21 世纪高等学校规划教材

Intel汇编语言程序设计

葛洪伟 姜浩伟 赵雅群 黄蓓 编著

中国电力出版社
www.infopower.com.cn

21 世纪高等学校规划教材

Intel 汇编语言程序设计

葛洪伟 姜浩伟 赵雅群 黄蓓 编著

中国电力出版社
www.infopower.com.cn

内容简介

本书以当前广泛使用的80x86/Pentium系列微型计算机为背景，系统介绍了汇编语言程序设计的基本理论、方法和应用技术。全书共分9章，主要内容包括：汇编语言程序设计的基础知识、寻址方式、程序结构、上机调试过程、指令系统及基本程序设计技术、模块化程序设计技术、高级汇编技术、中断及中断处理、输入/输出程序设计和保护模式下的编程技术等。本书内容由浅入深、循序渐进，理论和实际并重。书中提供了大量的程序实例，每章后面均附有习题。

本书是高等院校计算机及相关专业本、专科的汇编语言课程教材，也可作为从事相关工作的技术人员的参考书。

图书在版编目（CIP）数据

Intel汇编语言程序设计/葛洪伟等编著. —北京：中国电力出版社，2007

21世纪高等学校规划教材

ISBN 978-7-5083-4681-6

Ⅰ. I... Ⅱ. 葛... Ⅲ. 汇编语言-程序设计-高等学校-教材 Ⅳ. TP313

中国版本图书馆CIP数据核字（2006）第125677号

丛 书 名：21世纪高等学校规划教材

书　　名：Intel汇编语言程序设计

出版发行：中国电力出版社

地　　址：北京市三里河路6号　　邮政编码：100044

电　　话：（010）68362602　　传　　真：（010）68316497，88383619

本书如有印装质量问题，我社负责退换

服务电话：（010）88515918（总机）　　传　　真：（010）88518169

E-mail：infopower@cepp.com.cn

印　　刷：北京市同江印刷厂

开本尺寸：185×260　　**印　　张**：20.25　　**字　　数**：342千字

书　　号：ISBN 978-7-5083-4681-6

版　　次：2007年1月北京第1版

印　　次：2007年1月第1次印刷

印　　数：0001—4000册

定　　价：36.00元

前　言

汇编语言是能够充分发挥计算机所有硬件特性并能直接控制硬件的最快、最有效的语言，在对程序的时空效率要求很高及需要直接控制硬件的许多应用场合是必不可少的。因此，“汇编语言程序设计”历来是我国高等院校计算机专业必修的一门主干课程，也是自动控制等相关专业的一门重要基础课程。

考虑到国内的微型计算机广泛使用 Intel 的 80x86 / Pentium 系列或兼容的微处理器，所以本书以 80x86 / Pentium 系列微处理器为基础，系统地介绍了汇编语言程序设计的基础知识、程序设计的方法和应用技术。

本书遵循了汇编语言程序设计课程教学大纲的要求，将内容分为 9 章。第 1 章讲述学习汇编语言的基础知识。第 2 章讲述 80x86 微处理器的功能结构、寄存器结构、存储器组织及寻址方式。第 3 章讲述汇编语言语句、基本伪指令、汇编语言源程序的基本结构和汇编语言程序的上机过程。第 4 章讲述 80x86 的指令系统和基本的程序设计方法，并将指令的介绍融于相关程序设计方法的介绍之中，主要内容包括：数据传送程序、算术运算程序、逻辑运算程序、分支程序、循环程序和字符串操作程序的设计。第 5 章主要讲述模块化程序设计方法和子程序的参数传递方法。第 6 章讲述宏汇编、重复汇编和条件汇编等高级汇编技术。第 7 章、第 8 章重点讲述中断处理程序的设计和输入 / 输出程序的设计。第 9 章讲述保护模式下汇编语言程序设计的基本方法和应用技术，主要内容包括保护模式下的存储管理、实模式和保护模式之间的切换、保护模式下的中断、异常和 I/O 保护等。

本教材在编写过程中结合了作者长年从事汇编课程教学的经验，具有以下特点：

（1）本书充分考虑了目前普通高校学生知识、能力、素质的特点和实际教学情况，难点分散、循序渐进、深入浅出、实例丰富，注重教材的实用性，适应当前高校课程课时数压缩的特点，内容精炼，易教、易学。

（2）随着微型计算机技术的迅猛发展，32 位汇编语言程序设计技术也有了新的发展。本书较全面地介绍了 80x86 / Pentium 的 32 位汇编语言程序的设计方法和新技术。

