高等学校非计算机专业学生计算机二级考试用书

计算机基础

汪大菊　赵玉香　匙彦斌

天津大学出版社

高等学校非计算机专业学生
计算机二级考试用书

计 算 机 基 础

汪大菊　赵玉香　匙彦斌

天津大学出版社

内 容 提 要

本书是按照天津市普通高等学校非计算机专业学生计算机二级考试大纲要求编写的。其主要内容包括计算机硬件基础知识、数据结构、操作系统和数据库的基础知识、软件开发技术及语言处理基础知识等，涵盖了二级考试大纲对基础部分要求的全部内容。

计算机基础

汪大荫　赵玉香　匙彦斌

*

天津大学出版社出版

（天津大学内）

邮编：300072

衡水地区印刷厂排版

永清县第一胶印厂印刷

新华书店天津发行所发行

*

开本：850×1168 毫米 1/32 印张：9.5 字数：245 千

1995 年 8 月第一版　1995 年 8 月第一次印刷

印数：1--3000

ISBN 7-5618-0742-2

TP·61　定价：10.00 元

前　言

本书是按照天津市普通高等学校非计算机专业学生计算机二级考试大纲要求编写的，可作为应考人员的综合性辅导教材。其主要内容包括计算机硬件基础知识、数据结构、操作系统和数据库的基础知识、软件开发技术及语言处理基础知识等，涵盖了二级考试大纲对基础部分要求的全部内容。

本书第一、二章由汪大菊编写，第三、四章由赵玉香编写，第五、六章由匙彦斌编写。在本书编辑出版过程中得到了天津大学出版社的大力支持，在此谨表衷心感谢。书中错误、不当之处，恳请广大读者指正。

编　者

1994 年 4 月

目　录

第一章　硬件基础知识

一个完整的计算机系统应包括硬件部分和软件部分。只有硬件部分和软件部分相结合，计算机才能发挥应有的作用。计算机的硬件是指组成计算机实体的电子线路和物理装置，是计算机的物质基础。计算机的规模大小不同，在硬件配置上差别很大，但硬件基础知识是带有共性的。这就是本章所要讨论的主要内容。

§1　数制及其转换

数据在计算机中是以二进制形式存放和表示的，并且以二进制形式对数据进行各种运算。但若要求人也用二进制形式向计算机输入数或计算机以二进制的形式显示给人时，就会使人感到很不方便，因此在人-机交换数据的时候，一般使用人们所熟悉的十进制或八进制、十六进制表示。同一个数可以有不同的表示形式，它们之间可以相互转换。

一、进位计数制

无论是人们熟悉的十进制数还是计算机使用的二进制数，其共同之处都是进位计数制。一般来说，数制若采用 R 个基本符号，则称为 R 进制。例如：

十进制：R＝10，所用到的数码为 0，1，2，…9。

二进制：R＝2，所用到的数码为 0，1。

八进制：R＝8，所用到的数码为 0，1，2…7。

十六进制：R＝16，所用到的数码为 0，1，2，…9，A，B，…F。

进位计数制的编码符合逢 R 进位的规则，即当一个数的每一位计满 R 后，便向高位进位。

对任意 R 进制数 N，都可以写成：

$N=D_{n-1}\cdot R^{n-1}+D_{n-2}\cdot R^{n-2}+\cdots+D_0\cdot R^0+D_{-1}\cdot R^{-1}+D_{-2}\cdot R^{-2}+\cdots+D_{-m}\cdot R^{-m}$，其中 m，n 为正整数，$D_i$ 取 0 到 R－1 之间的任意数，R^i 称为该位数的权值，该式称为 R 进位计数的按权展开式。

例 1 二进制数 S＝1101.01，按权展开为：

$S=1\times2^3+1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}$

例 2 八进制数 A＝3207.5，按权展开为：

$A=3\times8^3+2\times8^2+0\times8^1+7\times8^0+5\times8^{-1}$

二、进位计数制之间的转换

1. R 进制转换成十进制

常用方法有两种：

1）按权转换　把任意 R 进制数写成按权展开式，然后求和，得到的就是 R 进制数对应的十进制数。

例如，将 $S=(1101.01)_2$ 转换成十进制。

$$\begin{aligned}S&=1\times2^3+1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}\\&=8+4+0+1+0+0.25\\&=13.25\end{aligned}$$

二进制数 1101.01 转换成十进制数为 13.25。

又如，将 $(732.6)_8$ 转换成十进制。

$$\begin{aligned}(732.6)_8&=7\times8^2+3\times8^1+2\times8^0+6\times8^{-1}\\&=7\times64+3\times8+2+6\times0.125\\&=(474.75)_{10}\end{aligned}$$

2）基数重复相乘相除法　该方法是将要转换数的整数部分和小数部分分别转换，整数部分用重复相乘，小数部分用重复相除。例如，N 为二进制数，$N=N_{整}\cdot N_{小}$。设整数部分为四位，可以写成：

$$\begin{aligned}N_{整}&=d_3\cdot2^3+d_2\cdot2^2+d_1\cdot2^1+d_0\cdot2^0\\&=[(d_3\cdot2+d_2)\cdot2+d_1]\cdot2+d_0\end{aligned}$$

