中等职业教育国家规划教材
全国中等职业教育教材审定委员会审定

计 算 机 原 理

（计算机及应用专业）

主　　编　梁　军
参　　编　王艳玲　刘益红
责任主审　宋方敏
审　　稿　钱树人　张幸儿

机械工业出版社

计算机原理是计算机类各专业的一门必修课，也是其他专业学习计算机基础知识的入门课程，它可为今后学习计算机知识打下良好的基础。

本教材是根据教育部最新颁布的中等职业学校计算机及应用专业《计算机原理》课程教学大纲的要求编写的。主要讲授了计算机中信息的表示方法、计算机系统的组成、中央处理器、指令系统、存储系统、总线系统、输入输出系统和外部设备。每章后面都配有习题。

本教材既可作为中等职业学校计算机及相关专业的教材也可作为有关专业技术人员的参考书。

图书在版编目（CIP）数据

计算机原理/梁军主编. —北京：机械工业出版社，2002.7
中等职业教育国家规划教材. 计算机及应用专业
ISBN 7-111-10517-6

Ⅰ. 计… Ⅱ. 梁… Ⅲ. 电子计算机—基础理论—专业学校—教材 Ⅳ. TP301

中国版本图书馆 CIP 数据核字（2002）第 045195 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

责任编辑：张 克
封面设计：姚 毅
责任印制：付方敏

煤炭工业出版社印刷厂印刷·新华书店北京发行所发行

2002 年 7 月第 1 版·第 1 次印刷
787mm×1092mm 1/16 · 10.5 印张 · 253 千字
0 001—5000 册
定价：12.60 元

出 版 说 明

为了贯彻《中共中央国务院关于深化教育改革全面推进素质教育的决定》精神，落实《面向 21 世纪教育振兴行动计划》中提出的职业教育课程改革和教材建设规划，根据教育部关于《中等职业教育国家规划教材申报、立项及管理意见》（教职成[2001]1 号）的精神，我们组织力量对实现中等职业教育培养目标和保证基本教学规格起保障作用的德育课程、文化基础课程、专业技术基础课程和 80 个重点建设专业主干课程的教材进行了规划和编写，从 2001 年秋季开学起，国家规划教材将陆续提供给各类中等职业学校选用。

国家规划教材是根据教育部最新颁布的德育课程、文化基础课程、专业技术基础课程和 80 个重点建设专业主干课程的教学大纲（课程教学基本要求）编写，并经全国中等职业教育教材审定委员会审定。新教材全面贯彻素质教育思想，从社会发展对高素质劳动者和中初级专门人才需要的实际出发，注重对学生的创新精神和实践能力的培养。新教材在理论体系、组织结构和阐述方法等方面均作了一些新的尝试。新教材实行一纲多本，努力为教材选用提供比较和选择，满足不同学制、不同专业和不同办学条件的教学需要。

希望各地、各部门积极推广和选用国家规划教材，并在使用过程中，注意总结经验，及时提出修改意见和建议，使之不断完善和提高。

教育部职业教育与成人教育司

2001 年 10 月

前　言

自从世界上第一台电子数字计算机 ENIAC 问世以来，计算机的发展异常迅速，在短短几十年的时间内，计算机的硬件系统经历了电子管、晶体管、小规模集成电路、大规模集成电路和超大规模集成电路等几个发展时代。在软件方面，操作系统、数据库系统、程序设计语言的发展也是日新月异。计算机的应用已深入到科学计算、信息处理、过程控制、事务处理、仪器仪表制造和家用电器等各个方面。因此，了解与掌握计算机系统的组成与基本工作原理是非常必要的。

