高等学校教材

数字逻辑与数字系统设计

张少敏　陈基禄　郑顾平　王保义

高等教育出版社

数字逻辑与数字系统设计

高 等 学 校 教 材

数字逻辑与数字系统设计

张少敏　陈基禄　郑顾平　王保义

高等教育出版社

内容提要

本书以知识性、实用性和先进性为宗旨，根据作者多年来从事计算机系统的研制经验和教学体会，全面而系统地介绍数字逻辑设计的基础知识，包括：数制与编码，布尔代数及其逻辑实现，集成逻辑门电路，触发器，为电路分析和设计准备基础知识；数字电路的分析和设计方法，包括：组合逻辑电路，同步时序逻辑电路和异步时序逻辑电路以及脉冲产生与整形电路；可编程逻辑器件，包括 ROM、PLA、PAL、GAL 及在系统编程技术。

本书内容丰富，力求反映数字电路与数字逻辑设计的最新发展技术，收编了大量的数字电路与数字逻辑设计的应用设计实例，做到理论联系实际，原理、技术和应用并重，内容由浅入深，通俗易懂。

本书可作为高等学校计算机、通信、电子、自动化类各专业的教学用书，也可作为从事计算机应用的工程技术人员的学习与参考用书。

图书在版编目(CIP)数据

数字逻辑与数字系统设计/张少敏等．—北京：高等教育出版社，2006.7

ISBN 7-04-019585-2

Ⅰ.数… Ⅱ.张… Ⅲ.①数字逻辑-高等学校-教材 ②数字系统-系统设计-高等学校-教材 Ⅳ.①TP302.2②TP271

中国版本图书馆 CIP 数据核字(2006)第 065223 号

策划编辑 倪文慧 **责任编辑** 欧阳舟 **封面设计** 刘晓翔 **责任绘图** 朱 静
版式设计 陆瑞红 **责任校对** 杨雪莲 **责任印制** 朱学忠

出版发行	高等教育出版社	**购书热线**	010-58581118
社　　址	北京市西城区德外大街4号	**免费咨询**	800-810-0598
邮政编码	100011	**网　　址**	http://www.hep.edu.cn
总　　机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经　　销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印　　刷	北京新丰印刷厂	**畅想教育**	http://www.widedu.com
开　　本	787×1092 1/16	**版　　次**	2006年7月第1版
印　　张	24.5	**印　　次**	2006年7月第1次印刷
字　　数	550 000	**定　　价**	30.50元

物料号　19585-00

前 言

数字逻辑和数字电路是计算机类各专业的主干基础课程，为了适应教学改革和课程整合的需要，将两门课程合并为一门课程，以压缩学时，精炼内容。在内容安排上以数字逻辑设计为主，辅以数字电路部分，互相融合，形成数字系统整体结构。本教材就是为适应这个需要而编写的，以改变长期以来需要选用两本教材的不便。

由于数字电子技术发展迅猛，教材内容力求跟上技术发展的需要，在保留经典内容体系的基础上增加了中、大规模集成电路的特性和应用，特别是在系统编程技术，它是数字系统设计的一次革命，教材内容将适应计算机专业人员对数字系统设计知识的需要。在内容编写过程中，力求由浅入深，层次清晰，简明扼要，概念清楚，步骤明确，重点讨论数字电路的分析和设计方法。考虑到读者参阅其资料的方便，本书在给出数字逻辑电路的国家标准的同时，还给出了惯用符号及国外一些国家使用的符号，以适应读者设计和调试电路的需要。

数字逻辑与系统设计课程是实践性很强的课程，除理论教学外，进行设计性实验是课程的重要内容。课程总学时安排70～80学时，其中实验学时安排20～30学时。每章后都给出了适量习题供读者选做。同时编写有《数字逻辑与数字系统设计实验指导书》，并开发有相应配套的实验仪。

