HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY PRESS

微型计算机原理及应用教程

WEIXING JISUANJI YUANLI JI YINGYONGJIAOCHENG

杨杰 刘清 张中洲 编

华中理工大学出版社

.1
1-J

微型计算机原理及应用教程

杨 杰 刘 清 张中洲 编

华中理工大学出版社

图书在版编目(CIP)数据

微型计算机原理及应用教程/杨杰等编

武汉:华中理工大学出版社,1998年8月

ISBN 7-5609-1792-5

Ⅰ.单…

Ⅱ.①杨… ②刘… ③张…

Ⅲ.单片计算机-基础知识-高等学校-教材

Ⅳ.TP·235

微型计算机原理及应用教程

杨 杰 刘 清 张中洲 编

责任编辑:徐祖兴 叶翠华

*

华中理工大学出版社出版发行

(武昌喻家山 邮编:430074)

新华书店湖北发行所经销

武汉市长江印刷厂印刷

*

开本:787×1092 1/16 印张:17.5 字数:448 000

1998年8月第1版 1998年8月第1次印刷

印数:1—3 000

ISBN 7-5609-1792-5/TP·292

定价:21.00元

(本书如有印装质量问题,请向出版社发行部调换)

内 容 提 要

本书全面地阐述了 MCS－51 和 MCS－96 系列中的 8051 及 8098 单片微型计算机的基本原理和应用技术。全书分为两篇，第一篇系统地论述了微型计算机的组成原理、工作过程、程序设计、系统扩展、应用系统设计；第二篇详细介绍了 8098 单片机原理、软件设计、接口技术及其在工业控制中的应用实例。

本书为大专院校非计算机专业微机原理课的教材，也可供计算机工程技术人员学习参考。

前　言

计算机的发展有两大趋向：一是巨型机，即向超高速、大容量、实时和智能化方向发展；二是向微型化（单片微控制器）、低功耗、低价格方向发展。单片机属于后者。我们以目前广泛应用的 INTEL8051 单片机为典型机，介绍了微型计算机的硬件、软件基础知识和应用方法。考虑 16 位单片机应用的不断深入，本书还详细地介绍了 INTEL8098 单片机的基本结构、指令系统、接口和应用方法。

本书分为两篇，第一篇的第一、二章详细地介绍了微型计算机的基础知识和基本概念，是全书的基础。第三章至第六章以 8051 单片机为例，介绍了单片微型计算机的硬件结构、特点、指令系统、扩展方法及一些外围芯片；第二篇着重介绍 8098 单片机硬件结构、软件编程和接口，以及几个成功的单片机应用实例。全书内容由浅入深，简单明了，自成系统，适用面广。

全书由杨杰、刘清、张中洲编写，杨杰编写第一篇的第七章和第二篇的第八，十，十一，十二，十三，十四章，刘清编写第一篇的第一、二、四、五章，张中洲编写第一篇第三章、第六章和第二篇第九章，最后由杨杰统稿。武汉水利电力大学姜明启副教授对本书进行了审阅，书中插图由龚昌奇副教授描绘。

由于作者学识水平有限，成稿仓促，书中的错误和疏忽在所难免，肯请读者批评指正。

编　者

1998 年 2 月

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第一篇　MCS－51单片微型计算机原理及应用

第一章　微型计算机中的数制和码制 …… (1)
§1.1　数和数制 …… (1)
§1.2　二进制中带符号数的表示方法及运算 …… (7)
§1.3　数的小数点表示法及计算机常用编码 …… (14)
习题和思考题 …… (19)
第二章　微型计算机基础 …… (20)
§2.1　微型计算机的组成 …… (20)
§2.2　半导体存储器 …… (23)
§2.3　微型计算机的工作过程 …… (35)
§2.4　单片机及其发展与应用 …… (37)
小结 …… (40)
习题和思考题 …… (42)
第三章　MCS－51单片机结构 …… (43)
§3.1　MCS－51系列单片机概述 …… (43)
§3.2　8051的基本结构 …… (44)
§3.3　中央处理器(CPU) …… (46)
§3.4　并行I/O端口 …… (49)
§3.5　存储器结构 …… (51)
小结 …… (55)
习题和思考题 …… (56)
第四章　指令系统及汇编语言程序设计 …… (57)
§4.1　概述 …… (57)
§4.2　寻址方式 …… (59)
§4.3　指令系统 …… (60)
§4.4　汇编语言基本概念 …… (68)
§4.5　汇编语言程序设计 …… (72)
小结 …… (89)
习题和思考题 …… (90)
第五章　MCS－51单片机定时/计数器、串行口及中断系统 …… (92)
§5.1　定时器 …… (92)
§5.2　串行口 …… (96)
§5.3　中断系统 …… (104)
小结 …… (112)
习题和思考题 …… (114)