本教材根据教育部最新颁布的中等职业学校计算机及应用专业《计算机原理》课程教学大纲的要求编写，按照面向实用，重视基础，便于理解的原则，从计算机基础知识入手，用通俗易懂的语言和简单明了的例题介绍了计算机硬件和软件的基础知识，讲述了计算机的工作原理。描述通俗易懂，概念准确，结构安排合理。全书共分 8 章。第 1 章讲述计算机中数据的表示形式，内容包括：各种进位计数制及其相互之间的转换、计算机中带符号数的表示方法、补码的加减法运算、溢出判断方法、定点数与浮点数的表示、计算机中常用的编码。第 2 章讲述计算机系统的组成，内容包括：计算机的发展概况和应用、计算机系统的组成、计算机的基本工作原理。第 3 章讲述中央处理器，内容包括：处理器的基本组成、CPU 的时序、常见的微处理器。第 4 章讲述指令系统与程序设计，内容包括：指令和程序、寻址方式、8086 指令系统、汇编语言源程序格式、汇编语言程序设计。第 5 章讲述存储系统，内容包括：存储器的基本结构、存储器的操作、计算机中的存储器类型、半导体存储器的性能指标、半导体存储器、存储器的扩展、存储器的分级结构、高速缓冲存储器、虚拟存储器。第 6 章讲述微机总线系统，内容包括：总线的概念、总线的分类、总线的工作原理、微机常用总线、外部通信总线。第 7 章讲述输入与输出，内容包括：输入输出的基本概念、输入输出传输方式、中断。第 8 章讲述计算机系统常用外部设备，内容包括：键盘、鼠标、打印机、显示器及外存。

本书第 1 章、第 2 章、第 5 章、第 7 章由梁军编写，第 3 章、第 4 章、第 6 章、第 8 章由王艳玲编写，刘益红参与了第 7 章、第 8 章部分内容的编写。李新平为本书绘制了大量插图，在此表示感谢。全书由梁军统稿。在本书的编写过程中，得到了山东省电子工业学校各级领导的大力支持和协助，并提出了许多宝贵意见，谨此表示由衷的感谢。

由于编者水平有限，加之时间仓促，书中可能还存在一些错误和疏漏，恳请广大读者批评批正。

编　者

目　录

第1章　计算机中数的表示方法

计算机的功能十分强大，它可以完成各种各样的工作。但无论是什么样的计算机，从本质上讲它的工作过程都是信息的传递过程，通过信息的不断传递，计算机可以完成各种工作。在计算机内部传送的信息分为两大类，一类是控制信息，另一类是数据信息。控制信息用于对计算机内部各个组成部件的控制，完成对数据信息的加工处理。数据信息是计算机加工处理的对象。数据信息又分为两种，一种用于各种数值运算，称之为数值数据；另一种用于逻辑运算和输入输出，称之为非数值数据。

1.1　计算机中的数制及转换方法

1.1.1　数制的基数与权

按进位方式计数的数制称为进位计数制，简称数制，数制是人们利用符号计数的方法。人们在很早的年代里就使用了逢十进一的十进制计数方法，随着人们对计数方法认识的不断提高，又出现了一些其他的数制。如时、分、秒之间使用的是六十进制，月和年之间使用的是十二进制，计算机中采用二进制等等。

任何一种数制都有两个要素，即基数和权。基数为某种数制中每个数位上允许使用的数码的个数。例如十进制的基数为 10，每个数位上允许使用的数码为 0、1、2、……、9 中的任意一个，六十进制的基数为 60，每个数位上允许使用的数码为 0、1、2、……、59 中的任意一个。所以，当基数为 R 时，该数制中每个数位上允许使用的数码的个数为 R 个，其取值范围为 0~（R-1）。在进行加减法运算时按逢 R 进 1，借 1 当 R 的规则进行。在某一种数制中，同一个数码处在不同的位置时，它所表示的数的大小是不同的。例如在十进制数 99.9 中，最高位是十位，该位上的 9 所表示的数值为 9×10^1，第二位是个位，该位上的 9 所表示的数值为 9×10^0，小数点后面第一位上的 9 所表示的数值为 9×10^{-1}。

$$99.9=9\times10^1+9\times10^0+9\times10^{-1}$$

可以看出，每个数码所表示的数值的大小与这个数所处的位置有关，由此可以得出位权的概念。位权是某一数位上的 1 所表示的数值的大小；它是一个指数，底是基数 R，幂是数码的位置号，数码的位置号从 0 开始，位权简称为权。将一个数中某一位的数码与该位的位权相乘，即为该位数码的数值。设一个任意进制数 S 的整数部分有 n 位，小数部分有 m 位，基数为 R，则 S 的按权展开表达式为：

$$S=k_{n-1}R^{n-1}+\cdots+k_1R^1+k_0R^0+k_{-1}R^{-1}+\cdots+k_{-m}R^{-m}=\sum_{i=-m}^{n-1}k_i\times R^i$$