（3）摒弃了以语言为中心的讲述方式，而采用以程序设计为主线的介绍方式，结合国外教材的特点，将指令融于相关程序设计方法的介绍之中，做到“学一点、用一点、巩固一点”，循序渐进，不再强调指令系统本身的完整性介绍。

（4）精选实例，力求紧扣重点，由易到难，通过典型例题使学生充分理解汇编语言程序设计的特点，把小粒度的众多知识点融化在应用实例中。

本书的编写得到了许多同事的鼓励和帮助，海军工程大学的李娟老师参加了部分章节的编写和校对工作，在此我们向他们表示衷心的感谢。另外，书末所列的参考文献及有关资料给了我们极大的帮助和启发，在此我们向与这些文献及资料有关的专家一并表示由衷的谢意。

由于时间仓促及作者水平所限，书中难免有疏漏或不妥之处，敬请广大读者批评指正。

编　者

2006 年 10 月

目　录

第1章 基础知识

本章介绍计算机中数值数据和非数值数据的表示、汇编语言中的基本数据类型以及基本逻辑运算的概念和规则，并对机器语言、汇编语言、高级语言之间的优缺点和使用场合作相应比较。

要求熟练掌握各种进位计数制之间的互相转换、补码表示法、补码的运算、常用 ASCII 码及几种常用逻辑运算规则。

1.1 数据表示与类型

数据是计算机处理的对象。从外部形式来看，计算机可处理数值、文字、图、声音、视频，甚至各种模拟信息量。这些形式的信息，在计算机系统内部，可表示成文件、图、表、树、阵列、队列、链表、栈、向量、串、实数、整数、布尔数、字符等。当然，硬件能够直接识别，能被指令系统直接调用的只是一些常用的、简单的几种基本数据类型，如字节、字、字符串等。而那些复杂的数据类型则是由基本数据类型按某种结构描述方式在软件中实现的，是数据结构研究的问题。

1.1.1 数值数据表示

输入计算机内部的数据若有确定的值，即在数轴上能找到其对应的点，则称为数值数据。也就是说，数值数据是表示数量多少和数值大小的数据。

确定一个数值数据需要考虑进位计数制和编码表示。进位计数制决定了表示数据可使用的基本符号个数，数的编码决定了数的正负号处理方式。

日常生活中，常使用十进制数表示数值数据，例如 6.18、-127 等等，但这种形式的数据在计算机内部难以直接存储、运算和传输。在计算机内部，数值是用二进制来表示的。但是二进制对于人们阅读、书写和记忆都是很不方便的。因此，为了便于人们对二进制数的描述，常选择易于与二进制数相互转换的八进制数和十六进制数。

计算机内部的数据除了数值部分外，还有符号部分，基于对正负号的不同处理方式，有 3 种编码方法，即原码、补码、反码。

1. 进位计数制及其各进位制数之间的转换

日常生活中我们最熟悉十进制数，一个数用 10 个不同的符号（0, 1, 2,…, 9）来表示，每一个符号处于十进制数中不同位置时，它所代表的实际数值不一样。例如，2585.62 代表的实际值是

$$2\times10^3+5\times10^2+8\times10^1+5\times10^0+6\times10^{-1}+2\times10^{-2}$$

一般地，任意一个十进制数：

$$D=d_nd_{n-1}\cdots d_1d_0d_{-1}d_{-2}\cdots d_{-m}\ (m, n \text{ 为正整数})$$

其值应为：

$$V(D) = d_n\times10^n + d_{n-1}\times10^{n-1} +\cdots+ d_1\times10^1 + d_0\times10^0 + d_{-1}\times10^{-1} + d_{-2}\times10^{-2}+\cdots+d_{-m}\times10^{-m}$$

其中的 $d_i(i=n,n-1,\cdots,1,0,-1,-2,\cdots,-m)$可以是 0,1,2,3,4,5,6,7,8,9 这十个数字符号中的任何一个，“10”称为基数，它代表不同数字符号的个数。10^i称为第 i 位上的权。在十进制数进行运算时，每位计满十之后就要向高位进一，即日常所说的“逢十进一”。

二进制和十进制类似，它的基数是 2，只使用两个不同的数字符号（0 和 1），与十进制数不同，运算时采用“逢二进一”的规则，第 i 位上的权是 2^i。例如，二进制数$(100101.01)_2$代表的实际值是：

$$(100101.01)_2 = 1\times2^5 + 0\times2^4+ 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0+ 0\times2^{-1} + 1\times2^{-2}=(37.25)_{10}$$

一般地，一个二进制数

$$B=b_nb_{n-1}\cdots b_1b_0b_{-1}b_{-2}\cdots b_{-m}$$

其值应为：

$$V(B) = b_n\times2^n + b_{n-1}\times2^{n-1} +\cdots+ b_1\times2^1 + b_0\times2^0 + b_{-1}\times2^{-1} + b_{-2}\times2^{-2}+\cdots+b_{-m}\times2^{-m}$$