第六章　常用接口芯片及系统的扩展 …… (116)
§6.1　可编程的并行输入/输出接口芯片 8255 …… (116)
§6.2　8155 RAM/IO/CTC 扩展器 …… (128)
§6.3　8253 可编程计数/定时器 …… (137)
§6.4　8279 可编程键盘/显示器接口 …… (142)
§6.5　扩展存储器 …… (149)
小结 …… (155)
习题和思考题 …… (155)
第七章　MCS－51 应用系统设计 …… (157)
§7.1　步进电机与单片机的接口及应用 …… (157)
§7.2　V/F 转换器与 MCS－51 系列单片机的接口 …… (162)
§7.3　单片机在浆纱测控系统中的应用 …… (166)
§7.4　采用单片机技术的电线电缆计长仪 …… (174)

第二篇　8098 单片机

第八章　8098 结构综述 …… (178)
§8.1　MCS－96 系列单片机概述 …… (178)
§8.2　8098 单片机的基本结构 …… (181)
§8.3　中央处理器(CPU) …… (183)
§8.4　存储器空间 …… (184)
§8.5　I/O 口 …… (189)
§8.6　复位与掉电 …… (191)
小结 …… (193)
第九章　8098 指令系统 …… (194)
§9.1　操作数类型和寻址方式 …… (194)
§9.2　8098 指令系统 …… (197)
小结 …… (204)
第十章　中断系统和定时器 …… (205)
§10.1　中断系统 …… (205)
§10.2　定时器 …… (213)
小结 …… (218)
第十一章　高速输入 HSI 和高速输出 HSO …… (219)
§11.1　高速输入 HSI …… (219)
§11.2　高速输出 HSO …… (223)
小结 …… (228)
第十二章　串行口、A/D 转换器和脉宽调制器 PWM …… (229)
§12.1　串行口 …… (229)
§12.2　A/D 转换器 …… (233)
§12.3　脉冲宽度调制器 PWM …… (236)
小结 …… (239)
第十三章　8098 硬件接口设计 …… (240)
§13.1　8098 与 ROM、RAM 的连接 …… (240)
§13.2　8098 单片机的 I/O 接口扩展 …… (242)

§13.3 通用用户系统板 …… (244)
小结 …… (245)
第十四章 8098单片机的应用 …… (246)
§14.1 8098单片机应用于交流电气参数测试 …… (246)
§14.2 IBM－PC/XT微型计算机与多个8098单片机的串行通讯 …… (250)
§14.3 高精度光电耦合放大器在8098测控系统中的应用 …… (254)
附录 …… (260)
一、MCS－51系列单片机指令速查表 …… (260)
二、8098单片机指令速查表 …… (264)
参考文献 …… (270)

第一篇　MCS－51单片微型计算机原理及应用

第一章　微型计算机中的数制和码制

数是客观事物的量在人们头脑中的反映。一个数可以用不同的计数制度来表示它的大小，虽然形式不同，但其“量”相等。

用一串数字和一串符号表示某个数时，这串数字或符号就是这个数的码或编码。表达一个数的大小和正负的不同方法称为码制。

§1.1　数和数制

一、数的位置表示法及各种进位制

用一组数字（或符号）表示数时，若每个数字表示的量不但决定于数字本身，且决定于它的位置，就称为位置表示法。在位置表示法中，对每一数位赋以一定的位值称为权。每个数位上的数字所表示的量是这个数字和权的乘积。如果相邻两位中高位的权与低位的权之比是常数，则此常数称为基数，用X表示，则数 $a_{n-1},a_{n-2},a_{n-3},\cdots,a_0,a_{-1},a_{-2},\cdots,a_{-(m-1)},a_{-m}$ 所表示的量N为

$$N = a_{n-1}X^{n-1} + a_{n-2}X^{n-2} + a_{n-3}X^{n-3} + \cdots + a_0X^0 + a_{-1}X^{-1} + \cdots + a_{-(m-1)}X^{-(m-1)} + a_{-m}X^{-m} = \sum_{i=-m}^{n-1} a_iX^i \qquad (1.1)$$

