高等学校“十一五”规划教材

数字逻辑

Shuzi Luoji

韩　进　主编

中国矿业大学出版社

China University of Mining and Technology Press

高等学校“十一五”规划教材

数字逻辑

主　编　韩　进
副主编　张秀娟　张　瑞

中国矿业大学出版社

内 容 提 要

本书系统阐述了数制与码制、逻辑代数基础、组合逻辑电路和时序逻辑电路的分析与设计、基本逻辑器件、数字系统设计方法、数字系统的基本算法与逻辑实现，以及 VHDL 语言描述数字系统。

本书可作为计算机类、电子信息类、通讯类各专业本科、专科学生的教材，亦可供相关工程技术人员参考。

图书在版编目(CIP)数据

数字逻辑/韩进主编. —徐州：中国矿业大学出版社，2006.11

ISBN 7-81107-447-8

Ⅰ.数… Ⅱ.韩… Ⅲ.数字逻辑 Ⅳ.TP302.2

中国版本图书馆 CIP 数据核字(2006)第 119473 号

书　　名 数字逻辑
主　　编 韩　进
责任编辑 何　戈
责任校对 杜锦芝
出版发行 中国矿业大学出版社
（江苏省徐州市中国矿业大学内　邮编 221008）
网　　址 http://www.cumtp.com　**E-mail** cumtpvip@cumtp.com
排　　版 中国矿业大学出版社排版中心
印　　刷 徐州中矿大印发科技有限公司
经　　销 新华书店
开　　本 787×1092　1/16　**印张** 14　**字数** 346 千字
版次印次 2006 年 11 月第 1 版　2006 年 11 月第 1 次印刷
定　　价 21.00 元

前　言

人类社会已进入信息时代，信息的处理、存储、传输都依靠电子信息技术、计算机技术、网络通信技术等来完成，而这些技术都是以数字技术为基础的。数字系统和数字设备已广泛应用于各个领域，因而电子信息技术、计算机技术以及相关技术领域的工程师和技术人员必须掌握数字系统的基础知识。

本书是高等学校信息与电气工程“十一五”规划教材，着重介绍数字系统逻辑设计的基本理论和方法，突出基本分析方法和设计方法，其内容吸收了当前数字逻辑设计技术的新发展、新思路，引入了电子设计自动化(EDA)的基本思想，从传统的单纯硬件设计转向硬件软件高度渗透的设计方法。

本书系统地阐述了数制与码制，逻辑代数基础，组合逻辑电路和时序逻辑电路的分析与设计，基本逻辑器件，数字系统设计方法基础，数字系统的基本算法与逻辑实现，以及VHDL语言描述数字系统。

本书可作为计算机科学类、电子信息类、通信类各专业本科、专科学生的教材，同时，也可作为相关专业工程技术人员的参考书。

我们在组织编写教材时，以培养现代化高级人才为重任，以提高学生综合素质、培养学生应用能力和创新能力为目的，以面向现代化、面向世界、面向未来为准绳，注重教材的特色和实用性，反映最新的教学与科研成果，体现本专业的时代特征。

本书由韩进教授和张秀娟教授组织选定内容，列出目录，由张瑞老师编写第一章和第二章，韩进编写第三章、第四章和第六章，张秀娟编写第五章，程勇教授审阅了全部书稿，并提出了许多建议和修改意见。由韩进负责全书的统稿。

本书编写过程中于小鸽、李会平、董建业参加了绘图、编程等许多必要的工作，东野长磊、陈秀霞老师参加了校对工作。由于编者的经验不足、水平有限，书中难免存在不妥乃至错误之处，敬请广大读者不吝赐教。