全书共分10章，第一、二章介绍数字逻辑设计的基础知识，包括：数制与编码，布尔代数及其逻辑实现；第三、五章介绍集成逻辑门电路及触发器，为电路分析和设计准备基础知识；第四、六、七、八章介绍数字电路的分析和设计方法，包括：组合逻辑电路，同步时序逻辑电路和异步时序逻辑电路以及脉冲产生与整形电路；第九、十章介绍可编程逻辑器件，包括ROM、PLA、PAL、GAL及在系统编程技术。

本书可作为计算机类的教材，亦可作为电气类各专业的参考书，还可作为相关科技人员的参考资料。

本书由张少敏副教授编写第八章、第十章，并对全书进行了统稿；陈基禄教授编写第一～四章，并制定了教材编写大纲；郑顾平副教授编写了第五～七章；王保义教授编写了第九章。

在教材编写过程中得到了华北电力大学教务部门及计算机系老师的指导，提出了许多宝贵意见，谨向他们表示诚挚的感谢。

由于作者学识所限，书中难免有错误和不妥之处，恳请读者批评指正。

编 者

2005年12月

目　　录

说明:加“*”的章节,可根据学时的安排,作为选学内容。

第一章　数制与编码

数字计算机现已广泛地应用于科学计算、数据处理和过程控制等领域。数字计算机是数字系统中最常见、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。离散元素通常是以二进制的形式出现的。为此,必须讨论数的代码特征及运算。

本章讨论数制和编码的概念。通过本章学习,读者将熟悉数字系统数制和编码的表示方法、性质及相互间的转换,带符号数和带小数点数的表示方法及字符编码,为离散元素的处理打下基础。

第一节　进位计数制

一、十进制计数制

十进制计数制早已为人们所熟悉。在十进制计数制中,采用了 10 个有序数字符号 0、1、2、3、4、5、6、7、8、9 和一个小数点符号“.”,并且是“逢十进一”。任何一个十进制数$(S)_{10}$,可以表示为

$$(S)_{10} = k_{n-1} \times 10^{n-1} + k_{n-2} \times 10^{n-2} + \cdots + k_0 \times 10^0 + k_{-1} \times 10^{-1} + \cdots + k_{-m} \times 10^{-m}$$
$$= \sum_{i=-m}^{n-1} k_i \times 10^i$$

其中,k_i 可以是 0～9 这 10 个数码中的任一个,m 和 n 是正整数,k_i、m、n 均由$(S)_{10}$决定,(S)的下标与式中的 10 是十进制的基数。由于基数为 10,每个数位计满 10 就向高位进位,即逢十进一,所以称它为十进制计数制。

例如,一个十进制数 2 003.3,可以表示为

$$(2\,003.3)_{10} = 2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 3 \times 10^0 + 3 \times 10^{-1}$$

二、二进制计数制

在数字系统中,为了便于工程实现,广泛采用二进制计数制。这是因为,二进制的每一位只用两个数码**0**和**1**,因而可以用具有两个不同稳定状态的电子元件来表示,并且二进制数运算简单,数的存储和传送也可用简单而可靠的方式进行。二进制基数是 2,其计数规律是逢二进一。

任意一个二进制数可以表示为

$$(S)_2 = k_{n-1} \times 2^{n-1} + k_{n-2} \times 2^{n-2} + \cdots + k_0 \times 2^0 + k_{-1} \times 2^{-1} + \cdots + k_{-m} \times 2^{-m} = \sum_{i=-m}^{n-1} k_i \times 2^i$$

其中，k_i 只能取**0**或**1**，m、n 为正整数，由$(S)_2$ 决定。

例如，一个二进制数**1011.101**可以表示为

$$(\mathbf{1011.101})_2=\mathbf{1}\times2^3+\mathbf{0}\times2^2+\mathbf{1}\times2^1+\mathbf{1}\times2^0+\mathbf{1}\times2^{-1}+\mathbf{0}\times2^{-2}+\mathbf{1}\times2^{-3}$$

