符号数的加、减运算*

带符号数的三种表示法的形成规则不同，算术运算的方法也不同。用原码进行算术运算使用起来并不方便。假如要建立一个原码运算的数字逻辑电路，此电路必须要首先判断加数的符号，才能确定进行何种操作。如果加数的符号相同，它就将两数的真值相加，然后给结果加上相同的符号；如果加数的符号不同，则它必须先比较两数真值的大小，然后用大数减去小数，再把大数的符号赋给结果。这些“加”、“减”、“判断比较”会使逻辑电路更加复杂。

如果使用补码和反码进行加、减运算就相对简单。因为使用它们的目的就是为了寻找出一种适合于加、减运算的统一法则。在补码、反码的运算规则中，可以将减去一个数看作加

上一个负数，“符号位”也被看成是一位数值位，该负数用补码或反码的形式表示，然后一律按加法规则进行运算，所得结果的符号位也就是正确结果的符号位。如果符号位不正确，则说明运算超出了范围，即产生了溢出等错误。

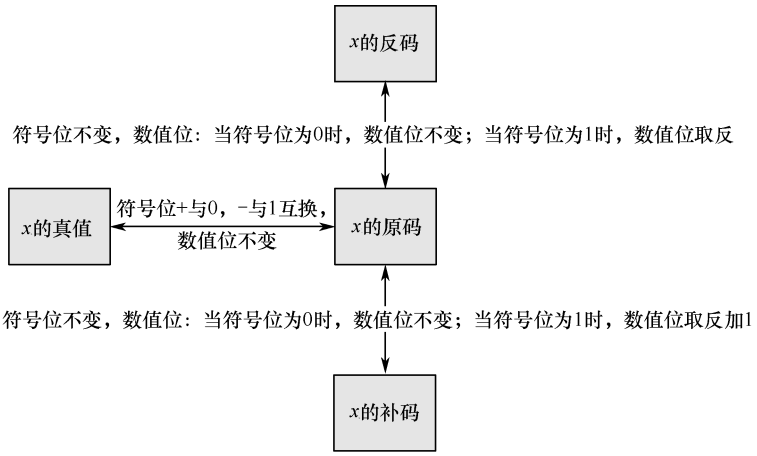

图 1.2　真值、原码、反码与补码之间的关系

补码运算和反码运算的差别在于：在补码运算时，符号位产生的进位要丢掉；而在反码运算时，符号位产生的进位要加到数值位的最低位。

【例 1-8】 求 $z=x-y$。其中，$x=+1011$，$y=+0010$。

解：（1）补码运算

$$[x]_{补}=01011 \qquad [-y]_{补}=[-0010]_{补}=11110$$

$$\begin{array}{r} 0\ 1\ 0\ 1\ 1 \\ +1\ 1\ 1\ 1\ 0 \\ \hline 丢掉\leftarrow 1\ 0\ 1\ 0\ 0\ 1 \end{array}$$

即$[z]_{补}=01001$，其真值为 $z=+1001$。通过真值的计算可以验证此结果。

注意，在补码运算中，可以把负数变成相应的补码，把减法归并为加法计算，符号位也参加了运算，而且符号位的进位丢掉了。这将简化电路的设计实现。

（2）反码运算

$$[x]_{反}=01011 \qquad [-y]_{反}=[-0010]_{反}=11101$$

$$\begin{array}{r} 0\ 1\ 0\ 1\ 1 \\ +1\ 1\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 0\ 0\ 0 \\ +\qquad\quad 1 \\ \hline 0\ 1\ 0\ 0\ 1 \end{array}$$

即$[z]_{反}=01001$，其真值为 $z=+1001$。

注意，在反码运算中，符号位的进位并没有丢掉，而是加到最低位去了。与补码计算相比较，可知计算结果是正确的。同样，也可以通过真值的计算验证此结果。

【例 1-9】 求 $z=x-y$。其中，$x=+0010$，$y=+1010$。

解：（1）补码运算

$$[x]_{补}=00010 \qquad [-y]_{补}=10110$$

$$\begin{array}{r} 0\,0\,0\,1\,0 \\ +\,1\,0\,1\,1\,0 \\ \hline 1\,1\,0\,0\,0 \end{array}$$

即$[z]_{补}$=11000，其真值 z=−1000。通过真值的计算可以验证此结果。

（2）反码运算

$[x]_{反}$=00010　　$[-y]_{反}$=10101

$$\begin{array}{r} 0\,0\,0\,1\,0 \\ +\,1\,0\,1\,0\,1 \\ \hline 1\,0\,1\,1\,1 \end{array}$$