编　者

2006.6.26 于青岛

目 录

第一章 数制与码制 …… 1
第一节 数制 …… 1
一、十进制 …… 1
二、二进制 …… 2
三、八进制 …… 2
四、十六进制 …… 3
五、数制之间的转换 …… 4
六、八进制、十六进制与二进制之间的相互转换 …… 6
第二节 带符号数的代码表示 …… 6
一、真值与机器数 …… 6
二、原码 …… 6
三、反码 …… 7
四、补码 …… 9
第三节 数的定点和浮点表示 …… 10
一、数的定点表示 …… 10
二、数的浮点表示 …… 11
第四节 编码 …… 11
一、十进制数的二进制编码 …… 11
二、可靠性编码 …… 13
三、字符编码 …… 14
本章小结 …… 15
习题一 …… 16

第二章 逻辑代数基础 …… 17
第一节 逻辑代数的基本概念 …… 17
一、三种基本逻辑运算 …… 18
二、关于正逻辑和负逻辑的概念 …… 20
三、几种复合逻辑运算 …… 21
四、逻辑函数的表示 …… 23
第二节 逻辑代数的公理、基本定理和规则 …… 25
一、公理 …… 25
二、基本定理 …… 25
三、重要规则 …… 26

第三节 逻辑函数表达式的形式与变换 …… 28
一、逻辑函数的基本形式 …… 28
二、逻辑函数的标准形式 …… 28
第四节 逻辑函数化简 …… 33
一、代数化简法 …… 33
二、卡诺图化简法 …… 34
三、逻辑函数化简中的两个实际问题 …… 40
本章小结 …… 43
习题二 …… 43

第三章 集成门电路 …… 46
第一节 晶体管的开关特性 …… 46
一、理想开关 …… 46
二、晶体二极管的开关特性 …… 47
三、晶体三极管的开关特性 …… 48
第二节 简单逻辑门电路 …… 50
一、二极管与门 …… 50
二、二极管或门 …… 51
三、三极管非门 …… 51
第三节 TTL 门电路 …… 52
一、TTL 与非门电路 …… 53
二、三态输出与非门电路(ST 门) …… 54
三、集电极开路的与非门(OC 门) …… 55
第四节 MOS 门电路 …… 57
一、NMOS 门电路 …… 57
二、CMOS 门电路 …… 58
本章小结 …… 59
习题三 …… 60

第四章 组合逻辑电路 …… 62
第一节 概述 …… 62
一、组合逻辑电路的定义 …… 62
二、组合逻辑电路的特点 …… 63
第二节 组合逻辑电路的分析 …… 63
第三节 组合逻辑电路的设计 …… 65
一、单输出组合逻辑电路的设计 …… 66
二、多输出组合逻辑电路的设计 …… 68
三、输入变量中无反变量提供的组合逻辑电路设计 …… 72
第四节 组合逻辑电路的险象 …… 74

一、竞争(race)与冒险(hazard) …… 75
二、险象的分类 …… 76
三、险象的判断 …… 76
四、险象的消除 …… 78
第五节　通用组合逻辑电路 …… 81
一、二进制加法器 …… 81
二、译码器和编码器 …… 85
三、多路选择器和多路分配器 …… 96
本章小结 …… 101
习题四 …… 102

第五章　时序逻辑电路 …… 105
第一节　概述 …… 105
一、时序逻辑电路的结构与类型 …… 105
二、状态表和状态图 …… 107
第二节　触发器及类型转换 …… 109
一、基本 R—S 触发器 …… 109
二、时钟控制 R—S 触发器 …… 111
三、D 触发器 …… 113
四、J—K 触发器 …… 114
五、T 触发器 …… 116
六、各类触发器的转换 …… 117
第三节　同步时序逻辑电路分析 …… 118
一、同步时序逻辑电路的分析方法 …… 119
二、同步时序逻辑电路分析举例 …… 122
第四节　同步时序逻辑电路的设计 …… 126
一、建立原始状态图 …… 126
二、状态简化 …… 130
三、状态编码 …… 138
四、确定激励函数和输出函数 …… 140
五、画逻辑电路图 …… 141
第五节　同步时序逻辑电路设计举例 …… 142
第六节　通用时序逻辑电路 …… 149
一、计数器 …… 149
二、寄存器 …… 159
三、顺序脉冲发生器 …… 162
四、序列信号发生器 …… 163
本章小结 …… 167
习题五 …… 167