三、八进制计数制和十六进制计数制

采用二进制计数制，对计算机等数字系统来说，数据的运算、存储和传送都极为方便，但二进制书写很不方便，且记忆困难，为此人们经常采用八进制计数制和十六进制计数制来进行书写或打印。

任意一个八进制数可以表示为

$$(S)_8=\sum_{i=-m}^{n-1}k_i\times8^i$$

其中，k_i 可取 0、1、2、…、7 这 8 个数码之一，m 和 n 为正整数，由$(S)_8$ 决定。八进制的基数是 8，其计数规律是逢八进一。

例如，八进制数$(537.25)_8$ 可以表示为

$$(537.25)_8=5\times8^2+3\times8^1+7\times8^0+2\times8^{-1}+5\times8^{-2}$$

任意一个十六进制数可以表示为

$$(S)_{16}=\sum_{i=-m}^{n-1}k_i\times16^i$$

其中，k_i 可取 0、1、2、…、8、9、A、B、C、D、E、F 等 10 个数码及 6 个字母之一，m 和 n 为正整数，由$(S)_{16}$决定，十六进制数的基数是 16，其计数规律是逢十六进一。

例如，一个十六进制数$(\mathrm{7FAE3.8C})_{16}$可以表示为

$$(\mathrm{7FAE3.8C})_{16}=7\times16^4+\mathrm{F}\times16^3+\mathrm{A}\times16^2+\mathrm{E}\times16^1+3\times16^0+8\times16^{-1}+\mathrm{C}\times16^{-2}$$

第二节 数制转换

一、整数转换方法

人们习惯的是十进制数，计算机及数字系统采用的是二进制数，人们书写时又多采用八进制数或十六进制数，因此，必然产生各种进位计数制间的相互转换问题，以便计算机和人之间相互理解。

对于基值大的整数转换成基值小的整数时，一般采用除基取余法进行，例如$(935)_{10}=(?)_2$可以用如下方法：

$$\begin{array}{ccccccccccccccccccccc}
1 & \leftarrow & 3 & \leftarrow & 7 & \leftarrow & 14 & \leftarrow & 29 & \leftarrow & 58 & \leftarrow & 116 & \leftarrow & 233 & \leftarrow & 467 & \leftarrow & 935 & \div 2 \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \\
\mathbf{1} & & \mathbf{1} & & \mathbf{1} & & \mathbf{0} & & \mathbf{1} & & \mathbf{0} & & \mathbf{0} & & \mathbf{1} & & \mathbf{1} & & \mathbf{1} & \text{余数}
\end{array}$$

转换结果为$(935)_{10}=(\mathbf{1110100111})_2$。

对于基值小的整数转换成基值大的整数时，一般采用按幂展开法进行，例如：

$$(\mathbf{1101001})_2=\mathbf{1}\times 2^6+\mathbf{1}\times 2^5+\mathbf{0}\times 2^4+\mathbf{1}\times 2^3+\mathbf{0}\times 2^2+\mathbf{0}\times 2^1+\mathbf{1}\times 2^0=(105)_{10}$$

二、小数转换方法

对于基值大的小数转换成基值小的小数时，一般采用乘基取整法进行，例如$(0.875)_{10}=(?)_2$可以用如下方法：

$$\begin{array}{ccccccccl}
0.875 & \rightarrow & 0.75 & \rightarrow & 0.5 & \rightarrow & 0.0 & & \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \times 2 \\
\mathbf{1} & & \mathbf{1} & & \mathbf{1} & & \mathbf{0} & & \text{整数}
\end{array}$$

转换结果为$(0.875)_{10}=(\mathbf{0.1110})_2$。

同样对于基值小的小数转换成基值大的小数时，一般采用按幂展开法进行，例如：

$$(\mathbf{0.1101})_2=\mathbf{1}\times 2^{-1}+\mathbf{1}\times 2^{-2}+\mathbf{0}\times 2^{-3}+\mathbf{1}\times 2^{-4}=(0.8125)_{10}$$