其中 k 的取值范围为 0~(R-1)。

1.1.2 二进制与十六进制

1．二进制

基数 R=2 的数制称为二进制，计数时逢二进一，借一当二。二进制是计算机惟一可以识别的数制，迄今为止，所有计算机都以二进制的形式进行算术运算和逻辑运算。为什么在计算机中要采用二进制呢？这是因为二进制具有以下优点：

（1）二进制只有 0 和 1 两个数码，实现起来最简单而且最可靠。由于数在计算机中是以器件的物理状态来表示的，所以一个具有两种稳定状态且能相互转换的器件就可以用来表示一位二进制数，实现起来非常方便。例如晶体管的饱和与截止、开关的接通与断开、灯的亮与灭等。一个任意的整数位为 n 位，小数位为 m 位的二进制数 S 的按权展开表达式为

$$S=k_{n-1}2^{n-1}+\cdots+k_1 2^1+k_0 2^0+k_{-1}2^{-1}+\cdots+k_{-m}2^{-m}=\sum_{i=-m}^{n-1} ki\times 2^i$$

其中 k 的取值范围为 0 和 1。

（2）二进制采用逢二进一的计数规则，运算简单。

2．十六进制

尽管二进制具有实现起来简单可靠、运算时方便的优点，但是在使用二进制编写程序或表示数据时既繁琐又容易出错，所以人们在编写程序时经常使用十六进制。十六进制具有以下特点：

（1）十六进制的基数 R=16，每个数位上的数码共有 16 个，其中 0～9 与十进制的表示相同，10～15 用英文字母 A～F 表示。

（2）十六进制采用逢十六进一，借一当十六的计数规则，各位的权是以 16 为底的幂。一个任意的整数位为 n 位，小数位为 m 位的十六进制数 S 的按权展开表达式为

$$S=k_{n-1}16^{n-1}+\cdots+k_1 16^1+k_0 16^0+k_{-1}16^{-1}+\cdots+k_{-m}16^{-m}=\sum_{i=-m}^{n-1} ki\times 16^i$$

其中 k 的取值范围为 0～F。

另外，在计算机中经常使用的数制还有八进制和十进制。八进制的基数 R=8，计数时采用逢八进一，借一当八的原则，每个数位可以使用的数码为 0～7。在表示一个数时，为了区分不同进制的数，采用各种进制的英文名称的第一个字母作为后缀加在数的后面进行区分。二进制数的后缀为 B，如 1011B；八进制数的后缀为 O，但因 O 与 0 容易混淆，所以八进制数常用 Q 作后缀，如 123Q；十进制数后缀为 D 或无后缀，如 123D 或 123；十六进制数后缀为 H，如 123FH。

1.1.3 数制的转换方法

在各种进位计数制中，十进制是人们日常生活中最习惯使用的计数方式，十六进制是程序设计中经常使用的计数方式，但是在计算机中只能使用二进制数。所以在各种进制之间要经常进行转换。

1．二进制、八进制、十六进制转换为十进制

把任意进制转换为十进制时，可以采用按权展开求和的方法完成。

【例 1-1】把 1011.11B、45.5Q、78.5H 转换为十进制数。

采用按权展开求和的方法把各数转换为十进制数。

$1011.11B=1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}=11.75$

$45.5Q=4\times8^1+5\times8^0+5\times8^{-1}=37.625$

$78.5H=7\times16^1+8\times16^0+5\times16^{-1}=120.3$

2．十进制转换为二进制、八进制、十六进制

十进制数转换为任意进制数时，整数部分和小数部分应分别进行。

（1）整数部分

整数部分的转换规则是将被转换的数反复除以基数取余数直到商为 0 为止。由此所得到的余数序列就是转换结果。其中该序列的第 1 个余数为转换结果的最低位，最后 1 个余数为转换结果的最高位。

【例 1-2】把十进制整数 49 分别转换为二进制数、八进制数和十六进制数。

		余数序列
2	49	
2	24	1
2	12	0
2	6	0
2	3	0
2	1	1
	0	1

49 转换为二进制数的结果为 110001B

		余数序列
8	49	
8	6	1
	0	6

49 转换为八进制数的结果为 61Q

		余数序列
16	49	
16	3	1
	0	3

49 转换为十六进制数的结果为 31H

如果要验证转换结果是否正确，可以把转换结果用按权展开求和的方法再转换成十进制数。

（2）小数部分

小数部分的转换规则是将被转换的数反复乘以基数取整数进位，直到乘积的小数部分为 0 或精度满足要求为止。由此可得到一个进位序列，该序列的第 1 个进位是转换结果的最高位，最后 1 个进位是转换结果的最低位。