第六章 面向可编程逻辑器件的数字系统设计 …… 172
第一节 数字系统设计方法概述 …… 172
一、传统的系统硬件设计方法 …… 172
二、基于硬件描述语言 HDL 的硬件设计方法 …… 173
第二节 FPGA/CPLD 的结构与工作原理 …… 174
一、简单 PLD …… 175
二、CPLD …… 180
三、FPGA …… 180
第三节 基于 FPGA/CPLD 的数字系统设计 …… 184
一、FPGA/CPLD 设计流程 …… 184
二、MAX+PLUSⅡ概述 …… 185
第四节 原理图输入设计方法 …… 186
一、项目建立 …… 186
二、输入设计项目、存盘 …… 186
三、项目编译 …… 188
四、时序仿真 …… 188
五、管脚锁定 …… 193
六、编程下载 …… 194
第五节 VHDL 设计 …… 195
一、VHDL 的特点 …… 195
二、VHDL 的描述风格 …… 196
三、VHDL 的基本结构 …… 196
第六节 常用逻辑电路 VHDL 描述 …… 198
一、半加器 …… 199
二、2—4 译码器 …… 199
三、编码器 …… 201
四、多路选择器 …… 202
五、数据分配器 …… 202
六、奇偶校验电路 …… 203
七、三态门 …… 205
八、计数器 …… 206
九、寄存器 …… 207
十、四位二进制全加器 …… 210
本章小结 …… 213
习题六 …… 213

参考文献 …… 215

第一章　数制与码制

【本章重点】 数字系统中数的各种基本表示方法，包括各种不同的进位计数体制及其相互之间的转换；带符号数的代码表示方法及其运算；计算机中常用的数码与字符的代码表示，如十进制常用编码(8421码、2421码、余3码)以及可靠性编码。

【本章难点】 数制之间的相互转换；原码、反码、补码的运算；可靠性编码的编码技术。

第一节　数　　制

数不是客观存在的，而是人们对客观事物数量关系的一种抽象。研究数字的任务就是研究被抽象了的数之间的关系。为了确定数之间的关系，人们提出了各种不同的计数方法。常见的有进位计数法和非进位计数法。进位计数法又称为按“权”计数；非进位计数法无固定权值。“权”为基数(radix)的整数次幂。

数制(number system)是进位计数体制的简称。进位计数体制即按进位方式实现计数的一种规则，是用统一的符号和规则表示数的一种方法。在日常生活中我们就是按进位计数制来计数的，如十进制数。对于任何一个数，我们可以用不同的进位体制来表示，但不同进位计数制的运算方法和难易程度各不相同，这对数字系统的性能有很大的影响。

一、十进制

以10为基数的进位计数制为十进制(decimal)。在十进制中，每一位用0～9共10个数码来表示，所以计数基数是10。超过9的数则需用多位数表示，其中低位数和相邻高位数之间的关系是逢十进一，故称为十进制，常用$(N)_{10}$或$(N)_D$表示。例如$(3456)_{10}$是一个4位十进制数，个位数6处于第0位，表示的数值大小为6×10^0，千位数3处于第3位，表示的数值为3×10^3，其他各位的含义如图1-1(a)所示。