三、二进制数与八进制数和十六进制数之间的转换

二进制数与八进制数和十六进制数之间有着确定的对应关系，对于二进制数与八进制数之间为每 3 位二进制数对应 1 位八进制数，当二进制数转换为八进制数时，从小数点开始分别向左或向右每 3 位一组划分，不足 3 位时用**0**填补，即可得到等值的八进制数，例如：

$$(\mathbf{1101011.1001})_2=(153.44)_8$$

而八进制数转换为二进制数时，则直接展开成 3 位一组二进制数即可，然后去掉整数最高位的**0**和小数最低位的**0**，例如：

$$(1635.724)_8=(\mathbf{001110011101.111010100})_2=(\mathbf{1110011101.1110101})_2$$

对于二进制数与十六进制数之间为每 4 位二进制数对应 1 位十六进制数，当二进制数转换为十六进制数时，从小数点开始分别向左或向右按每 4 位一组划分，不足 4 位时用**0**填补，即可得到等值的十六进制数，例如：

$$(\mathbf{110100111.101101})_2=(1A7.B4)_{16}$$

而十六进制数转换为二进制数时，则直接展开成 4 位一组二进制数即可，然后去掉整数最高位的**0**和小数最低位的**0**，例如：

$$(3FC7.A4)_{16}=(\mathbf{0011111111000111.10100100})_2=(\mathbf{11111111000111.101001})_2$$

四、数制转换中应注意的问题

(1) 实数转换时应将整数和小数分别转换，然后再拼接起来。

(2) 基值大的小数不一定有确定对应的基值小的小数，可能出现无限循环或无限不循环的小数，例如十进制小数 0.6，其等效的二进制小数**0.100110011001**…，为一无限循环小数，应按精度要求取相应位数。

(3) 不同进位计数制可分别用后缀来标注,B 表示二进制数,Q 表示八进制数,H 表示十六进制数,D 或缺省则表示十进制数。例如:

$$139D = \mathbf{10001011}B = 213Q = 8BH$$

第三节 带符号二进制数的表示方法

一、机器数与真值

带符号二进制数在计算机内部如何表示呢,通常将数的符号数码化,即用**0**表示"+",用**1**表示"-",例如,数$(-\mathbf{1101})_2$是机器数$(\mathbf{11101})_2$的真值;而数$(+\mathbf{1011})_2$则是$(\mathbf{01011})_2$的真值。

二、原码表示法

原码定义
$$[X]_{原} = \begin{cases} X & 1 > X \geqslant 0 \\ \mathbf{1} - X & -1 < X \leqslant 0 \end{cases}$$

由原码定义可以看出,任何大于 0 的数的原码就是符号位为**0**,尾数为真值本身;小于 0 的数的原码符号位为**1**,尾数仍为真值本身,从真值求原码就是正数前加符号**0**,负数前加符号**1**即可。例如:

$$X = +\mathbf{0.1011}, \qquad [X]_{原} = \mathbf{0.1011}$$
$$X = -\mathbf{0.1011}, \qquad [X]_{原} = \mathbf{1.1011}$$

由原码定义可知,$[+\mathbf{0}]_{原} = \mathbf{0.0000}$,而$[-\mathbf{0}]_{原} = \mathbf{1.0000}$,说明**0**在原码表示中不是唯一的。

如用 X_0 表示符号位,则有

$$X_0 = \begin{cases} \mathbf{1} & X \leqslant 0 \\ \mathbf{0} & X \geqslant 0 \end{cases}$$