即$[z]_{反}$=10111，其真值 z=−1000。与补码计算相比较，可知计算结果是正确的。

5）补码的本质

由于补码加、减运算的简便性，所以有必要对补码的本质做进一步的说明。为了理解补码，我们首先应该明白什么是“模”。

“模”也称为“模数”，就好比一个计量器的容量，一个计量器就构成了一个模数系统。如手表上显示 12 个小时，这 12 就是这个手表的模数，手表也构成一个模为 12 的模数系统。在模数系统中，通常称“模数减去一个数等于该数的补，或称减数与差互补”。如 12−1=11，将 11 称为 1 的补，1 称为 11 的补，或称 1 与 11 互补。**在模数系统中减去一个数就等于加上这个数的补。**如在手表中，X−1=X+11，假如 X=3，则 X−1=2 而且 X+11=14，在手表上 14 就是 2，所以也把 14 与 2 称为同余数。

在二进制的计算机系统中，对于字长为 8 的机器，它的容量是 256，所以 255 与 1 互补，也就是说减去 1 等于加上 255，通常称−1 的补码是 255。事实上，这也是对 256 求补。如果对 255 求补，则 1 的补就是 254，1 与 254 对于 255 来说互补，但不把 254 称为−1 的补码。补码是二进制数字系统中带符号数的一种表示。所以，书中常说，0 对 1 求补是 1，1 对 1 求补是 0，事实上是求反；2 对 9 求补是 7，7 对 9 求补是 2，事实上也是求反。所以书中常把“求反”称为“求补”就是这个原因，关键是对“谁”求补。

另外，在数字系统中，已知一个数的补码而求其相反数的补码，即已知$[x]_{补}$，求$[-x]_{补}$称之为求补运算。这里并不考虑 x 原本是正数还是负数。在补码体系中，符号位和数值位一起参与运算和转换，遵守完全一样的规则，例如求补运算时均为“按位取反加一”，这就使得补码运算变得简单。

1.2.3　十进制数的代码表示及其运算

在数字系统中，十进制数的表示还有一种称为十进制数的代码表示。它具有二进制数的形式，又具有十进制数的特点。这满足了数字系统必须使用二进制数运算的要求，所以，计算机可以对这种形式表示的数直接进行运算，但往往需要校正。

十进制数的常用代码主要有以下几种，见表 1.3。

从表 1.3 可以看出，在 4 位二进制代码所表示的十进制中，从 0000 到 1111 全部 16 个代码形式中，“8421”码占用了前 10 个，而舍弃了后 6 个代码；而“2421”码则采用了前 5 个和后 5 个代码表示，舍弃了中间 6 个代码；余 3 码由“8421”码加 0011 得到，它占用了中间 10 个，而舍弃了头、尾各 3 个代码。

表 1.3 3 种十进制数的代码表示法

十进制整数	8421 码	2421 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

1．“8421”码

“8421”码也称为二—十进制码或 BCD 码。它将十进制的每个数字符号用 4 位二进制表示，每位都有固定的权。因此，这种代码被称为有权码。按照从左到右的顺序，各位的权分别是 2^3，2^2，2^1 和 2^0，即 8，4，2，1。设“8421 码”4 位二进制数字符号为 $B_3B_2B_1B_0$，则它所代表的十进制数值为：$8B_3+4B_2+2B_1+B_0$。

“8421”码在十进制的 10 个数字符号的表示中与普通二进制中的表示完全一样，与二进制数对权的规定是一致的。但在“8421”码中没有 1010～1111 这 6 个代码。

“8421”码和十进制数之间的转换可以按组直接转换，例如：

$$(00010101)_{BCD}=(10101)_{BCD}=(15)_{10}$$

$$(1100001010001)_{BCD}=(1851)_{10}$$

注意，两个 BCD 码表示的数相加，其和可能不是一个正确的 BCD 码！？

2．“2421”码

如果假设“2421”码中 4 位二进制数字符号为 $B_3B_2B_1B_0$，则它们所代表的十进制数值为 $2B_3+4B_2+2B_1+B_0$。所以，“2421”码也是一种有权码，从左到右 4 位的权分别是 2，4，2，1。大家也已发现，其中有两位的权都是 2，这就可能使得“2421”码的编码方法会有多种，而且它们并不一定都是自补代码。

在十进制加、减运算中，常需要求取十进制数字对 9 的补，即求取 9 与该数字的差。例如，3 对 9 的补是 9−3=6，0 对 9 的补是 9−0=9。“2421”码能很方便地求得对 9 的补，故其被称为是一种对 9 的自补代码，即“2421”码表示的数，只要自身按位取反，就能得到该数对 9 之补的“2421”码。