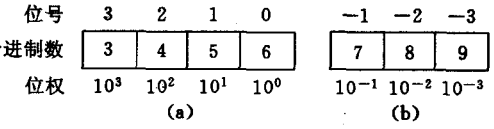

图1-1　各位数字的含义

由此

$$(3456)_{10}=3\times10^3+4\times10^2+5\times10^1+6\times10^0$$

其中10^3、10^2、10^1、10^0依次称为第3、2、1、0位的权，进而，第i位的权为10^i。

$(0.789)_{10}$是3位十进制小数。图1-1(b)给出了它的权，由此

$$(0.789)_{10}=7\times10^{-1}+8\times10^{-2}+9\times10^{-3}$$

基数和权是进位计数制的两个基本要素，根据基数和权的概念，对于任何一个由n位整数、m位小数组成的十进制数N，可以用按位计数法表示如下：

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

也可以用按权展开式表示如下：

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

二、二进制

以 2 为基数的进位计数制为二进制(binary)。在二进制中，每一位用 0、1 共 2 个数码来表示，所以计数基数是 2，超过 1 的数则需用多位数表示，其中低位数和相邻高位数之间的关系是逢二进一，故称为二进制。常用$(N)_2$或$(N)_B$表示。对于任何一个由 n 位整数、m 位小数组成的二进制数，有：

$$\begin{aligned}(N)_2&=(r_{n-1}r_{n-2}\cdots r_1r_0.r_{-1}r_{-2}\cdots r_{-m})_2\\&=r_{n-1}\times2^{n-1}+r_{n-2}\times2^{n-2}+\cdots+r_1\times2^1+r_0\times2^0+r_{-1}\times2^{-1}+\\&\quad r_{-2}\times2^{-2}+\cdots+r_{-m}\times2^{-m}\\&=\sum_{i=-m}^{n-1}r_i\times2^i\end{aligned}$$

如：$(1001.01)_2=1\times2^3+0\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}$。

在数字系统中，常用二进制来表示数和进行数学运算，这是因为二进制具有如下特点：

(1) 二进制只有 0 和 1 两个数字符号，容易用物理状态来表示，任何具有两个不同稳定状态的元件都可以用来表示 1 位二进制数，例如：晶体管的导通和截止，脉冲的“有”和“无”等。

(2) 二进制运算规则简单，便于进行算术运算。

(3) 二进制还可实现逻辑运算，从而可以用布尔代数对数字系统进行分析、综合，便于逻辑电路的设计优化。

(4) 二进制只有两个状态，数字的传输和处理不容易出错，可靠性高。

(5) 采用二进制来表示数可以节省设备。

三、八进制

以 8 为基数的进位计数制为八进制(octal)。在八进制中，每一位用 0～7 共 8 个数码来表示，所以计数基数是 8，超过 7 的数则需用多位数表示，其中低位数和相邻高位数之间的关系是逢八进一，故称为八进制，常用$(N)_8$或$(N)_O$表示。对于任何一个由 n 位整数、m 位小数组成的 8 进制数，有：

$$\begin{aligned}(N)_8&=(r_{n-1}r_{n-2}\cdots r_1r_0.r_{-1}r_{-2}\cdots r_{-m})_8\\&=r_{n-1}\times8^{n-1}+r_{n-2}\times8^{n-2}+\cdots+r_1\times8^1+r_0\times8^0+r_{-1}\times8^{-1}+\\&\quad r_{-2}\times8^{-2}+\cdots+r_{-m}\times8^{-m}\\&=\sum_{i=-m}^{n-1}r_i\times8^i\end{aligned}$$

这里该数制的基数为 8，8^i 为八进制数第 i 位的权。

因为二进制仅用两个符号，所以对于数字系统中数据存储和处理有很大的使用价值，但使用二进制表示给定的数比相应的十进制数有更长的序列组，如果要人工输入数据，仅需要一个 2 键的键盘，但这些键将被敲击多次。经常采用成组处理二进制数来解决数据输入问题。八进制数根据表 1-1 使用 3 位一组的数。

表 1-1　　二进制数与八进制数的对应关系

二进制	000	001	010	011	100	101	110	111
八进制	0	1	2	3	4	5	6	7

每一个八进制符号代表与 3 位一组的二进制数相同的数字，这 8 个符号组成了基数为 8 的数制。这种情况下数据输入需要一个 8 键的键盘，但敲击次数仅是二进制键盘的 1/3。所以在书写计算机程序时，经常使用八进制。