【例 1-3】将十进制数 0.625 分别转换为二进制数、八进制数和十六进制数。

$0.625\times2=1.25$　　整数进位为 1，小数部分为 0.25

0.25×2=0.5　　整数进位为 0，小数部分为 0.5

0.5×2=1　　整数进位为 1，小数部分为 0

0.625 转换为二进制数的结果为 0.101B

0.625×8=5.0　　整数进位为 5，小数部分为 0

0.625 转换为八进制数的结果为 0.5Q

0.625×16=10.0　　整数进位为 A，小数部分为 0

0.625 转换为十六进制数的结果为 0.AH

3．二进制、八进制、十六进制之间的相互转换

前面讨论了各种进制与十进制之间的转换方法，所以二进制、八进制、十六进制之间的相互转换可以利用十进制作为桥梁来完成，但是这样转换比较麻烦，三种进制之间的转换可以借助二进制数完成。

（1）二进制转换为八进制或十六进制

二进制转换为八进制时，以小数点为中心，向左向右三位分为一组，每一组转换为一位八进制数；二进制转换为十六进制时，以小数点为中心，向左向右四位分为一组，每一组转换为一位十六进制数。在分组时如果两端位数不够时可以补零。

【例 1-4】将二进制数 100110101.11011B 转换为八进制和十六进制。

100110101.11011B=100 110 101.110 110B =465.66Q

100110101.11011B=0001 0011 0101.1101 1000B =135.D8H

（2）八进制或十六进制转换为二进制

把八进制或十六进制转换为二进制时，可以把每位数直接写成三位或四位二进制便可以完成转换。

【例 1-5】将 672.35Q 和 3F4D.ABH 转换为二进制数。

672.35Q=110111010.011101B

3F4D.ABH=11111101001101.10101011B

（3）八进制与十六进制之间的转换

八进制与十六进制之间的转换可以借助于二进制完成，先将八进制或十六进制转换为二进制，然后再将二进制转换为十六进制或八进制。

【例 1-6】将八进制数 45Q 转换为十六进制数。

45Q=100101B=25H

1.2　二进制数运算

二进制的基本运算规则是逢二进一，借一当二，由于它只有两个数码 0 和 1，所以二进制的四则运算比其他进制简单。

1．加法运算

1）0+0=0

2）0+1=1

3）1+0=1

4）1+1=0，进位为 1

2．减法运算

1）0-0=0

2）1-0=1

3）1-1=0

4）0-1=1，借位为 1

3．乘法运算

1）0×0=0

2）0×1=0

3）1×0=0

4）1×1=1

【例 1-7】求 1011 与 1101 的乘积。

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

4．除法运算

除法运算的规则与十进制类似，从被除数的最左位开始检查，找到超过除数的位置，在此置商 1，并将被除数减去除数得到余数，然后将被除数的下一位添到余数上，重复进行前面的运算，直到被除数的所有位都被除完为止。

【例 1-8】求 10011101÷1101

$$\begin{array}{r} 1100 \\ 1101\overline{)10011101} \\ \underline{1101} \\ 01101 \\ \underline{1101} \\ 00000 \\ \underline{0000} \\ 0001 \end{array}$$

1.3 带符号数的表示方法

1.3.1 机器数和真值

前面所提到的各种进制的数，没有涉及到符号问题，都是按无符号数进行处理。在实际应用中表示一个数时，除了采用什么数制之外，一个数还可以有正数和负数之分。在计算机中无法直接表示符号，那么在计算机中是怎样表示一个数的正负的呢？由于一个数的符号只有“正”或“负”两种情况，正好可以用一位二进制数来表示，所以在计算机中把一个数的

最高位当做符号位，用来表示数的正或负，用 0 表示“+”号，用 1 表示“-”号，这就是一个有符号数在计算机中的表示形式，符号位数字化的数称之为机器数，而原来带“+”号和“-”号的数称为真值。在计算机中常用的机器数有原码、反码、补码三种形式，当真值为 X 时，其原码、反码、补码分别用$[X]_{原}$、$[X]_{反}$、$[X]_{补}$表示。