例如，4 的“2421”码是 0100，4 对 9 的补是 9−4=5，而 5 的“2421”码是 1011，0100 是 1011 自身按位取反的结果，1011 也是 0100 自身按位取反的结果。

请问，两个 2421 码表示的数相加，其和是一个正确的 2421 码？

3．余 3 码

余 3 码是一种无权码，也就是说，该代码中各位的“1”不表示一个固定值。如果余 3 码中四位二进制数字符号为 $B_3B_2B_1B_0$，则它们所代表的数值为：$8B_3+4B_2+2B_1+B_0-3$。但是，余 3 码也是一种对 9 的自补代码。

同样，两个余 3 码表示的数相加，其和也可能不是一个正确结果的余 3 码！

对于余 3 码来说，由于每个余 3 码都“余 3”，其和就“余 6”。如果不进位，结果需要减 3；如果有进位，进位到高位的“1”在余 3 码中仅表示十进制数代码中的 10，而不是按

二进制数运算得到的进位所代表的是 16，也就是说少计算了 6 个数，则需要加 6。但是，原本余 3 码相加的结果“余 6”而结果需要减 3，则最后的结果就需要加 6 减 3，即需加 3。

1.2.4　可靠性编码

编码在形成、存储或传输过程中可能会发生错误。当用 BCD 码做加 1 计数时，从 3 变到 12，4 位都要发生变化。精确地说，对于计数的设备，其 4 位状态很有可能不能完全同时发生改变，于是就有可能出现下面分 4 个步骤完成转化的情况：

① 3 → 2

0 0 1 1 → 0 0 1 0 ；

② 2 → 0

0 0 1 0 → 0 0 0 0 ；

③ 0 → 8

0 0 0 0 → 1 0 0 0 ；

④ 8 → 12

1 0 0 0 → 1 1 0 0 。

可以看出，尽管从 3 最终变到 12，但这个转换过程产生了许多中间值。如果结果用来控制其他设备，那么控制将可能出现重大问题。

为了减少这种错误，采用了可靠性编码，该编码本身具有一种特征或能力，使得在形成中不易出错，或者编码在出错时容易被发现，甚至能查出出错的位置并予以纠正。常用的可靠性编码有格雷码、奇偶校验码和海明码等。下面将分别介绍这些编码。

1．格雷码

格雷码是指在一组数的编码中任意两个相邻数的编码只有一位二进制数不同的编码。格雷码有很多种，具有代表性的格雷码列于表 1.4 中。其中，典型格雷码可以通过二进制码直接转换而来，另外还有一个特点，就是所有对应于十进制数 2^m-1（m 为正整数）的典型格雷码仅在 m 位上有 1，其他位都为 0。

表 1.4　几种格雷码、步进码和二进制码对照表

十进制数	二进制码	典型格雷码	十进制格雷码	步进码
0	0000	0000	0000	00000
1	0001	0001	0001	00001
2	0010	0011	0011	00011
3	0011	0010	0010	00111
4	0100	0110	0110	01111
5	0101	0111	1110	11111
6	0110	0101	1010	11110
7	0111	0100	1011	11100
8	1000	1100	1001	11000
9	1001	1101	1000	10000
10	1010	1111		
11	1011	1110		
12	1100	1010		
13	1101	1011		
14	1110	1001		
15	1111	1000		

这就是说，对应于十进制数 2^m-1 的典型格雷码与十进制数 0 的典型格雷码之间都只有一位的差别。如果从这些数回到 0，都仍能保持一位差别的特点，所以又称这种**典型格雷码**为循环码，它特别适合用做二进制编码计数器，因为计数器计数的过程都是一个重复循环的过程。典型格雷码与二进制数之间存在一定的转换关系，这种关系如下：

假设二进制数为：$B_{n-1}B_{n-2}\cdots B_{i+1}B_i\cdots B_1B_0$，其等值的典型格雷码为：

$$G_{n-1}G_{n-2}\cdots G_{i+1}G_i\cdots G_1G_0;$$

则 $G_0=B_1\oplus B_0$，$G_1=B_2\oplus B_1$，$G_i=B_{i+1}\oplus B_i$，…，$G_{n-1}=0\oplus B_{n-1}=B_{n-1}$

因此，也就有：$B_{n-1}=G_{n-1}$，$B_{n-2}=G_{n-1}\oplus G_{n-2}$，…，$B_i=G_{n-1}\oplus G_{n-2}\oplus\cdots\oplus G_{i+1}\oplus G_i$，

$$B_0=G_{n-1}\oplus G_{n-2}\oplus\cdots\oplus G_{i+1}\oplus G_i\oplus\cdots\oplus G_1\oplus G_0$$