其中的 $b_i(i=n, n-1,\cdots,1,0,-1,-2,\cdots-m)$只可以是 0 和 1 两种不同的数字。

一般地说，在某个数字系统中，若采用 R 个基本符号（0, 1, 2,…, R−1）表示各位上的数字，则称其为基 R 数制（或称 R 进制数字系统），R 被称为该数字系统的基，采用“逢 R 进一”的运算规则，对于每一个数位 i，其该位上的权为 R^i。

在计算机系统中，常用的几种进位计数制有下列几种，这些进位制数之间的对应关系如表 1-1 所示。

二进制　R=2，基本符号为 0 和 1

八进制　R=8，基本符号为 0,1,2,3,4,5,6,7

十进制　R=10，基本符号为 0,1,2,3,4,5,6,7,8,9

十六进制 R=16，基本符号为 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

表 1-1　4 种进位制数之间的对应关系

二进制数	八进制数	十进制数	十六进制数
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C

续表

二进制数	八进制数	十进制数	十六进制数
1101	15	13	D
1110	16	14	E
1111	17	15	F

从表 1-1 中可看出，十六进制系统的前 10 个数字与十进制系统中的相同，后 6 个基本符号 A, B, C, D, E, F 的值分别为十进制的 10, 11, 12, 13, 14, 15。在书写时可使用后缀字母标识该数的进位计数制，一般用 B（Binary）表示二进制，用 O（Octal）表示八进制，用 D（Decimal）表示十进制（十进制数的后缀可以省略），而字母 H（Hexadecimal）则是十六进制数的后缀，例如：二进制数 10011B，十进制数 56D 或 56，十六进制数 308FH、3C.5H 等等。

计算机内部所有的信息采用二进制编码表示。但在计算机外部，为了书写和阅读的方便，大都采用八、十或十六进制表示形式。因此，计算机在数据输入后或输出前都必须实现这些进位制数之间的转换。

（1）R 进制数转换成十进制数。

任何一个 R 进制数转换成十进制数时，只要“按权展开”即可。

例 1.1 二进制数转换成十进制数。

解：$(10101.01)_2=(1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2})_{10}=(21.25)_{10}$

例 1.2 八进制数转换成十进制数。

解：$(307.6)_8=(3\times8^2+7\times8^0+6\times8^{-1})_{10}=(199.75)_{10}$

例 1.3 十六进制数转换成十进制数。

解：$(3A.C)_{16}=(3\times16^1+10\times16^0+12\times16^{-1})_{10}=(58.75)_{10}$

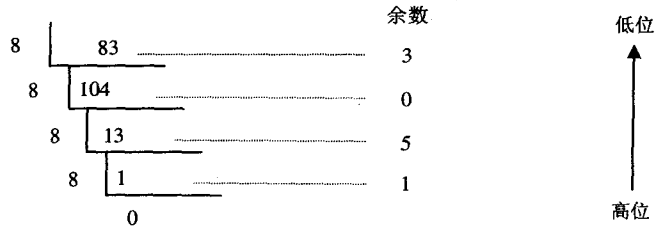
（2）十进制数转换成 R 进制数。

任何一个十进制数转换成 R 进制数时，要将整数和小数部分分别进行转换。

整数部分的转换方法是“除基取余，上右下左”。也就是说，用要转换的十进制整数去除以基数 R，将得到的余数作为结果数据中各位的数字，直到商为 0 为止。上面的余数（即先得到的余数）作为右边的低位数位，下面的余数作为左边的高位数位。