1.3.2 原码、反码和补码

1．原码

原码的定义如下：

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

其中$[X]_{原}$为原码，X 为真值，n 为原码的位数。

由真值求原码非常简单，原码的的数值部分与真值的绝对值相同，只需符号位 1 或 0 替换“+”、“-”号就可以了，当真值为正数时，符号位为 0，当真值为负数时，符号位为 1。所以原码与真值的差别只是用 0 和 1 代替“+”号和“-”号，而数值位与真值相同。在原码中 0 有+0 和-0 之分。原码表示数的范围为$-(2^{n-1}-1)\leqslant X\leqslant +(2^{n-1}-1)$，当 n=8 时，原码表示数的范围为$-127\leqslant X\leqslant +127$

【例 1-9】已知 X_1=0001011B，X_2=-1011111B，X_3=0,X_4=-0 求出它们的原码。要求写成 8 位的形式。

$[X_1]_{原}$=00001011B

$[X_2]_{原}$=11011111B

$[X_3]_{原}$=00000000B

$[X_4]_{原}$=10000000B

2．反码

反码定义如下：

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

其中$[X]_{反}$为反码，X 为真值，n 为反码的位数。

设真值为 X，当 X>0 时，其反码与原码相同，当 X<0 时把原码的数值位各位取反即是反码。在反码中 0 同样有正负之分。反码的取值范围与原码相同。

【例 1-10】求例【例 1-9】中各数的反码。

$[X_1]_{反}$=00001011B

$[X_2]_{反}$=10100000B

$[X_3]_{反}$=00000000B

$[X_4]_{反}$=11111111B

3．补码

在计算机中进行乘除法运算时，使用原码非常方便。将两个数的数值部分作绝对值乘除法，求出运算结果的绝对值，然后将两个数的符号位进行异或运算求出运算结果的符号，两部分合成后便可以求出运算结果。但使用原码作减法运算则比较麻烦，首先要判断被减数和

减数绝对值的大小，然后用绝对值大的数减绝对值小的数，结果的符号由绝对值大的数的符号决定。用补码进行加减法运算时，无论是加法还是减法都作加法运算，运算结果的符号位和数值位同时得到，运算过程比用原码简单。由于在解决一个问题时，可能既有加减法又有乘除法，如果乘除法用原码进行，加减法用补码完成，则需要经常进行原码和补码之间的转换，所以在计算机中各种运算都用补码完成。补码是如何将减法转换为加法的呢？首先介绍一下“模”的概念和性质。一个计数系统可以表示的数的个数称为计数系统的模，记为 M 或 MOD M。例如：一个 3 位的十进制计数系统表示数的范围是 0~999，可以表示的数为 1000 个，它的模等于 1000；一个 n 位二进制计数系统，它可以表示 2^n 个不同的数，它的模为 2^n；钟表可以表示 12 个钟点，它的模为 12。

模具有这样的性质，当模为 2^n 时，2^n 和 0 表示形式是相同的。例如一个 n 位二进制计数器，可以从 0 计数到 2^{n-1}，如果再加 1，计数器就变成了 0。所以 2^n 和 0 在 n 位二进制计数器中的表示形式是一样的，同样钟表的零点和 12 点的表示形式是相同的。由此也可以看出，当一个计数系统的计数值达到模时，模将会自动丢失。引入模的概念后，可以把减法转换成加法进行。例如：当前正确的时间为北京时间下午 5 点，而您的手表停在上午 8 点，为了把表对准您可以顺时针拨 9 个小时，也可以逆时针拨 3 个小时。如果顺时针定义为加法，逆时针则为减法，可以得到下面两个表达式：

顺时针　　8+9=17=12+5=5　　模 12 自动丢失

逆时针　　8−3=5

比较以上两个表达式可以发现，当以 12 为模时 8+9=17，由于 12 自动丢失，所以 8+9=5，与 8−3 相等。在这里把 9 称为−3 的补码。由此可以看出减法可以转换为加法进行，减去一个数等于加上这个数的补码。在计算机中，n 位加法器最高位的进位也会和钟表中的时针一样自动丢失模。它所丢失的模为 2^n。