三、补码表示法

原码表示通俗易懂,而且与真值间转换很简单。但是用原码表示时加/减运算很不方便。例如,当 A、B 两数相加时,还要视其同号,还是异号。若同号,则数值相加,结果符号不变;如果两数异号,则实际做减法,并判断两数的绝对值,让绝对值大的数减去绝对值小的数,其结果符号与绝对值大的数相同。在整个操作中费时、费设备。为了解决这些矛盾,人们寻求一种补码表示法。

1. 补码的概念和定义

补码是根据同余的概念提出来的,例如:

$$8 - 4 = 8 + 6 \qquad (\text{mod } 10)$$
$$11 - 2 = 11 + 10 \qquad (\text{mod } 12)$$

其中 10 和 12 称为模数，在相应模数下等号两边余数是相同的，当以 10 为模时 8 - 4 可以用 8 + 6 来计算，6 就是以 10 为模 - 4 的补码。同样 10 就是以 12 为模 - 2 的补码。对于小于 1 的二进制数的补码定义如下：

$$[X]_{补} = \begin{cases} X & 0 \leqslant X < 1 \\ 2 + X & -1 \leqslant X \leqslant 0 \end{cases} \quad (\mathrm{mod}\ 2)$$

或写成同余式：

$$[X]_{补} = X(\mathrm{mod}\ 2)$$

2. 补码的转换

(1) 由真值求补码。根据补码的定义，任何正数的补码就是真值本身；负数的补码按其定义为

$$[X]_{补} = 2 + X$$

如

$$X = -\mathbf{0}.\ X_1 X_2 \cdots X_n$$

$$\begin{aligned}[X]_{补} &= \mathbf{10.000\cdots00} + (-\mathbf{0}.\ X_1 X_2 \cdots X_n) \\ &= \mathbf{1.111\cdots11} + \mathbf{0.000\cdots01} - \mathbf{0}.\ X_1 X_2 \cdots X_n \\ &= \mathbf{1}.\overline{X_1}\ \overline{X_2}\cdots\overline{X_n} + \mathbf{0.000\cdots01}\end{aligned}$$

其中

$$\mathbf{1} = \sum_{i=1}^{n} 2^{-i} + 2^{-n}$$

从而得到由负数值求补码的方法：符号位为**1**，尾数各位取反（**0**变**1**，**1**变**0**），然后在末位加**1**（简称除符号位外求反加**1**）就是$[X]_{补}$。

(2) 由原码求补码。由原码和补码的定义可知，正数的原码和补码相同，就是真值的表示。

对于负数 X，当 $X = -\mathbf{0}.\ X_1 X_2 \cdots X_n$

$$[X]_{原} = \mathbf{1}.\ X_1 X_2 \cdots X_n$$

$$\begin{aligned}[X]_{补} &= 2 + X = 2 - \mathbf{0}.\ X_1 X_2 \cdots X_n \\ &= \mathbf{1} + (\mathbf{1} - \mathbf{0}.\ X_1 X_2 \cdots X_n) \\ &= \mathbf{1} + (\mathbf{0}.\overline{X_1}\ \overline{X_2}\cdots\overline{X_n} + \mathbf{0.000\cdots01})\end{aligned}$$

这个结果表明任何负数的补码$[X]_{补}$等于它的原码除符号位以外各位“求反加**1**”。

(3) 由一个数的补码表示求其负数的表示

这是实现补码减法运算的基础。

若$[X]_{补} = \mathbf{0}.\ X_1 X_2 \cdots X_n$　　则 $[X]_{原} = \mathbf{0}.\ X_1 X_2 \cdots X_n$

$-X = -\mathbf{0}.\ X_1 X_2 \cdots X_n$　　$[-X]_{补} = \mathbf{1}.\overline{X_1}\ \overline{X_2}\cdots\overline{X_n} + 2^{-n}$

若$[X]_{补} = \mathbf{1}.\ X_1 X_2 \cdots X_n$　　则$[X]_{原} = \mathbf{1}.\overline{X_1}\ \overline{X_2}\cdots\overline{X_n} + 2^{-n}$