四、十六进制

以 16 为基数的进位计数制为十六进制(hexadecimal)。在十六进制中，每一位用 0～9、A～F (A、B、C、D、E、F 依次表示 10、11、12、13、14、15)16 个数码来表示，所以计数基数是 16，超过 F 的数则需用多位数表示，其中低位数和相邻高位数之间的关系是逢十六进一，故称为十六进制，常用$(N)_{16}$或$(N)_{\mathrm{H}}$表示。对于任何一个由 n 位整数、m 位小数组成的十六进制数，有：

$$\begin{aligned}(N)_{16} &= (r_{n-1}r_{n-2}\cdots r_1 r_0 . r_{-1}r_{-2}\cdots r_{-m})_{16}\\ &= r_{n-1}\times 16^{n-1}+r_{n-2}\times 16^{n-2}+\cdots+r_1\times 16^1+r_0\times 16^0+r_{-1}\times 16^{-1}+\\ &\quad r_{-2}\times 16^{-2}+\cdots+r_{-m}\times 16^{-m}\\ &= \sum_{i=-m}^{n-1} r_i\times 16^i\end{aligned}$$

这里该数制的基数为 16，16^i 为十六进制数第 i 位的权。

十六进制计数法将成组的思想扩充到 4 位，位组与相应的十六进制符号如表 1-2 所示。

表 1-2　　位组和相应的十六进制符号

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

十六进制的符号 0 到 9 与最前面的 10 个 4 位二进制数组的十进制数的值是相等的。因为没有单个十进制数码能表示比 9 大的数，表示最后 6 组我们需要新符号，用字母 A、B、C、D、E、F 6 个字母表示。十六进制数需要 16 键的键盘，但敲击率仅是二进制键盘的1/4。所以用十六进制符号书写程序就十分方便，使得十六进制的应用比八进制的应用还要广泛。

以上分别介绍了十进制、二进制、八进制和十六进制，如果某数制含有 R 个数值符号和

小数点，且逢 R 进一，则称为 R 进制。对于任何一个由 n 位整数、m 位小数组成的 R 进制数，有：

$$\begin{aligned}(N)_R &= (r_{n-1}r_{n-2}\cdots r_1 r_0 . r_{-1}r_{-2}\cdots r_{-m})_R \\ &= r_{n-1}\times R^{n-1}+r_{n-2}\times R^{n-2}+\cdots+r_1\times R^1+r_0\times R^0+r_{-1}\times R^{-1}+ \\ &\quad r_{-2}\times R^{-2}+\cdots+r_{-m}\times R^{-m} \\ &= \sum_{i=-m}^{n-1} r_i\times R^i\end{aligned}$$

这里 R 是大于 1 的任意正整数，称为该数制的基数，R^i 为 R 进制数第 i 位的权。

五、数制之间的转换

（一）非十进制——十进制转换

把 R 进制数转换成等值的十进制数，转换的方法是将 R 进制数按权展开，再把各项数值以十进制形式相加，所得的和即为等值的十进制数。

1. 二—十进制转换

例 1-1 将二进制数$(10110.101)_2$转换成十进制数。

按权展开得：

$$\begin{aligned}(10110.101)_2 &= 1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3} \\ &= (22.625)_{10}\end{aligned}$$

2. 八—十进制转换

例 1-2 将八进制数$(127.53)_8$转换成十进制数。

按权展开得：

$$\begin{aligned}(127.53)_8 &= 1\times8^2+2\times8^1+7\times8^0+5\times8^{-1}+3\times8^{-2} \\ &= (87.671875)_{10}\end{aligned}$$

3. 十六—十进制转换

例 1-3 将十六进制数$(5C1.0B)_{16}$转换成十进制数。

按权展开得：

$$\begin{aligned}(5C1.0B)_{16} &= 5\times16^2+12\times16^1+1\times16^0+0\times16^{-1}+11\times16^{-2} \\ &= (1473.04296875)_{10}\end{aligned}$$