补码定义如下：

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

其中 $[X]_{补}$ 为补码，X 为真值，n 为补码的位数。

在补码中 0 没有正负之分，它的表示形式是惟一的。补码表示数的范围为 $-2^{n-1}\leqslant X\leqslant 2^{n-1}-1$，当 n=8 时，补码表示数的范围为 $-128\leqslant X\leqslant 127$，与原码和反码相比，补码表示数的范围是不对称的。

比较原码、反码、补码三者的定义可以看到以下几点：

（1）原码、反码、补码的符号位都是用 1 表示正数，用 0 表示负数。

（2）正数的原码、反码、补码相同。

（3）当 X 为负数时，反码数值位与原码各位相反，在反码的的数值位末位加 1 便可以得到补码。

（4）在原码和反码中 0 有正负之分，在补码中 0 的表示是惟一的。

表 1-1 中给出了 8 位机器数的三种表示形式的对应关系。

因为原码与真值最接近，所以当真值为负时，不必直接由定义求它的反码和补码，可以根据原码、反码、补码三者之间的关系，先求出原码，保持原码的符号位不变，数值位各位

取反可以得到反码，反码末位加 1 可以得到补码。

表 1-1　8 位二进制数的表示形式

二进制数码形式	无符号十进制数	原　码	反　码	补　码
00000000B	0	+0	+0	+0
00000001B	1	+1	+1	+1
00000010B	2	+2	+2	+2
…	…	…	…	…
01111110B	126	+126	+126	+126
01111111B	127	+127	+127	+127
10000000B	128	-0	-127	-128
10000001B	129	-1	-126	-127
…	…	…	…	…
11111101B	253	-125	-2	-3
11111110B	254	-126	-1	-2
11111111B	255	-127	-0	-1

【例 1-11】求【例 1-9】中各数的补码。

$[X_1]_{补}$=00001011B

$[X_2]_{补}$=10100001B

$[X_3]_{补}$=00000000B

$[X_4]_{补}$=00000000B

如果已知机器数，需要求出相应的真值，应该如何进行？由于原码与真值最相近，所以只需要把各种机器数转换为原码，然后把原码的符号位用正负号替换即可。真值为正数时，原码、反码、补码相同，无需转换。真值为负数时，反码和补码转换为原码的方法与原码转换为反码和补码的方法相同。反码的数值位各位取反，可转换为原码，补码的数值位取反后末位加 1，可转换为原码。

【例 1-12】已知$[X_1]_{补}$=10011111B，$[X_2]_{补}$=01101101B，求 X_1、X_2。

$[X_1]_{原}$=11100001B　　X_1= -1100001B

$[X_2]_{原}$=01101101B　　X_2=01101101B

1.3.3　补码运算

如前所述，在计算机中，为了避免各种机器数之间相互转换所带来的麻烦，有符号数一律采用补码表示。在补码运算中，加减法运算是最基本的运算，补码的乘除法运算都可以通过加减法完成。

1．补码加法运算

补码加法运算规则是：

$[X+Y]_{补}=[X]_{补}+[Y]_{补}$　　(MOD 2^n)

【例 1-13】已知 X=1100011B，Y= -00111B，求 X+Y=？。

$[X]_{补}$=01100011B，$[Y]_{补}$=11111001B

$[X]_{补}+[Y]_{补}$=　01100011B

　　+11111001B

模溢出→　101011100 B

$[X+Y]_{补}=[X]_{补}+[Y]_{补}$=01011100B，最高位进位为模溢出，自动丢失。

X+Y=1011100B。

2. 补码减法运算

在微型计算机中减法运算通过补码也转换为加法运算完成，减法运算的规则可由加法运算规则得出：

$[X-Y]_{补}=[X+(-Y)]_{补}=[X]_{补}+[-Y]_{补}$

其中$[-Y]_{补}$称为$[Y]_{补}$的机器负数，可以直接由$[Y]_{补}$求出，把$[Y]_{补}$的符号位与数值位一起取反，末位加 1，结果就等于$[-Y]_{补}$。