$-X = -[-(\mathbf{0}.\overline{X_1}\ \overline{X_2}\cdots\overline{X_n} + 2^{-n})]$　　$[-X]_{补} = \mathbf{0}.\overline{X_1}\ \overline{X_2}\cdots\overline{X_n} + 2^{-n}$

结论：对绝对值小于 1 的任何正、负数 X，通过对$[X]_{补}$连同符号位一起“求反加**1**”便可得到$[-X]_{补}$，称$[-X]_{补}$为$[X]_{补}$的机器负数，由$[X]_{补}$求$[-X]_{补}$的过程称为对$[X]_{补}$求$[-X]$的补码。

3. 补码的性质

(1) 若用 X_0 表示符号位，则有

$$X_0=\begin{cases}\mathbf{1} & X<0\\ \mathbf{0} & X>0\end{cases}$$

（2）真值0在补码表示中是唯一的。

$$\begin{aligned}[+\mathbf{0}]_{补}&=\mathbf{0.000\cdots0}\\ [-\mathbf{0}]_{补}&=\mathbf{10.000\cdots0-0.000\cdots0}\\ &=\mathbf{10.00\cdots00}\\ &=\mathbf{0.000\cdots0}(\text{mod}2)\end{aligned}$$

即$[+\mathbf{0}]_{补}=[-\mathbf{0}]_{补}=\mathbf{0.000\cdots0}$

4．补码的运算规则

$$[X]_{补}\pm[Y]_{补}=[X\pm Y]_{补}$$

即补码运算结果仍然是补码，当结果的最高位有向上进位时，丢掉不要，即为正确结果。

5．补码的溢出判断与变形补码

当两正数相加结果超过 1 或两负数相加结果超过 -1 时，其结果超出了机器所能表示数的范围，则产生溢出错误，为能正确判断其溢出错误常采用变形补码，即采用两位符号位。

$$[X]_{补}=\begin{cases}X & 1>X\geqslant0\\ 4+X & 0\geqslant X\geqslant-1\end{cases}\qquad(\text{mod}4)$$

例如：

$$X=\mathbf{0.101101}\qquad[X]_{补}=\mathbf{00.101101}$$
$$X=-\mathbf{0.101101}\qquad[X]_{补}=\mathbf{11.010011}$$

当两数运算结果符号位相异时，则产生溢出，结果符号$=\mathbf{01}$为正溢出，结果符号$=\mathbf{10}$为负溢出。例如：$X=\mathbf{0.1101}$，$Y=\mathbf{0.1010}$，求$[X+Y]_{补}$。

$$[X]_{补}=\mathbf{00.1101},\qquad[Y]_{补}=\mathbf{00.1010}$$
$$[X]_{补}+[Y]_{补}=\mathbf{01.0111},\qquad\text{产生正溢出。}$$

四、反码表示法

在补码表示中，如果只求反末位不加1，就得到了另一种机器数表示法——反码表示法。可从补码的定义推出反码的定义为

$$[X]_{反}=\begin{cases}X & 1>X\geqslant0\\ (2-2^{-n})+X & 0\geqslant X>-1\end{cases}\qquad(\text{mod}2-2^{-n})$$

其中，n 为尾数的位数。

在反码表示中0的表示不是唯一的，$[+\mathbf{0}]_{反}=\mathbf{0.00\cdots00}$，而$[-\mathbf{0}]_{反}=\mathbf{1.11\cdots11}$。

反码的运算规则为

$$[X]_{反}\pm[Y]_{反}=[X\pm Y]_{反}$$

结果为反码，如果结果的最高位有向上的进位，则应加到结果的末位，即循环进位，才能得到

正确结果。

第四节　数的定点与浮点表示法

一、数的定点表示法

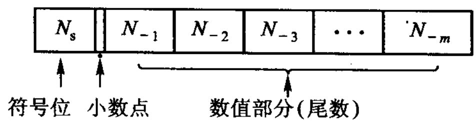
定点数是各种数据表示中最简单,最基本的一种数据表示形式,它用于表示二进制形式具有固定比例换算的量。在定点表示中约定机器中所有数据的小数点位置是固定不变的,因而小数点就不必要再使用记号表示。