式中，a_i 为系数（或编码），范围为0～X－1的整数；m、n为幂指数，为正整数；X为基数，X的不同取值就得到不同进位制数的表达式。

当X＝10时，得十进制数的表达式为

$$(N)_{10} = \sum_{i=-m}^{n-1} a_i10^i$$

其特点是：系数 a_i 只能在0～9这十个数字中取值；每个数位上的权是10的某次幂；在加减运算中，采用“逢十进一”“借一当十”的规则。例如：

$$(1392.67)_{10} = 1\times10^3 + 3\times10^2 + 9\times10^1 + 2\times10^0 + 6\times10^{-1} + 7\times10^{-2}$$

十进制计数制是人们日常生活中最常用的一种计数制。

当X＝2时，得二进制数的表达式为

$$(N)_2 = \sum_{i=-m}^{n-1} a_i2^i$$

其特点是：系数 a_i 只能在0和1这两个数字中取值；每个数位上的权是2的某次幂；在加减运算中，采用“逢二进一”和“借一当二”的规则。例如：

$$(10111.011)_2 = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}$$

二进制计数制中，各数位上的系数只有 0 和 1 两种取值，用电路实现时最为方便，因而它是计算机内部采用的计数制。

当 X=8 时，得八进制数的表达式为

$$(N)_8 = \sum_{i=-m}^{n-1} a_i 8^i$$

其特点是：系数 a_i 只能在 0～7 这 8 个数字中取值；每个数位上的权是 8 的某次幂；在加减法运算中，采用“逢八进一”和“借一当八”的规则。例如：

$$(137.56)_8 = 1\times 8^2 + 3\times 8^1 + 7\times 8^0 + 5\times 8^{-1} + 6\times 8^{-2}$$

同理，当 X=16 时，得十六进制数的表达式为

$$(N)_{16} = \sum_{i=-m}^{n-1} a_i 16^i$$

其特点是：系数 a_i 只能在 0～15 这 16 个数字中取值（其中 0～9 这 10 个数字借用十进制中的数码，10～15 这 6 个数可用 A、B、C、D、E、F 表示）；每个数位上的权是 16 的某次幂；加减运算中，采用“逢十六进一”和“借一当十六”的规则。例如：

$$(32AF.EB)_{16} = 3\times 16^3 + 2\times 16^2 + 10\times 16^1 + 15\times 16^0 + 14\times 16^{-1} + 11\times 16^{-2}$$

八进制数和十六进制数常常用来书写计算机程序。

表 1.1 列出了 17 个数的四种进位制中数的表示法，其中尾标 B 是 Binary 的缩写，表示二进制数，Q 表示八进制数（Octal 的缩写为“O”，为区别于数字“0”写为 Q），H 是 Hexadecimal 的缩写，表示该数是十六进制数，十进制数尾标 D 可省去。

表 1.1 十进制、二进制、八进制、十六进制数码对照表

十进制	二进制	八进制	十六进制	十进制	二进制	八进制	十六进制
0	0000B	0Q	0H	9	1001B	11Q	9H
1	0001B	1Q	1H	10	1010B	12Q	AH
2	0010B	2Q	2H	11	1011B	13Q	BH
3	0011B	3Q	3H	12	1100B	14Q	CH
4	0100B	4Q	4H	13	1101B	15Q	DH
5	0101B	5Q	5H	14	1110B	16Q	EH
6	0110B	6Q	6H	15	1111B	17Q	FH
7	0111B	7Q	7H	16	10000B	20Q	10H
8	1000B	10Q	8H				

二、各种进位制数间的转换

1. 任意进制数与十进制数间的相互转换

(1)任意进制数转换成十进制数：这种转换的最简单的方法是根据任意进位制数的表达式按权展开后相加即可。例如：

$$1011.110B = 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2} + 0\times 2^{-3}$$
$$= 11.75$$

$$732.14Q = 7\times 8^2 + 3\times 8^1 + 2\times 8^0 + 1\times 8^{-1} + 4\times 8^{-2}$$
$$= 474.1875$$

$$3AF.E6H = 3\times 16^2 + 10\times 16^1 + 15\times 16^0 + 14\times 16^{-1} + 6\times 16^{-2}$$
$$= 943.8984375$$