（二）十进制—非十进制转换

由十进制数向二进制数的转换分为两步，即把十进制数的整数部分和小数部分分别进行转换，再把各自转换的结果合并成最后的结果。

1. 整数部分

若要将十进制整数$(N)_{10}$转换成等值二进制数$(K_{n-1}K_{n-2}\cdots K_1K_0)_2$，则可写成下列等式：

$$\begin{aligned}(N)_{10} &= K_{n-1}\times2^{n-1}+K_{n-2}\times2^{n-2}+\cdots+K_1\times2^1+K_0\times2^0 \\ &= 2\times(K_{n-1}\times2^{n-2}+K_{n-2}\times2^{n-3}+\cdots+K_2\times2^1+K_1)+K_0\end{aligned}$$

将上式两边均除以 2，可得商为$(K_{n-1}\times2^{n-2}+K_{n-2}\times2^{n-3}+\cdots+K_2\times2^1+K_1)$，余数为 K_0。将商再除以 2，得新余数 K_1，依此类推，继续用 2 除，直到最后的商为零，就可以分别求出 $K_0, K_1, K_2, \cdots, K_{n-1}$。

由以上分析可知，其转换方法就是将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 作为二进制的最低位，把得到的商再除以 2，再取余数 1 或 0 作为二进制的次高位，依此类推，直到商为 0，所得余数为最高位。把全部余数按最高位到最低位次序排列，就得到等值的二进制数。

例 1-4　将十进制数$(21)_{10}$转换成二进制数。

解

2	21	余数	
2	10	1	（最低位）
2	5	0	
2	2	1	
2	1	0	
	0	1	（最高位）

即：
$$(21)_{10}=(10101)_2$$

2. 小数部分

若要将十进制小数$(N)_{10}$转换成等值二进制数$(0.K_{-1}K_{-2}\cdots K_{-m})_2$，则可写成下列等式：

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

将上式两边均乘以 2，可得$2(N)_{10}=K_{-1}+(K_{-2}\times 2^{-1}+K_{-3}\times 2^{-2}+\cdots+K_{-m}\times 2^{-m+1})$，整数为$K_{-1}$。将小数部分再乘以 2，得新整数$K_{-2}$，依此类推，继续用 2 乘，就可以分别求出$K_{-3},K_{-4},K_{-5},\cdots,K_{-m}$。

由以上分析可知，十进制数的小数部分采用“乘 2 取整”法进行转换，即将十进制小数逐次乘以 2（每次只将小数部分乘以 2），并依次记下整数，直到积的小数部分出现全 0，或小数部分虽不为 0，但二进制纯小数的位数或精度已达到预定的要求。第一次得到的“积的整数”为最高位，最后一次得到的“积的整数”为最低位。把全部整数按次序排列，就得到等值的二进制数。

例 1-5　将十进制小数$(0.625)_{10}$转换成二进制小数：

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

小数部分已全为 0，得：

$$(0.625)_{10}=(0.101)_2$$

由上可见，若把$(21.625)_{10}$转换成二进制数，其结果是：

$$(21.625)_{10}=(10101.101)_2$$

有时乘 2 取整的小数部分总不等于零，就产生转换误差，此时可根据所需读取二进制的位数确定。如把$(0.71)_{10}$转换成 6 位二进制小数为$(0.101101)_2$。

由十进制转换成八进制的方法是，整数部分“除 8 取余”，小数部分“乘 8 取整”。

由十进制转换成十六进制整数部分“除 16 取余”，小数部分“乘 16 取整”。

六、八进制、十六进制与二进制之间的相互转换

前面已经介绍过一位八进制数可以用 3 位二进制数来表示，一位十六进制数可以用 4 位二进制数来表示，因此，二进制转换成八进制（或十六进制）时，二进制的整数部分从低位开始，每三位（或四位）分为一组，若最左边一组不足三位（或四位），可在左边添 0 补足。二进制小数部分从小数点向右每三位（或四位）分为一组，最后不足三位（或四位），可在右边添 0 补足。然后，将每组二进制数转换为八进制（或十六进制）数。采用上述的逆过程，可将八进制（或十六进制）转换成二进制。例如：