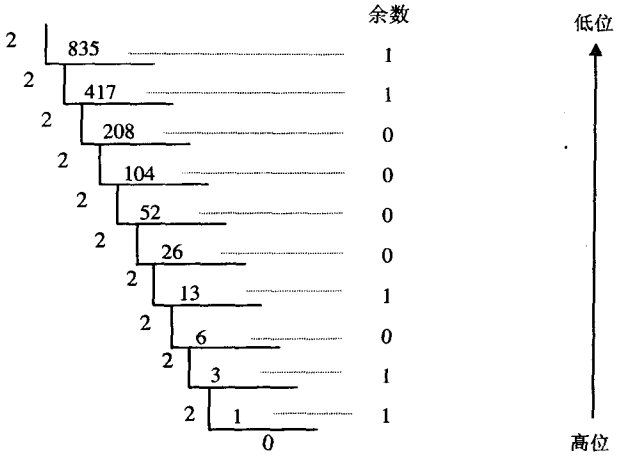
例 1.4 将十进制整数 835 分别转换成二、八进制数。

解：①转换成八进制数



所以，$(835)_{10}=(1503)_8$

②转换成二进制数



所以，$(835)_{10}=(1101000011)_2$

小数部分的转换方法是“乘基取整，上左下右”。也就是说，用要转换的十进制小数去乘以基数 R，将得到的乘积的整数部分作为结果数据中各位的数字，小数部分继续与基数 R 相乘。依次类推，直到某一步乘积的小数部分为 0 或已得到希望的位数为止。最后，将上面的整数部分作为左边的高位数位，下面的整数部分作为右边的低位数位。在进行转换的过程中，可能乘积的小数部分总得不到 0，即转换得到希望的位数后还有余数，这种情况下得到的是近似值。

例 1.5 将十进制小数 0.6875 分别转换成二、八进制数。

解：①转换成二进制数

0.6875×2=1.375　　整数部分=1　　(高位)

0.375×2=0.75　　整数部分=0　　↓

0.75×2=1.5　　整数部分=1　　↓

0.5×2=1.0　　整数部分=1　　(低位)

所以，$(0.6875)_{10}=(0.1011)_2$

② 转换成八进制数

0.6875×8=5.5　　整数部分=5　　(高位)

0.5×8=4.0　　整数部分=4　　↓

所以，$(0.6875)_{10}=(0.54)_8$　　(低位)

例 1.6 将十进制小数 0.63 转换成二进制数。

解：0.63×2=1.26　　整数部分=1　　(高位)

0.26×2=0.52　　整数部分=0　　↓

0.52×2=1.04　　整数部分=1　　↓

0.04×2=0.08　　整数部分=0　　(低位)

所以，$(0.63)_{10}=(0.1010)_2$　　(近似值)

例 1.7 将十进制数 835.6875 转换成二、八进制数。

解：$(835.6875)_{10}=(1101000011.1011)_2=(1503.54)_8$

（3）二、八、十六进制数的相互转换。

1）八进制数转换成二进制数。

八进制数转换成二进制数的方法很简单，只要把每一个八进制数字改写成等值的3位二进制数即可，且保持高低位的次序不变。八进制数字与二进制数的对应关系如下：

$(0)_8$=000　$(1)_8$=001　$(2)_8$=010　$(3)_8$=011

$(4)_8$=100　$(5)_8$=101　$(6)_8$=110　$(7)_8$=111

例 1.8　将$(13.724)_8$转换成二进制数。

解：$(13.724)_8$=(001 011 . 111 010 100)$_2$=$(1011.1110101)_2$

2）十六进制数转换成二进制数。

十六进制数转换成二进制数的方法与八进制数转换成二进制数的方法类似，只要把每一个十六进制数字改写成等值的4位二进制数即可，且保持高低位的次序不变。十六进制数字与二进制数的对应关系如下：

$(0)_{16}$=0000　$(1)_{16}$=0001　$(2)_{16}$=0010　$(3)_{16}$=0011

$(4)_{16}$=0100　$(5)_{16}$=0101　$(6)_{16}$=0110　$(7)_{16}$=0111

$(8)_{16}$=1000　$(9)_{16}$=1001　$(A)_{16}$=1010　$(B)_{16}$=1011

$(C)_{16}$=1100　$(D)_{16}$=1101　$(E)_{16}$=1110　$(F)_{16}$=1111

例 1.9　将十六进制数 2B.5E 转换成二进制数。

解：$(2B.5E)_{16}$ = (0010 1011 . 0101 1110) $_2$ = $(101011.0101111)_2$

3）二进制数转换成八进制数。

二进制数转换成八进制数时，整数部分从低位向高位方向每3位用一个等值的八进制数来替换，最后不足3位时在高位补0凑满3位；小数部分从高位向低位方向每3位用一个等值的八进制数来替换，最后不足3位时在低位补0凑满3位。

例 1.10　将二进制数转换成八进制数。

解：$(0.10101)_2$ = (000 . 101 010) $_2$ = (0.52) $_8$

$(10011.01)_2$ = (010 011 . 010) $_2$ = $(23.2)_8$

4）二进制数转换成十六进制数。

二进制数转换成十六进制数时，整数部分从低位向高位方向每4位用一个等值的十六进制数来替换，最后不足4位时在高位补0凑满4位；小数部分从高位向低位方向每4位用一个等值的十六进制数来替换，最后不足4位时在低位补0凑满4位。