也可按下列方法将任意进位制数的整数部分和小数部分分开转换。

1)整数部分转换：设 N 为 n 位任意进位制整数，则由(1.1)式有

$$N = a_{n-1}X^{n-1} + a_{n-2}X^{n-2} + \cdots + a_1X^1 + a_0X^0$$
$$= \{\cdots\{[(a_{n-1}X + a_{n-2})X + a_{n-3}]X + a_{n-4}\}X \cdots + a_1\}X + a_0$$

由此得转换步骤为：先将最高位乘以基数 X，加上次高位，令其结果为 Y_1；再将 Y_1 乘以 X，加上第三位，结果为 Y_2；又将 Y_2 乘以 X，加上第四位，结果为 Y_3。如此进行下去直至加上最低位为止，便得到所要求的任意进制结果。如上例中 3 个整数部分转换如下：

$$1011B = [(1 \times 2 + 0) \times 2 + 1] \times 2 + 1 = 11$$
$$732Q = (7 \times 8 + 3) \times 8 + 2 = 474$$
$$3AFH = (3 \times 16 + 10) \times 16 + 15 = 943$$

2)小数部分转换：设 N 是 m 位的任意进位制纯小数，即同样有

$$N = a_{-1}X^{-1} + a_{-2}X^{-2} + \cdots + a_{-m+1}X^{-m+1} + a_{-m}X^{-m}$$
$$= X^{-1}\{a_{-1} + X^{-1}[a_{-2} + X^{-1}(a_{-3} + \cdots + X^{-1}(a_{-m+1} + X^{-1}a_{-m}))]\}$$

由此得转换步骤为：先将最低位除以 X，加次低位，结果为 R_1；再将 R_1 除以 X，加上第三位，结果为 R_2。如此进行下去直至加上最高位后被 X 除为止，便得到所要求的任意进位制结果。对上例中 3 个小数部分转换如下：

$$0.110B = 2^{-1}[1 + 2^{-1}(1 + 2^{-1} \times 0)] = 0.75$$
$$0.14Q = 8^{-1}(1 + 8^{-1} \times 4) = 0.1875$$
$$0.E6H = 16^{-1}(14 + 16^{-1} \times 6) = 0.8984375$$

显然，两种方法得到相同转换结果。

(2)十进制整数转换为任意进制整数：设 N 是要转换的十进制整数，它相应的任意进位制整数共有 n 位，即

$$N = a_{n-1}X^{n-1} + a_{n-2}X^{n-2} + \cdots + a_1X^1 + a_0X^0$$

等式两边同除以基数 X，得商 Q_1 和余数(都为整数)，即

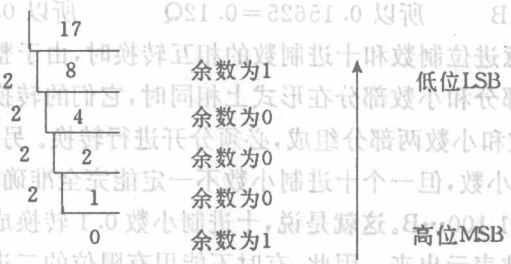
$$\frac{N}{X} = a_{n-1}X^{n-2} + a_{n-2}X^{n-3} + \cdots + a_1X^0 \quad \text{余 } a_0$$

余数正是所要求的任意进制数的最低位 a_0，再将 Q_1 除以 X，得商 Q_2 和余数，即

$$\frac{Q_1}{X} = a_{n-1}X^{n-3} + a_{n-2}X^{n-4} + \cdots + a_2X^0 \quad \text{余 } a_1$$

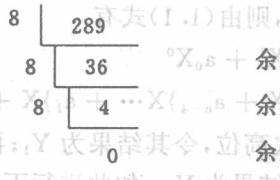
余数正好是任意进制数的次低位。如此进行下去直至商等于 0 为止，所得的一系列余数，正好是所要求的任意进制数各位。