八进制	2		5		7		0		5	5	4
二进制	010		101		111		000		101	101	100
十六进制		A		F				1	6	C	

八进制和十六进制之间的转换可用二进制作桥梁。

第二节 带符号数的代码表示

一、真值与机器数

算术运算中的数是带符号的数，例如 +25 和 −25，由两部分组成：符号和绝对值。但机器只能识别二进制数，有鉴于此，常用二进制数码的最高位来表示符号，称为符号位，且 0 表示正，1 表示负，其余各位用来表示绝对值，并称为数值位，构成带符号的二进制数。如：

$$N_1 = +1001 \qquad N_2 = -1001$$

在机器中分别表示为：

这种将数值部分及符号部分统一用代码表示的带符号数称为机器数，如 N_1、N_2 的机器数表示形式就是 01001、11001。而把原来的数值形式称为机器数的真值，如 N_1、N_2 的真值表示形式是 +1001、−1001。

由前面介绍的二进制加、减、乘、除运算可知，乘法实际是移位相加，而除法是移位相减，即在机器中只需要做加、减运算。但减法运算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，最后的运算结果和绝对值大的符号一致。虽然逻辑电路可以实现减法运算，但电路复杂，运算时间长。为了能使减法转换为加法运算，人们提出了 3 种机器数的表示形式，即原码、反码和补码。

二、原码

原码表示方法是将符号位用数码 0 表示正号，用 1 表示负号，而对数值位不做任何改变，仍然采用原来的二进制数表示。

例如，两个带符号的二进制数 $N_1 = +1001101$ 及 $N_2 = -1001101$ 的原码表示形式为：

$$[N_1]_{原}=01001101 \qquad [N_2]_{原}=11001101$$

用数学表达式来描述原码的形成规则是：

一个 n 位的整数 N（包含一位符号位）的原码一般表示式为：

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

如 $N=-111$，则 $[N]_{原}=2^{4-1}-N=1000-(-111)=1111$。

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：

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

如 $N=-0.101$，则 $[N]_{原}=1-N=1-(-0.101)=1.101$。

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：

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

原码的运算：

原码中的符号位仅仅用来表示数的正、负，不参加运算，进行运算的只是数值部分。两数相加时，如果同号，则数值相加，符号不变；如果异号，就要进行减法，而在相减时，须先比较两数绝对值的大小，然后从绝对值较大的数中减去绝对值较小的数，差值的符号与绝对值较大的数的符号一致。

例 1-6　试利用原码求 26－21，设字长为 8 位。

解　26 的绝对值大于 21 的绝对值，所以 26－21 的结果中符号位为正。

$$\begin{array}{rr} & 00011010\\ - & 00010101\\ \hline & 00000101\end{array}$$

故 $[26-21]_{原}=00000101$，其真值 26－21＝5。

例 1-7　试利用原码求 21－26，设字长为 8 位。

解　21 的绝对值小于 26 的绝对值，所以 21－26 的结果中符号位为负。

由例 1-6 知 $[26-21]=00000101$，故 $[21-26]_{原}=10000101$，其真值为 21－26＝－5。

可见，采用原码运算，为了判断是同号还是异号、比较数的绝对值的大小等，就要增加机器中的设备和降低运算速度。为了解决这一问题，采用补码表示法。

三、反码

反码又称为“对 1 的补数”。用反码表示时，对于正数，反码和原码相同；对于负数，反码就是将原码的数值位按位取反，即“1”变“0”，“0”变“1”，而符号位为 1。