【例 1-14】已知 X=1000111B，Y=1001B，求 X-Y=？。

$[X]_{补}$=01000111B，$[Y]_{补}$=00001001B，$[-Y]_{补}$=11110111B

$[X]_{补}+[-Y]_{补}$=　01000111B

　　+ 11110111B

模溢出 →　100111110B

$[X-Y]_{补}$=00111110B

X-Y=00111110B

3. 溢出判断

（1）溢出的概念

在计算机内部表示数据与手工表示数据的情况不同，手工表示数据时，数据的位数可以为任意位，而在计算机内只能用有限位来表示数据，所以表示的数有一定的范围，如果超出范围，计算机将无法正确表示，这时会造成数据的最高位丢失，使数据产生错误，将这种情况称为溢出。例如：8 位补码所能表示数的范围为-128~+127，如果数据超出这个范围，就会产生溢出，出现溢出时，计算机应停止运算，进行错误处理。

（2）溢出判断

两个用补码表示的数作加减法运算时，如果是同号相减或异号相加，只能使数据的绝值越来越小，运算结果不可能产生溢出；如果是同号相加或异号相减，则运算结果可能会出现溢出。此时可以把运算结果的符号与参与运算的数据的符号相比较，如果出现正数加正数得负数或负数加负数得正数的情况，则可以断定运算结果出现了溢出。

【例 1-15】已知 X=114，Y=77，利用 8 位补码计算 X+Y=?

分析：如果人工计算，正确的运算结果应该为 191，但是这个结果显然已经超过了 8 位补码的表示范围，计算结果将会出现溢出。

X=114=1110010B　Y=77=1001101B

$[X]_{补}$=01110010B　$[Y]_{补}$=01001101B

$[X]_{补}+[Y]_{补}$=　01110010B

　　+ 01001101B

　　10111111B

$[X+Y]_{补}$=10111111B

X+Y=-33

两个正数相加，结果为负数，这是由于运算结果溢出造成的。

【例 1-16】已知 X=-1100111B　Y=1001100B，求 X-Y=?

$[X]_{补}$=10011001B，$[Y]_{补}$=01001100B，$[-Y]_{补}$=10110100B

$[X]_{补}+[-Y]_{补}=$ 10011001B

+ 10110100 B

模溢出 → 101001101B

运算结果为负数加负数结果为正数，产生溢出。

【例 1-17】已知 X=1100111B Y=1110011B， 求 X−Y=?

$[X]_{补}$=01100111B， $[Y]_{补}$=01110011B， $[-Y]_{补}$=10001101B

$[X]_{补}+[-Y]_{补}=$ 01100111B

+ 10001101B

11110100B

由于是同号相减，运算结果不可能产生溢出。

所以 $[X-Y]_{补}$=11110100 B，X−Y= −0001100B

判断溢出的另一种方法是采用双符号位补码进行加减法运算，双符号位补码用 00 表示正数，用 11 表示负数。用双符号位补码判断运算结果是否有溢出时，只需要判断结果的双符号位是否相同即可，如果双符号位相同，运算结果没有溢出，否则运算结果有溢出。

【例 1-18】 已知 X= −1100111B Y=1001100B， 求 X+Y 和 X−Y。

$[X]_{变形补}$=110011001B，$[Y]_{变形补}$=001001100B，$[-Y]_{变形补}$=110110100B

$[X]_{变形补}+[Y]_{变形补}=$ 110011001B

+ 001001100B

111100101B

双符号位相同，结果无溢出。

X+Y= −0011011

$[X]_{变形补}+[-Y]_{变形补}=$ 110011001B

+ 110110100B

1101001101B

双符号位不同，结果溢出。

4. 无符号数的加减法运算

在做无符号数的加减法运算时，可以把两个数都看成是正数，并且所有各位都是数值位。当两个无符号数相加时，由于两个数都是正数，所以运算结果也为正数；当两个无符号数相减时，在计算机中也是采用补码完成。

1.4 定点数和浮点数

在计算机处理的数据中，多数情况下带有小数点，所以表示一个数时除了要考虑数制和数的符号如何表示外，还要考虑到小数点位置的处理。在计算机中采用两种方法表示数据的小数点，一种是定点表示法，另一种是浮点表示法。如果将小数点固定在某一位置，则称为定点表示法；如果小数点位置可以任意移动，则称为浮点表示法。

1.4.1 定点表示法

用定点表示法表示的数称为定点数。定点数规定：数的小数点位置固定不变，可以隐含