例如，若需将 17，289，3910 分别转换成相应的二进制数、八进制数、十六进制数，则用竖式进行计算和转换如下：

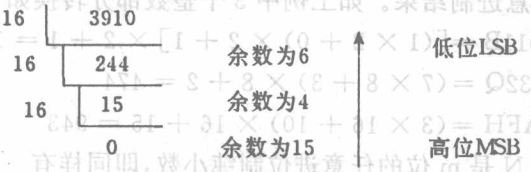


所以 $17 = 10001B$

此为试读，需要完整PDF请访问：www.ertongbook.com



所以 $289=441Q$



所以 $3910=F46H$

(3)十进制小数转换为任意进制小数：设 N 为要转换的十进制小数，它相应的任意进位制小数共 m 位，则

$$N = a_{-1}X^{-1} + a_{-2}X^{-2} + \cdots + a_{-m}X^{-m}$$

等式两边同乘以基数，得到

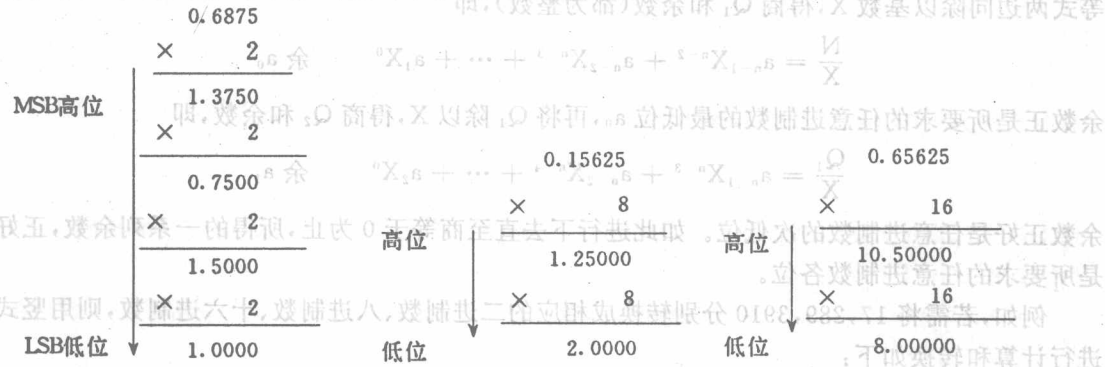
$$XN = a_{-1} + (a_{-2}X^{-1} + \cdots + a_{-m}X^{-m+1}) = a_{-1} + D_1$$

式中 a_{-1} 为整数部分，它正好等于所要求的任意进位制数的最高位，D_1 为所剩的小数部分。若再将 D_1 乘以 X 便得

$$XN = a_{-2} + (a_{-3}X + \cdots + a_{-m}X^{-m+2}) = a_{-2} + D_2$$

整数部分正好是所要求的任意进位制数的次高位。如此继续直至 $D_m=0$，即可得所要求的任意进位制小数各位。

例如，要将 0.6875，0.15625，0.65625 三个十进制小数分别转换为二进制小数、八进制小数和十六进制小数，可采用列竖式计算的办法转换如下：



所以 $0.6875=0.1011B$　　所以 $0.15625=0.12Q$　　所以 $0.65625=0.A8H$

必须注意，在进行任意进位制数和十进制数的相互转换时，由于整数部分和小数部分的转换方法截然不同，当整数部分和小数部分在形式上相同时，它们的转换结果在形式上却完全不同。因此，若一个数由整数和小数两部分组成，必须分开进行转换。另外，一个二进制小数能够完全准确地转换成十进制小数，但一个十进制小数不一定能完全准确地转换成二进制小数，例如，0.1=0.0001 1001 1001 100…B。这就是说，十进制小数 0.1 转换成二进制后成为一个无限循环的小数，不能准确地被表示出来。因此，有时不能用有限位的二进制小数去表示任意一个有限位的十进制小数，这是二进制计数制的一个缺点。

2. 八进制数与二进制数之间的相互转换

由于 $8=2^3$，故一位八进制数相当于三位二进制数，因此，八进制与二进制之间的相互转换十分简便。将一个八进制数转换成二进制数时，只要将每位八进制数用三位二进制数表示即可；而将一个二进制整数转换成八进制整数时，只要从低位开始，每三位分为一组，不够三位的以 0 补足三位，然后将每组二进制数分别用相应的八进制数表示即可。将一个二进制小数转换成八进制小数时，则从最高位开始，每三位分为一组，不足三位的以 0 补足，然后把每组二进制数分别用相应的八进制数表示。例如：