1. 定点小数格式

小数点位置固定在最高有效数位的左边。任一定点小数表示为

$$N = N_s . N_{-1} N_{-2} N_{-3} \cdots N_{-m} \qquad (m\ 为数据位数)$$

在计算机中的表示形式为

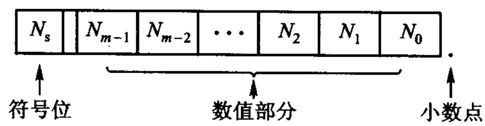


2. 定点整数格式

小数点固定在最低数据位的右边,任一定点整数表示为

$$N = N_s N_{m-1} N_{m-2} N_{m-3} \cdots N_2 N_1 N_0 \qquad (其中\ m\ 为数据位数)$$

在计算机中整数表示形式为



定点数可以表示为带符号数或不带符号数。表示算术操作数时,应用带符号的数,一般以左边最高位表示符号位。不带符号的数一般表示逻辑量或某些特征值。

二、浮点表示法

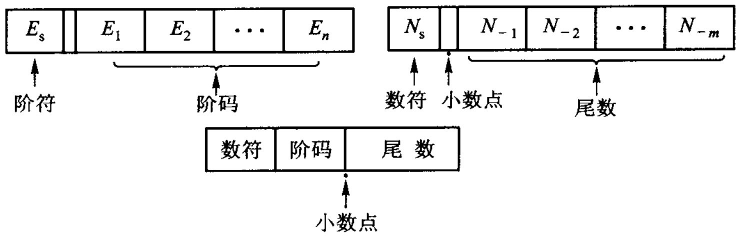
定点数所表示的数值范围在许多应用中是不够用的,特别是科学计算方面,数值变化很大,一般在 $10^{-10} \sim 10^{10}$之间,在天文学计算中,可能包括电子质量数据为 10^{-28} g,太阳质量数据为 10^{33} g,其数值范围超过 10^{60},如若引入浮点数,表示是非常方便的。

一个数 N 的浮点形式可表示为

$$N = M \cdot R_m^E$$

其中,M(Mantissa)表示尾数,E(Exponent)表示阶码,R_m(Radix)表示基数。R_m 一般为 2、8、16

等,由于 R_m 是常数,不需在数码中表示出来,故浮点数只需用一对定点数表示。一个尾数是 M,在浮点表示中尾数通常为规格化的纯小数,即 $|M| \geqslant \frac{1}{R_m}$,另一个是阶码,通常是整数。浮点数在计算机中的表示形式为



第五节 字符编码

在数字系统中,要求处理的信息不仅是数码,而且大量处理非数码信息,包括文字、字母和其他专用符号,这些符号都必须按照一定的规则用一组二进制编码来表示才能被计算机所识别、处理、存储和传送。

1. ASCII 码

ASCII 码是美国信息交换标准代码。ASCII 码是用 7 位二进制码编码的,故可表示 $2^7 = 128$ 个字符,其中包括:10 个十进制数(0~9),52 个大、小写英文字母(A~Z,a~z),32 个通用控制字符,34 个专用字符。我国标准"信息交换用的七位编码字符集 80—GB1988"规定了信息处理交换用的图形字符和控制字符共 128 个,每个字符用 7 位二进制数码进行编码,与 ASCII 码兼容。

2. 十进制数编码

在数字系统中十进制数也是应用二进制数编码的,即二进制编码的十进制数,其中应用最广泛的是 ASCII 码和 BCD 码,ASCII 码的十进制数主要用于非数值计算,它们所表示的数串可以作为字符串来运算或操作。作为十进制数进行算术运算则采用 BCD 码。