例 1.11　将二进制数转换成十六进制数。

解：$(11001.11)_2$ = (0001 1001 . 1100) $_2$ = (19.C) $_{16}$

从以上可以看出，二进制数与八进制数、十六进制数有很简单直观的对应关系。二进制数太长，书写、阅读均不方便；八进制数和十六进制数却像十进制数一样简练，易写易记。计算机中只使用二进制一种计数制，并不使用其他计数制。但为了开发程序、调试程序、阅读机器内部代码时的方便，人们经常使用八进制或十六进制来等价地表示二进制，所以大家也必须熟练掌握八进制和十六进制。

2. 编码系统

一个正数的所有编码表示（如：原码、补码、反码）都是相同的。其结果总是符号位取值为0，数值部分取值为真值中对应的数值部分。下面，分别介绍负数的3种编码表示。

（1）原码表示法。

原码表示法也称“符号—数值”表示法，也就是说原码表示的机器数由符号位后直接跟上

真值的数值构成。即任意一个数的原码的数值部分都同其真值的数值部分。因此，原码表示法中，正数和负数的编码表示结果仅符号位不同，数值部分完全相同，都取真值的数值部分。总结负数的原码表示法的编码规则如下：

① 定点负整数：$[X_T]_{原}= 1\times2^n + |X_T|$ $(-2^n < X_T \leqslant 0$，n 为原码数值部分的位数)

② 定点负小数：$[X_T]_{原}=1+|X_T|$ $(-1<X_T\leqslant 0)$

原码 0 有两种表示形式：$[+0]_{原}=0\ 00\cdots0$

$[-0]_{原}=1\ 00\cdots0$

原码表示的优点是与真值的对应关系直观、方便，因此与真值的转换简单，并且用原码实现乘除运算比较简便，但其加减法运算规则复杂。在进行原码的加减运算过程中，要判定是否是两个异号数相加或两个同号数相减，若是，则必须判定两个数的绝对值的大小，并根据判断结果决定结果符号。因此，在计算机中一般用补码实现加减运算。

（2）补码表示法。

假定补码的位数为 n（其中符号占 1 位，数值部分占 n-1 位），则负数补码表示的定义如下：

①定点负整数：$[X_T]_{补}= 2^n - |X_T|$ $(-2^{n-1} \leqslant X_T \leqslant 0$，$\mathrm{mod}\ 2^n)$

②定点负小数：$[X_T]_{补}=2- |X_T|$ $(-1\leqslant X_T\leqslant 0$，$\mathrm{mod}\ 2)$

补码 0 的表示是唯一的：$[+0]_{补}=0\ 0\cdots0$

$[-0]_{补}= 2^n-0=0\ 0\cdots0 \quad (\mathrm{mod}\ 2^n)$

例 1.12 设补码的位数为 n，求负数-2^{n-1} 的补码表示。

解：由上述定义，$[-2^{n-1}]_{补}= 2^n-2^{n-1}=2^{n-1}=1\ 0\cdots0$ (n-1 个 0)

例 1.13 设补码的位数为 n，求负数-1 的补码表示。

解：从上述定义来看，对于整数补码有：$[-1]_{补}= 2^n-1=11\cdots1$ (n 个 1)

对于小数补码有：$[-1]_{补}=2-1=1.00\cdots0$ (n-1 个 0)

由此可见，“-1”既可在整数范围内表示，也能在小数范围内表示，在计算机中有两种不同的补码表示。每个编码的小数点位置在约定好的默认位置处，因此，上述“-1”的补码小数表示中的小数点“.”在计算机内实际上是不存在的，只是书写时为了将小数编码和整数编码区别开来而加上去的。因此，“-1”的补码小数表示与“-2^{n-1}”的补码表示结果相同，都是符号位为 1，数值部分为 n-1 个 0。“-1”与“-2^{n-1}”分别是补码小数和补码整数中可表示的最小负数。

例 1.14 设补码的位数为 6，求负数-0.10110 的补码表示。

解：$[-0.10110]_{补}=2-0.10110=10.00000-0.10110=1.01010$

上面例子用补码的定义求某个负数的补码，实际上可以用以下简单方法求负数的补码：“符号位固定为 1，其余各位由真值中相应各位取反后，末尾加 1 所得。”

由此，得到由补码求真值的简便方法：“若符号位为 1，则真值的符号为负，其数值部分的各位由补码中相应各位取反后，末尾加 1 所得。”

进一步可以得到求一个补码“取负”后的补码表示方法：“只要对该已知补码各位取反，末尾加 1 即可。”这里要注意最小负数取负后会发生溢出，即最小负数取负后的补码表示不存在。