467Q = 100110111B

0.532Q = 0.101011010B

010 111 000 101B = 2705Q

0.100 101 011 110B = 0.4536Q

3. 十六进制数与二进制数之间的转换

由于 $16=2^4$，故一位十六进制数相当于四位二进制数，因此十六进制数与二进制数的转换，与前述八进制数与二进制数转换相类似。转换方法是：十六进制数转换为二进制数时，无论是整数还是小数，只要把每一位十六进制数用相应的四位二进制数代替即可；而二进制数转换为十六进制数与二进制数转换为八进制数方法类似，所不同的只是每四位分为一组。例如：

EFB.3DH = 1110 1111 1011.0011 1101B

0001 1010 1111.1000 0101 B = 1AF.85H

计算机中，数以二进制形式表示和运算，但二进制数书写不方便，易错，因此，常用八进制或十六进制数来书写。

三、二进制数的运算方法

二进制计数制除物理实现简便外，运算也较十进制大为简单。以下介绍二进制数的加、减、乘、除运算。

1. 二进制加法

二进制的加法规则为

0+0=0

0+1=1+0=1

1+1=0　　进位 1

1+1+1=1　　进位 1

故两个二进制数相加时，每一位可能有三个数（本位被加数和加数以及低位来的进位）相加，相加时，可按二进制数的加法规则进行。例如，1011B+1010B：

```
被加数      1 0 1 1
加数        1 0 1 0
进位    +1  1
------------------
        1 0 1 0 1
```

所以　　1011B+1010B=10101B

2. 二进制减法

二进制减法规则为

0−0=0　　0−1=1　　有借位

1−1=0　　0−1−1=0　　有借位

1−0=1

与加法相似，每一位也可能有三个数(本位被减数和减数以及从低位来的借位)参加运算，相减时按二进制减法规则进行。例如：

11000000B－00101010B写成竖式为

```
被减数    1 1 0 0 0 0 0 0
减数      0 0 1 0 1 0 1 0
借位    －   1 1 1 1 1
        ─────────────────
          1 0 0 1 0 1 1 0
```

所以　　11000000B－00101010B＝10010110B

3. 二进制乘法

二进制数的乘法规则为

0×0＝0　　1×0＝0

0×1＝0　　1×1＝1

只有两个1相乘时，积才为1，否则积为0。

两个二进制数相乘与两个十进制数相乘类似，可用乘数的每一位去乘被乘数，然后把乘得的中间结果同时相加即可。例如，1110B×0110B用竖式计算如下：

```
被乘数          1 1 1 0
乘数          × 0 1 1 0
              ─────────
中间结果        0 0 0 0
中间结果      1 1 1 0
中间结果    1 1 1 0
中间结果  ＋) 0 0 0 0
              ─────────
积        1 0 1 0 1 0 0
```

每一次的中间结果取决于乘数，若乘数的某一位为1，中间结果便为被乘数；若该位为0，则中间结果为0。这种算法在乘数位数超过2时，计算机实现较困难，为此，用移位加法进行。下面介绍移位加的算法。如上例采用移位加可重作如下：

```
被乘数                          1 1 1 0
乘　数                          0 1 1 0
初始部分积                      0 0 0 0
乘数最低位为0，加全0          ＋ 0 0 0 0
部分积                          0 0 0 0
部分积右移一位                  0 0 0 0 0
乘数次低位为1，加被乘数       ＋   1 1 1 0
部分积                          1 1 1 0 0
部分积右移一位                  0 1 1 1 0 0
乘数第三低位为1，加被乘数     ＋ 1 1 1 0
部分积                        1 0 1 0 1 0 0
部分积右移一位                  1 0 1 0 1 0 0
乘数最高位为0，加全0          ＋ 0 0 0 0
部分积                          1 0 1 0 1 0 0
部分积右移一位得乘积          0 1 0 1 0 1 0 0
```

此方法是将原来的中间结果左移一位改为部分积右移一位，中间结果不动，所以两种方法结果相同。

4. 二进制数的除法

除法是乘法的逆运算，二进制除法既可采用类似于十进制除法的竖式算法，又可采用移位减的方法。下面以 100011B 除以 101B 为例来说明二进制竖式除法的过程：

```
                    000111          商
除 数   101  ⟌     100011          被除数
                    101
                   ------
                    0111            余 数
                     101
                   ------
                    00101           余 数
                      101
                   ------
                        0           余 数
```