例如，两个带符号的二进制数 $N_1=+1001101$ 及 $N_2=-1001101$ 的反码表示形式为：

$$[N_1]_{反}=01001101 \qquad [N_2]_{反}=10110010$$

用数学表达式来描述反码的形成规则是：

一个 n 位的整数 N（包含一位符号位）的反码一般表示式为：

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

若 $N=-1001$，则 $[N]_{反}=2^5-1+N=100000-1+(-1001)=10110$。

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：

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

如 $N=-0.1001$，则 $[N]_{反}=2-2^{-4}+N=10-0.0001+(-0.1001)=1.0110$。

从反码的一般表示式中可以看出：

(1) 当 N 为正数时，$[N]_{反}$ 与 $[N]_{原}$ 相同。

(2) 当 N 为负数时，$[N]_{反}$ 符号位为 1，数值部分是将原码数值位按位取反。

(3) 在反码表示中，有两种不同形式的 0，即：

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

反码的运算：

设 A 和 B 依次为被加数（或被减数）和加数（或减数）。运算规则是：

$$[A+B]_{反}=[A]_{反}+[B]_{反}$$
$$[A-B]_{反}=[A]_{反}+[-B]_{反}$$

用反码实现加/减运算的步骤如下：

第一步　把 A 与 B（减法运算时为 $-B$）均表示成反码形式；

第二步　两个反码相加/减，且把符号位也看成二进制数的最高位参与运算；

第三步　若结果中最高位有进位，则将该进位与和数的最低位再相加（称为循环进位）。运算结果仍为反码。

例 1-8　试利用反码求 26−21，设字长为 8 位。

$$[26]_{反}=00011010$$
$$[-21]_{反}=11101010$$

且 $[26]_{反}+[-21]_{反}$ 为：

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

循环进位

故

$$[26-21]_{反}=[26]_{反}+[-21]_{反}=00000101$$

即

$$26-21=5$$

例 1-9　试利用反码求 21−26，设字长为 8 位。

$$[21]_{反}=00010101$$

$$[-26]_{反}=11100101$$

且$[21]_{反}+[-26]_{反}$为：

$$\begin{array}{r} 00010101 \\ +\ 11100101 \\ \hline 11111010 \end{array}$$

故：$$[21-26]_{反}=[21]_{反}+[-26]_{反}=11111010$$

即：$$21-26=-5$$

四、补码

补码又称为“对二的补数”。用补码表示二进制数时，正数的表示跟原码和反码的表示相同。负数的补码可从原码转换而来，转换规则为：符号位仍为1，数值部分按位取反，然后在最低有效位上加1。例如，两个带符号的二进制数 $N_1=+1001101$ 及 $N_2=-1001101$ 的补码表示形式为：

$$[N_1]_{补}=01001101 \qquad [N_2]_{补}=10110011$$

用数学表达式来描述补码的形成规则是：

一个 n 位的整数 N（包含一位符号位）的补码的一般表示式为：

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

若 $N=-1001$，则$[N]=2^5+N=100000+(-1001)=10111$。

对于定点小数，它的补码一般表示式为：

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

若 $N=-0.1001$，则$[N]_{补}=2+N=10+(-0.1001)=1.0111$。

从补码的一般表示式中可以看出：

（1）当 N 为正数时，$[N]_{补}$ 与 $[N]_{原}$ 相同。

（2）当 N 为负数时，$[N]_{补}$ 符号位为1，数值部分是将原码数值位按位取反然后加1。

（3）在补码表示中，0的表示式是惟一的，即：

$$[+0]_{补}=0.00\cdots0$$

$$[-0]_{补}=0.00\cdots0$$

补码的运算：

设 A 和 B 依次为被加数（或被减数）和加数（或减数）。运算规则是：

$$[A+B]_{补}=[A]_{补}+[B]_{补}$$

$$[A-B]_{补}=[A]_{补}+[-B]_{补}$$

运算步骤与反码相似，但不进行循环移位，即进位自动丢失。

例1-10　试利用补码求26－21，设字长为8位。

$$[26]_{补}=00011010$$

$$[-21]_{补}=11101011$$

且$[26]_{补}+[-21]_{补}$为：