例 1.15 已知：$X_T=-0.1011010$，求$[X_T]_{补}$。

解：$[X_T]_{补}=1.0100101+2^{-7}=1.0100101+0.0000001=1.0100110$

例 1.16 已知：$[X_T]_{补}$ = 1 011010，求 X_T 。

解：X_T=−100101+1=−100110

例 1.17 已知：$[X_T]_{补}$=1 011010，求$[-X_T]_{补}$。

解：$[-X_T]_{补}$=0 100101+1=0 100110

（3）反码表示法。

负数的补码可采用“各位求反，末尾加 1”的方法得到，如果仅各位求反而末尾不加 1，那么就可得到负数的反码表示，因此负数反码的定义就是在相应的补码表示中再末尾减 1。

假定反码的位数为 n（其中符号占 1 位，数值部分占 n−1 位），则负数的反码表示的定义如下：

①定点负整数：$[X_T]_{反}=(2^n-1)-|X_T|$ $(-2^{n-1}<X_T\leqslant 0,\ \mathrm{mod}\ 2^n-1)$

②定点负小数：$[X_T]_{补}=(2-2^{-(n-1)})-|X_T|$ $(-1<X_T\leqslant 0,\ \mathrm{mod}\ 2-2^{-(n-1)})$

反码表示的零有两种：$[+0]_{反}$=0 0…0

$[-0]_{反}$= 1 1…1

从公式中可看出：反码就是以$(R^n-R^{-(m-1)})$为模的补码（m 为小数部分的位数）。若是二进制数（即 R=2），那么，当为 n 位定点整数编码（即 m=0）时，模为(2^n-1)；当为 m 位定点小数编码（即 n=1）时，模为$(2-2^{-(m-1)})$。反码运算是在以$(R^n-R^{-(m-1)})$为模的条件下进行的。所以，

$R^n-R^{-(m-1)} \equiv 0(\mathrm{mod}\ R^n-R^{-(m-1)}) \Rightarrow R^n \equiv R^{-(m-1)} \quad (\mathrm{mod}(R^n-R^{-(m-1)}))$

这意味着若运算中产生了 R^n（即最高位有进位），就必须把它加到末位上去，这叫“循环进”。

例 1.18 用反码运算来计算 9828.45−1928.45=？。

解：9828.45−1928.45=9828.45+(10^4-10^{-2}−1928.45)

=9828.45+8071.54

```
   9828.45
+) 8071.54
-------------
  [1]7899.99
        1
-------------
   7900.00
```

1.1.2 BCD 码数据表示

有些计算机内还同时采用一种用二进制编码的十进制数来表示数值数据，以便用在一些要求直接用十进制数进行计算的场合。这种十进制数用二进制编码的形式被称为 BCD 码（Binary Coded Decimal）。许多计算机都有专门的十进制运算指令，并有专门的逻辑线路在 BCD 码运算时使每 4 位二进制数按十进制处理。

每位十进制数的取值可以是 0,1,2,…,9 这 10 个数之一，因此，每一个十进制数位必须至少有 4 位二进制位来表示。而 4 位二进制位可以组合成 16 种状态，去掉 10 种状态后还有 6 种冗余状态，所以从 16 种状态中选取 10 种状态表示十进制数位 0～9 的方法很多，可以产生多种 BCD 码。但常用的编码是 8421BCD 编码，如表 1-2 所示。这种 BCD 编码实际上就是 0～9 的“等值”二进制数。

表 1-2　8421BCD 编码列表

十进制数字	8421BCD 码	十进制数字	8421BCD 码
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

用 BCD 码进行进制的转换时，是要求在两种进制的表现形式上快速转换，而不是要求在“数值相等”的含义快速转换。

例 1.19　求十进制数 2000 的 BCD 编码和其二进制数。

解：2000 的 BCD 编码是把每位上的数分别转换为其对应的 BCD 编码：

0010、0000、0000 和 0000

把它们合在一起就是 2000 的 BCD 编码：0010 0000 0000 0000。

1.1.3　非数值数据表示

计算机除了具有进行数值计算能力之外，还具有进行非数值计算的能力。现在，后者的应用领域已远远超过了前者的应用领域，如文字处理、图形图像处理、信息检索和日常的办公管理等。所以，对非数值信息的编码就显得越加重要。

1. 逻辑数据

正常情况下，每个字或其他可寻址单位（字节、半字等）是作为一个整体数据单元看待的。但是，某些时候还需要将一个 n 位数据看成是由 n 个 1 位项组成，每项取值为 0 或 1。当数据以这种方式看待时，就被认为是逻辑数据。因此 n 位二进制数可表示 n 个逻辑数据，逻辑数据只能参加逻辑运算，按位进行，如按位“与”、按位“或”、逻辑左移、逻辑右移等。