所以
$$100011B \div 101B = 111B$$

二进制除法是从被除数最高位开始，将被除数（或中间余数）与除数相比较，若被除数（或中间余数）大于除数，则被除数（或中间余数）减去余数，商 1，并将相减后得到的中间余数左移一位（中间余数的最低位用下一位被除数补充）作为下一次的中间余数。否则，则不作减法，商 0，并将本次的中间余数左移一位（中间余数的最低位用下一位被除数补充）得下一位的中间余数。如此逐次地进行比较、相减和移位，就可完成二进制除法运算。

§1.2 二进制中带符号数的表示方法及运算

上节叙述数的表示和运算都没有涉及符号，计算机进行运算时，往往会碰到带符号数（正、负数）的运算，带符号数在计算机中有原码、反码和补码三种表示法。反码运算对计算机的结构有特殊要求，不常采用，故本节着重介绍原码、补码及其相应的运算法则。

一、原码

对一个二进制数而言，若用最高位表示数的符号（常以 0 表示正数，1 表示负数），其余各位表示数值本身，则称为该二进制数的原码表示法。例如：

$$X = +1011100 \qquad [X]_{原} = \underset{符号}{0}\ \underbrace{1011100}_{数值}$$

$$Y = -1011100 \qquad [Y]_{原} = \underset{符号}{1}\ \underbrace{1011100}_{数值}$$

$[X]_{原}$和$[Y]_{原}$分别为 X 和 Y 的原码，是符号数值化了的数，可在计算机中使用，称为机器数。原来带正负号的数 X 和 Y 称为相应机器数的真值。原码$[X]_{原}$和真值 X 之间的关系如下：

1. 正数的原码表示

设 $X = +X_{n-2}X_{n-3}\cdots X_1X_0$（即 n－1 位二进制正数），则

$$[X]_{原} = 0X_{n-2}X_{n-3}\cdots X_1X_0 \qquad \text{（n 位二进制数，其中最高位为符号位）。}$$

2. 负数的原码表示

设 $X = -X_{n-2}X_{n-3}\cdots X_1X_0$（即 n－1 位二进制负数），则

$$[X]_{原} = 1X_{n-2}X_{n-3}\cdots X_1X_0 = 2^{n-1} + X_{n-2}X_{n-3}\cdots X_1X_0$$
$$= 2^{n-1} - (-X_{n-2}X_{n-3}\cdots X_1X_0) = 2^{n-1} - X$$

也是一个 n 位二进制数，其中最高位为符号位。

3. 零的原码表示

二进制数原码表示，有正零和负零之分，即

$$\left.\begin{aligned}[+0]_{原}&=000\cdots00\\ [-0]_{原}&=100\cdots00\end{aligned}\right\}\text{(n 位二进制数,最高位为符号位)}$$

综上所述,原码和真值的关系可归纳为如下数学定义式:

$$[X]_{原}=\left.\begin{cases}X & (X\geqslant+0)\\ 2^{n-1}-X & (X\leqslant-0)\end{cases}\right\}\text{n 位二进制数,最高位为符号位。}$$

原码表示法的最大优点是形式直观,同真值间的转换方便,但原码数进行运算时完全类同于正负数的笔算,计算机处理不方便。例如,两个正数相减,如果被减数的绝对值小于减数的绝对值,笔算时,必须将两者颠倒过来,再作减法,并把求得的差加上负号;用计算机实现时,同样有笔算的全部步骤,处理非常繁琐。

二、反码

反码是另一种带符号数的表示方法。一个二进制正数的反码与原码相同;一个二进制负数的反码为其原码除符号位外其余各位按位取反,即"1"都换成"0","0"都换成"1"。例如:

$X=+111\ 1111 \qquad [X]_{反}=0111\ 1111=[X]_{原}$

$X=-111\ 1111 \qquad [X]_{反}=1000\ 0000$

同样可导出反码和真值关系的数学定义式为:

$$[X]_{反}=\left.\begin{cases}X & (X\geqslant+0)\\ 2^{n-1}+X & (X\leqslant-0)\end{cases}\right\}\text{n 位二进制数,最高位为符号位}$$