对于十进制编码计数器，典型格雷码是不可靠的。因而，要使用保持循环码特色的十进制格雷码，即 9 与 0 之间也只有一位差别的编码。表 1.4 中列出了一种十进制格雷码。

另外，还有一种格雷码，它是字长 5 位、循环长度为 10 的步进码，见表 1.4。这种步进码在加 1 计数时，只需将最左一位取反移到最右一位，而其余各位均左移一位即可。由于它实现起来特别容易，从而得到了广泛应用。

2．奇偶校验码

奇偶校验码是在数字系统中广泛采用的可靠性编码，它由若干个信息位加一个校验位构成，其中校验位的取值为 0 或 1。对于任何 n 位二进制数，只要增加一个校验位，便可构成 $n+1$ 位的奇偶校验码。所构成的编码“1”的个数为奇数的过程称为**奇校验**，所构成的编码为奇校验码；所构成的编码“1”的个数为偶数的过程称为**偶校验**，所构成的编码为偶校验码。

奇偶校验码具有发现一位出错的能力。但是，如果编码在存取过程中发生了两位出错，则简单地使用奇偶校验就检测不出它的错误来了。另外，奇偶校验码不具有自动纠错的能力，这是因为它虽能检测出一位出错，但不能确定错在哪一位。然而，如果在信息传送过程中采用横向与纵向奇偶检验码，对于一位出错就可以定位了。

总的来说，由于两位出错的概率远小于一位出错的概率，因而奇偶校验码仍不失为一种增加设备不多，而收益不少的实用的可靠性编码。

3．海明校验码

海明校验码是一种可靠性编码，一种既能检测错误又能纠正错误的编码。海明码也由“信息位”和“校验位”两部分构成。下面以“8421”海明校验码为例来说明。

“8421”海明码是由“8421”码加 3 位校验码所组成，简称“8421”海明码。设“8421”码为 $B_8B_4B_2B_1$，所加的 3 位校验码为 $P_3P_2P_1$，则“8421”海明码由下列位序的 7 位编码组成：

$$B_8B_4B_2P_3B_1P_2P_1$$

其中，校验位 $P_3P_2P_1$ 的值由下面公式确定：

$$P_3=B_8\oplus B_4\oplus B_2$$

$$P_2=B_8\oplus B_4\oplus B_1$$

$$P_1=B_8\oplus B_2\oplus B_1$$

例如，已知“8421”码为 $B_8B_4B_2B_1=0101$，则

$$P_3=B_8\oplus B_4\oplus B_2=0\oplus 1\oplus 0=1$$
$$P_2=B_8\oplus B_4\oplus B_1=0\oplus 1\oplus 1=0$$
$$P_1=B_8\oplus B_2\oplus B_1=0\oplus 0\oplus 1=1$$

根据海明码的位序要求，将校验位插入“8421”码，可得到“8421”码 0101 所对应的“8421”海明码为 0101101。下面分析海明码的校验与纠错原理。

从上面 $P_3P_2P_1$ 的三个公式可以知道，$P_3P_2P_1$ 分别是“8421”码中若干位的偶校验位。例如，P_3 的取值将使 B_8，B_4，B_2 和 P_3 中的“1”的个数为偶数。显然，按上述校验位公式所生成的“8421”海明码，若经过传输后不再满足“1”的个数为偶数的关系，便表明传输过程中出现了差错。

假定上述三个公式中 P_2P_1 不满足，且只有一位发生错误，便可肯定是 B_1 了，因为只有 B_1 包含在这两个公式中。判断其他位出错的方法与此相同。为了方便地找出错位，可以在接收端先算出“校验和”$S_3S_2S_1$：

$$S_3=B_8\oplus B_4\oplus B_2\oplus P_3$$
$$S_2=B_8\oplus B_4\oplus B_1\oplus P_2$$
$$S_1=B_8\oplus B_2\oplus B_1\oplus P_1$$

很明显，只有当 $S_3=S_2=S_1=0$ 时，才表示接收到的海明码是正确的。如果校验和表达式中的某一个 S_i 的值不为 0，则表示其等号右边各位中必有一个出错。于是，根据 $S_3S_2S_1$ 所表示的二进制值，便可确定上述“8421”海明码中位序为该值的那一位出了错。例如，