这种按位排列和运算的方式在有些情况下很有效。例如，有时需要存储一个布尔或二进制数据项阵列，阵列中的每项只能取值为 1（真）或 0（假）；还有的时候可能需要提取一个数据项中的某些位进行诸如“置位”或“清 0”等操作。

逻辑数据和数值数据在形式上无任何差异，都是一串 0/1 序列，因此机器本身不能识别是逻辑数据还是数值数据，是靠指令的操作码类型来识别的。例如，逻辑运算指令处理逻辑数据，算术运算指令处理的是数值数据。所以有时同样的一个数据，可能既被看成是数值数据，又被看成是逻辑数据，主要看处理它的是哪种指令。

2. 西文字符

西文是由拉丁字母、数字、标点符号及一些特殊符号所组成的，它们统称为“字符”（character）。所有字符的集合叫做“字符集”。字符不能直接在计算机内部进行处理，因而也必须对其进行数字化编码，字符集中每一个字符都有一个代码（二进制编码的 0/1 序列），这些代码具有唯一性，互相区别，构成了该字符集的代码表，简称码表。

字符集有多种，每一个字符集的编码方法也多种多样。字符主要用于外部设备和计算机之间交换信息。一旦确定了所使用的字符集和编码方法后，计算机内部所表示的二进制代码和外部设备输入、打印和显示的字符之间就有唯一的对应关系。

目前计算机中使用得最广泛的西文字符集及其编码是 ASCII 码，即美国标准信息交换码（American Standard Cord for Information Interchange），ASCII 字符编码见附录 A。

ASCII 码表中，每个字符都由 7 个二进位 $b_6b_5b_4b_3b_2b_1b_0$ 表示，其中 $b_6b_5b_4$ 是高位部分，$b_3b_2b_1b_0$ 是低位部分。一个字符在计算机中实际上是用 8 位表示的。一般情况下，最高一位 b_7 为“0”。在需要奇偶校验时，这一位可用于存放奇偶校验值，此时称这一位为奇偶校验位。7 个二进位 $b_6b_5b_4b_3b_2b_1b_0$ 从 0000000～1111111 共表示 128 种编码，可用来表示 128 个不同的字符，其中包括 10 个数字、26 个小写字母、26 个大写字母、算术运算符、标点符号、商业符号等。表中共有 95 个可打印（或显示）字符和 33 个控制字符。这 95 个可打印（或显示）字符在计算机键盘上能找到相应的键，按键后就可将对应字符的二进制编码送入计算机内。表中第 0 列和第 1 列以及第 7 列最末一个字符（DEL）称为控制字符，共 33 个，它们在传输、打印或显示输出时起控制作用。

ASCII 字符编码有两个规律：

（1）字符 0～9 这 10 个数字字符的高三位编码为 011，低四位分别为 0000～1001。当去掉高三位时，低四位正好是二进制形式的 0～9。这样既满足了正常的排序关系，又有利于实现 ASCII 与二进制数之间的转换。

（2）英文字母字符的编码值也满足正常的字母排序关系，而且大、小写字母的编码之间有简单的对应关系，差别仅在 b_5 这一位上，若这一位为 0，则是大写字母；若为 1，则是小写字母。这使得大、小写字母之间的转换非常方便。

西文字符集的编码不止 ASCII 码一种，较常用的还有一种是用 8 位二进制数表示一个字符的 EBCDIC 码（Extended Binary Coded Decimal Interchange Code），该码共有 256 个编码状态，最多可表示 256 个字符，但这 256 个编码并没有全部被用来表示字符。0～9 十个数字字符的高 4 位编码为 1111，低四位还是分别对应 0000～1001。大、小写英文字母字符的编码也是仅一位（第 2 位）不同。

3. 汉字字符

西文是一种拼音文字，用有限的几个字母可以拼写出所有单词。因此西文中仅需要对有限个少量的字母和一些数学符号、标点符号等辅助字符进行编码，所有西文字符集的字符总数不超过 256 个，所以使用 7 个或 8 个二进位就可表示。中文信息的基本组成单位是汉字，汉字也是字符。但汉字是表意文字，一个字就是一个方块图形。计算机要对汉字信息进行处理，就必须对汉字本身进行编码，但汉字的总数超过 6 万字，数量巨大，给汉字在计算机内部的表示、汉字的传输与交换、汉字的输入和输出等带来了一系列问题。为了适应汉字系统各组成部分对汉字信息处理的不同需要，汉字系统必须处理以下几种汉字代码：输入码/内码/字模点阵码。