ASCII 码十进制数是用一个字节来表示的,0~9 等 10 个数字用 ASCII 码字符集中 30~39(十六进制数)10 个编码表示。BCD 码是用二进制编码的十进制数码(通常用 8421 有权码),每两个十进制数占一个字节,称为压缩十进制数串,其符号则用 BCD 码中的冗余码来表示,如用 10、12、14 或 15 表示"+";用 11 或 13 表示"-"。

对于 BCD 码表示的十进制数在进行加/减运算时要考虑修正问题,对 8421 编码的十进制数,其修正方法是:当两十进制数相加结果小于等于 9 时,不修正;当结果大于 9 或产生向高位进位时,均加 6 修正,以得到正确结果。例如:

$$X = 358 \quad Y = 929$$

8421 码表示为

$$X = 0011 \quad 0101 \quad 1000$$
$$Y = 1001 \quad 0010 \quad 1001$$

先按二进制数运算规则计算：

$$
\begin{array}{rrrr}
 & 0011 & 0101 & 1000 \\
+) & 1001 & 0010 & 1001 \\
\hline
 & 1100 & 1000 & 1\ 0001 \\
\text{结果修正：}\quad +) & 0110 & 0000 & 0110 \\
\hline
 & 1\ 0010 & 1000 & 0111
\end{array}
$$

其中个位和有向十位进位，个位和需修正，百位和大于9，亦需进行修正得到正确结果。

$$X + Y = 1\ 287$$

3．汉字编码

中国是占世界人口近四分之一的国家，对汉字的处理能力影响着计算机的广泛应用，国内外专家都对汉字信息处理进行了大量研究，取得了许多成果，对汉字信息不仅可以通过键盘进行编码输入，而且已能进行声音或图像输入，对汉语语言的理解和处理已走向智能化。为了能在各个处理环节中方便和确切地表示汉字，在汉字系统中要涉及到各种汉字代码，包括：汉字输入码、汉字交换码、汉字机内码以及汉字输出码等。

(1) 汉字输入码：汉字输入码是将汉字的不同编码规则通过键盘或其他设备，以编码形式输入计算机。汉字输入计算机可以通过图像识别、汉字语音识别、汉字编码输入和非编码输入等方法。如电报明码、国标区位码、整字输入、字元编码、笔形编码、拼音编码等，可通过标准键盘击键输入。

(2) 汉字机内码：汉字编码输入到计算机系统后，要将其转换成计算机内部表示汉字的机内码，按照程序的要求，控制计算机对机内码进行加工处理。

(3) 汉字交换码：汉字交换码是用于不同的汉字系统之间交换汉字信息的。中华人民共和国国家标准信息交换用汉字字符集 GB 2312—80，共收录一级汉字 3 755 个，二级汉字 3 008 个，各种图形符号 682 个，共计 7 445 个。GB 2312—80 规定每个汉字、图形符号都用两个字节表示，每个字节 7 位二进制代码，与 94 个可打印字符(除空格外)的 ASCII 字符取值范围相同，GB 13000 大字符集则收录了 10 902 个汉字与字符，符合国际肆八位编码字符集。汉字交换码还用区位码。区位码由 4 位数字组成，前两位是区号 01～94，后两位是位号 01～94。区位码是把图形符号分为 94 个区(行)，每个区又分为 94 位(列)，并分别标出区号、位号来表示汉字的编码。当一个汉字在表中的位置确定之后，就能从纵、横坐标上找到它的相应区号与位号，区号、位号连接起来就构成区位码。

(4) 汉字输出码：汉字信息加工处理后的结果如以汉字形式输出，则又应将汉字机内码再转换成汉字交换码或直接变换成汉字地址码，按这些地址从汉字库中取出汉字字形存储码，根据输出设备的要求再转换成字形输出码，供显示或打印。汉字字形存储码是每一个汉字信息的字形