例如，发送编码 5 的海明码是：　0　1　0　1　1　0　1

接收到的海明码是：　0　1　0　1　0　0　1

根据接收到的海明码算出各校验和：

$$S_3=B_8\oplus B_4\oplus B_2\oplus P_3=0\oplus 1\oplus 0\oplus 1=0$$
$$S_2=B_8\oplus B_4\oplus B_1\oplus P_2=0\oplus 1\oplus 0\oplus 0=1$$
$$S_1=B_8\oplus B_2\oplus B_1\oplus P_1=0\oplus 0\oplus 0\oplus 1=1$$

由 $S_3S_2S_1$ 构成的校验和是 3，可知接收到的第 3 位（B_1）出现了错误，则将第 3 位“0”改为“1”就纠正了错误。**多么巧妙呀！你能想到这样的方法吗？**

应该强调，**纠错仅在出现单错的情况下有效。**这里的“8421”海明码是由 4 位信息位和 3 位校验位组成的。那么，n 位信息码需要增加多少校验码，才能组成能够检测并校正一位错的海明码呢？

假设信息码为 n 位，校验码为 k 位，则可推得

$$(2^k-1)-k\geqslant n$$

或

$$2^k-1\geqslant k+n$$

这就是说，如果校验位为 k 位，则最多可以校验的信息码的位数是 n。例如，k=4，则 $n\leqslant 11$，即信息位最多为 11 位，海明码位数最多 15 位。

1.3　逻辑代数基础

布尔代数（1854 年，英国数学家 George Boole 提出）是将人的逻辑思维规律和推理过程归结为一种数学运算的代数系统。

逻辑代数，即开关代数，又称为二值布尔代数（1938 年，贝尔实验室研究员 Claude E．Shannon 提出），是将布尔代数的一些基本前提和定理应用于开关电路的分析与描述，是一个由逻辑变量集、常量 0 和 1 及“与”、“或”、“非”三种运算符所构成的代数系统。逻辑代数是二值逻辑运算中的基本数学工具，它研究了逻辑变量之间的因果关系以及依据这些关系进行的逻辑推理，它是逻辑电路的数学基础，被广泛地应用于数字系统的分析和设计中。

1.3.1 逻辑代数常用概念及逻辑运算

1．逻辑代数中的常用概念

这里先介绍逻辑代数中几个常用的概念。

1）逻辑状态

在一定条件下，事物的某种性质仅表现为两种互不相容的状态，两种状态必出现且仅出现一种，一种状态是另一种状态的反状态。逻辑状态用来描述逻辑电路的功能。

例如，信号的有与无，电路的开与关，事物的是与非、真与假等。

逻辑状态就是用符号“0”和“1”分别表示这两种状态。这里的“0”和“1”是符号，不是通常的数，代表两种状态，通常也称之为 0 状态和 1 状态。0 状态一般表示逻辑条件的假或无效，1 状态一般表示逻辑条件的真或有效。

2）逻辑变量

随着条件的变化，事物的状态会发生变化，表示事物状态的逻辑状态也随之变化，这种没有被确定的逻辑状态称为逻辑变量。逻辑变量反映逻辑状态的变化，逻辑变量仅能取值“0”或“1”。而“0”和“1”被称为逻辑常量。

3）逻辑电平

在逻辑电路中，把电压这样的物理量离散成两种相对于参考地的电压值，即两种电平：高电平和低电平。抽象化的高、低电平代表着一定范围的物理量的电压值，忽略了物理量值的实际含义，还可以用来表示其他物理含义。在采用不同工艺的器件组成的电路中，高、低电平代表的范围是不同的，当电源电压为 5V 时，见表 1.5。

表 1.5 不同工艺器件定义的逻辑电平

工 艺	逻 辑 电 平	
	L	H
TTL	0～0.40V	3.0～5.0V
CMOS	0～0.80V	2.0～5.0V

通常用 H 代表具有正得较多的或正的电平，用 L 代表具有正得较少的或负的电平；或者，用 H 电平表示在规定时间内一定幅度的正脉冲的出现，用 L 电平表示在规定时间内没有脉冲，或者负脉冲的出现。另外，大部分电平物理量在定义的高、低电平之间有一个逻辑不确定区间，称为**噪声区**。如果电平物理量稳定于噪音区，则称为**逻辑模糊**。在逻辑电路中不允许出现逻辑模糊。

4）逻辑约定

逻辑电路中的物理量通常用代表物理量值的逻辑电平表示，而描述器件功能的是逻辑状态，因此必须规定逻辑电平与逻辑状态之间的关系，这种关系被称为逻辑规定，又称为逻辑约定，也被称为逻辑化过程。有两种约定方法，即正逻辑约定和负逻辑约定。

- 正逻辑约定：高电平 H 用状态 1 表示，低电平用状态 0 表示；