（1）汉字的输入码。

由于计算机最早是由西方国家研制开发的，最重要的信息输入工具——键盘是面向西文设计的，一个或两个西文字符对应着一个按键，非常方便。但汉字是大字符集，专门的汉字输入键盘由于键多、查找不便、成本高等原因而几乎无法采用。怎样向计算机输入汉字呢？一种是手写汉字联机识别输入，或者是印刷汉字扫描输入后自动识别，这两种方法现均已达到实用水平。现在还有一种用语音输入汉字的方法，虽然简单易操作，但离实用阶段还相差很远。目前来说，最简便、最广泛采用的汉字输入方法是利用英文键盘输入汉字。由于汉字字数多，无法使每个汉字与西文键盘上的一个键相对应，因此必须使每个汉字用一个或几个键来表示，这种

对每个汉字用相应的按键进行的编码表示就称为汉字的“输入码”，又称外码。因此汉字的输入码的码元（即组成编码的基本元素）一定是西文键盘中的某个按键。

汉字的输入编码方案有几百种之多。能够被广泛接受的编码方案应具有下列特点：易学习、易记忆、效率高（击键次数较少）、重码少、容量大（包含汉字的字数多）等等。到目前为止，还没有一种在所有方面都很好的编码方法，真正推广应用较好的也只有少数几种。汉字输入编码方法大体分成 4 类。①数字编码：用一串数字来表示汉字的编码。例如电报码、区位码等，它们难以记忆，不易推广。②字音编码：基于汉语拼音的编码。简单易学，适合于非专业人员。缺点是同音字引起的重码多，需增加选择操作。例如，现在常用的微软拼音输入法和智能 ABC 输入法等。③字形编码：将汉字的字形分解归类而给出的编码。重码少、输入速度快，但编码规则不易掌握。例如，五笔字形法和表形码就是这类编码。④形音编码：将汉字读音和形状结合起来考虑的编码，它吸取了字音编码和字形编码的优点，使编码规则简化、重码减少，但不易掌握。

（2）字符集与汉字内码。

汉字通过输入码从键盘或通过语音识别从麦克风或通过联机手写或印刷体文字扫描输入等各种手段被输入到计算机内部后，就按照一种称为“内码”的编码形式在系统中进行存储、查找、传送等处理。对于西文字符数据，它的内码就是 ASCII 码。对于汉字内码的选择，我们必须考虑以下几个因素：①不能有二义性，即不能和 ASCII 码有相同的编码。②要与汉字在字库中的位置有关系，以便于汉字的处理、查找。③编码应尽量短。

为了适应计算机处理汉字信息的需要，1981 年我国颁布了 GB2312-80《信息交换用汉字编码字符集·基本集》。该标准选出 6763 个常用汉字，为每个汉字规定了标准代码，以供汉字信息在不同计算机系统之间交换使用。这个标准称为国标码，又称国标交换码。

GB2312 国标字符集由 3 部分组成：第一部分是字母、数字和各种符号，包括英文、俄文、日文平假名与片假名、罗马字母、汉语拼音等共 687 个；第二部分为一级常用汉字，共 3755 个，按汉语拼音排列；第三部分为二级常用字，共 3008 个，因不太常用，所以按偏旁部首排列。

GB2312 国标字符集中为任意一个字符（汉字或其他字符）规定了一个唯一的二进制代码。码表由 94 行（十进制编号 0～93 行）、94 列（十进制编号 0～93 列）组成，行号称为区号，列号称为位号。每一个汉字或符号在码表中都有各自的位置，因此各有一个唯一的位置编码，该编码用字符所在的区号及位号的二进制代码表示，7 位区号在左、7 位位号在右，共 14 位，这 14 位代码就叫该汉字的“区位码”。因此区位码指出了该汉字在码表中的位置。

汉字的区位码并不是其国标码（即国标交换码）。由于信息传输的原因，每个汉字的区号和位号必须各自加上 32（即 16 进制的 20h），这样区号和位号各自加上 32 后的相应的二进制代码才是它的“国标码”，因此在“国标码”中区号和位号还是各自占 7 位。在计算机内部，为了处理与存储的方便，汉字国标码的前后各 7 位分别用一个字节来表示，所以共需两个字节才能表示一个汉字。因为计算机中的中西文信息是混合在一起进行处理的，所以汉字信息如不予以特别的标识，则它与单字节的 ASCII 码就会混淆不清，无法识别。这就是前面给出的第一个要考虑的因素。为了解决这个问题，采用的方法之一，就是使表示汉字的两个字节的最高位(b_7)总等于“1”。这种双字节（16 位）的汉字编码就是其中的一种汉字“机内码”（即汉字内码）。目前 PC 机中汉字内码的表示大多数都是这种方式。例如：汉字“大”的国标码为：3473h（0011 0100 0111 0011B），前面的 34h 和字符“4”的 ACSII 码相同，后面的 73h 和字符“s”