三、补码

在数的补码表示法中,参加运算的数的符号与数一样,也可以参加运算,因而计算机中用来表示带符号数的实际方法是补码表示法。

为说明补码的概念,先从时钟谈起。若现在是北京时间 3 点整,而时钟却指着 5 点整,快两小时。要将时钟拨准有两种方法:一种是把时钟倒拨两小时,若把这种逆时钟拨法看成是作减法,则相当于 5−2=3;另一种是顺时针拨 10 小时,若看成是作加法,则相当于:

$$5+10\xrightarrow[12\text{丢失}]{\text{在钟面上}}3$$

钟面上看到两种拨法都能拨准到 3 点钟。这是因为,从时钟面来看,时钟走到 12 点相当于零点整,12 小时一个循环,因而对钟面上任一时刻而言,减 2(或加−2)和加 10 在效果上是一样的;同理,减 5(或加−5)可以用加 7 来代替,而且它们有这样的关系,即 10=12+(−2),7=12+(−5)。我们称 10 是(−2)对 12 的补码,7 是(−5)对 12 的补码。

从进位的概念来看,时钟是十二进制,5+10 逢 12 进 1,在钟面上进位不能记录下来而丢失,所以只留下 3。在数学上把 12 叫做"模"(mod),是尺度的意思。因而上述问题可以说,10 是(−2)对模 12 的补码,7 是(−5)对模 12 的补码,也可以写出如下等式:

$$10+5=3 \qquad (\text{mod } 12)$$

无疑,对于十二进制(以 12 为模),如下等式总是成立:

$$X+12=X \qquad (\text{mod } 12)$$

因此当 X 为负数,如 X=−5,在模 12 的意义下有:

$$-5+12=-5 \quad (\text{mod } 12)$$

$$7=-5 \quad (\text{mod } 12)$$

这样,在模 12 的意义下,负数就可以转化为正数,而正负数相加也就可以转化为正数间的相

加，例如：

$$4+(-5)=4+7 \pmod{12}$$

考虑计算机运算的特点，计算机中的部件都有固定的位数，设位数为 n，则计算机中最大的计数值（包括符号位）为 2^n-1，逢 2^n 进 1。因此，计算机中数的补码是以 2^n 为模，即

$$[X]_{补}=2^n+X$$

当 X 为正数，如 $X=+X_{n-2}X_{n-3}\cdots X_1X_0$，则

$$[X]_{补}=2^n+X=2^n+X_{n-2}X_{n-3}\cdots X_1X_0$$
$$=0X_{n-2}X_{n-3}\cdots X_1X_0=[X]_{原}$$

正数的补码与原码相同。

当 X 为负数时，如 $X=-X_{n-2}X_{n-3}\cdots X_1X_0$，则

$$[X]_{补}=2^n+X=2^{n-1}+(2^{n-1}+X)$$

可见负数的补码仍是一个负数，它是将其原码的符号位保持不变，而将其数值部分求补得到的。例如：

若 $X_1=+1010011B$，$X_2=-1010011B$，则

$$[X_1]_{补}=01010011B$$
$$[X_2]_{补}=2^8+(-1010011B)=2^7+(2^7-1010011B)$$
$$=10101101B$$

当 X 为正零时，$[+0]_{补}=2^n+0000\cdots0=0$

当 X 为负零时，$[-0]_{补}=2^n-000\cdots0=0$

补码中零只有一种表示，无正负零之分。

一个二进制数，若以 2^n 为模，它的补码叫做 2 补码，简称补码；同理，一个十进制数，若以 10^n 为模，它的补码叫 10 补码。而一个二进制数，若以 2^n-1 为模，它的补码叫做 1 补码。由反码的定义式，显见 1 补码就是 X 的反码。

四、求补码的方法

因为正数的补码就是它本身，等于原码，只有负数才有求补的问题。以下介绍两种求补码的方法。

1. 根据定义求

$$[X]_{补}=2^n+X=2^n-|X|,X<0$$

即负数 X 的补码等于模 2^n 加上其真值（或减去其真值的绝对值）。如 $X=-1010111B$，$n=8$，则

$$[X]_{补}=2^8+(-1010111B)=100000000B-1010111B$$
$$=10101001B \pmod{2^n}$$

因为要作一次减法，这种方法很不方便。

2. 利用原码求

因为

$$[X]_{补}=2^n+X$$
$$[X]_{反}=2^n-1+X$$

所以

$$[X]_{补}=[X]_{反}+1$$

而一个负数的反码为原码除符号位外其余各位按位取反，所以一个负数 X 的补码等于其原码除符号位保持不变外，其余各位按位取反，再在最低位加 1。例如：

此为试读，需要完整PDF请访问：www.ertongbook.com