小数部分为四位，可以写成：

$$N_{小}=d_{-1}\cdot 2^{-1}+d_{-2}\cdot 2^{-2}+d_{-3}\cdot 2^{-3}+d_{-4}\cdot 2^{-4}$$
$$=2^{-1}\cdot\{d_{-1}+2^{-1}\cdot[d_{-2}+2^{-1}\cdot(d_{-3}+2^{-1}\cdot d_{-4})]\}$$

即整数部分由最高位开始乘以 2，加上次高位后，再乘以 2，如此重复，直到加上最低位。小数部分则从最低位开始，除以 2，加上次低位后再除以 2，直到加上小数点后第一位，最后再除以 2。运算后分别得到转换后的十进制整数部分和小数部分。例如，

$N=(1011\cdot 011)_2$ 转换成十进制数：

$N_{整}=1011$，用重复相乘：

$$1\times 2+0=2$$
$$2\times 2+1=5$$
$$5\times 2+1=11$$

$N_{小}=.011$，用重复相除：

$$1\div 2+1=1.5$$
$$1.5\div 2+0=0.75$$
$$0.75\div 2=0.375$$

因此 $N=(1011.011)_2=(11.375)_{10}$。

以上方法同样适用于八进制和十六进制，只是将乘以 2 和除以 2 分别改为乘以 8 或 16，除以 8 或 16。

2. 十进制转换成 R 进制

任何一个十进制数均可以表示为整数部分和小数部分。十进制数需要转换成 R 进制数时，可将整数部分和小数部分分开转换。

1）整数部分转换

整数部分转换常用两种方法：

①除以 R 取余数　将整数除以 R 进制的 R，得到商 Q_1 和余数 d_1，然后用商 Q_1 再除以 R，得到商 Q_2 和余数 d_2，再用商 Q_2 除以 R，得到商 Q_3 和余数 d_3，…这样一直除下去，直到商为 0 时，得到余数 d_n，将各次求得的余数以先后次序 $d_nd_{n-1}\cdots d_3d_2d_1$ 由低向

高位排列，即得到转换的 R 进制的值。

例如，将十进制数 47 转换成二进制。

		余数
2	47	1
2	23	1
2	11	1
2	5	1
2	2	0
2	1	1
	0	

最后得到结果：$(47)_{10}=(101111)_2$。

例如，将十进制数 69 转换成十六进制。

		余数
16	69	5
16	4	4
	0	

最后得到结果：$(69)_{10}=(45)_{16}$。

②减权定位法　确定要转换的十进制的 R 次幂数，然后找对应位上 N_i 的取值。

例如，将$(47)_{10}$转换成二进制数。

$$(47)_{10}=N_5\cdot 2^5+N_4\cdot 2^4+N_3\cdot 2^3+N_2\cdot 2^2+N_1\cdot 2^1+N_0\cdot 2^0$$
$$=1\times 2^5+0\times 2^4+1\times 2^3+1\times 2^2+1\times 2^1+1$$
$$=(101111)_2$$

2）小数部分转换

常用的方法是：将十进制的小数部分乘以 R 进制的 R，所得乘积的整数部分保留，再将小数部分乘以 R，如此进行下去，直到小数部分为零，最后将各次保留的整数部分按由高到低排列，即为 R 进制的值。

例如，将$(0.6875)_{10}$转换成二进制。

$0.6875\times2=1.375$ 1

$0.375\times2=0.75$ 0

$0.75\times2=1.5$ 1

$0.5\times2=1.0$ 1

结果由高到低排列，加小数点得到：$(0.6875)_{10}=(0.1011)_2$。

例如：将$(0.6328125)_{10}$转换成八进制。

$0.6328125\times8=5.0625$ 5

$0.0625\times8=0.5$ 0

$0.5\times8=4.0$ 4

所以$(0.6328125)_{10}=(0.504)_8$。

需要注意的是，十进制小数有时不能精确地换算成其他进制的小数，此时可按精度要求取确定的小数位数。例如$(0.423)_{10}$转换成二进制：

$0.423\times2=0.846$ 0

$0.846\times2=1.692$ 1

$0.692\times2=1.384$ 1

$0.384\times2=0.768$ 0

$0.768\times2=1.536$ 1

$0.536\times2=1.072$ 1

假设取5位小数，最后得到近似值$(0.423)_{10}=(0.01101)_2$。

3. 二进制与八进制之间转换

三位二进制数所能表示的8个数为：000，001，010，…，111，正好是八进制的0～7的8个代码。因此将二进制转换成八进制数的方法是：以小数点为界，左右分别按三位一划分，不足三位补0，然后用八进制数写出。

例如，将$(10110011.0101111)_2$转换成八进制。

$$\underset{2}{\underline{010}}\ \underset{6}{\underline{110}}\ \underset{3}{\underline{011}}\ .\ \underset{2}{\underline{010}}\ \underset{7}{\underline{111}}\ \underset{4}{\underline{100}}$$

所以$(10110011.0101111)_2=(263.274)_8$。

反之，将八进制数转换成二进制时，只要将八进制数每位用三位二进制数表示出来即可。

例如：

$(67.52)_8=\underline{110}\ \underline{111}.\ \underline{101}\ \underline{010}$

4. 二进制与十六进制之间的转换

因为四位二进制数正好表示了十六进制数的16个代码，所以转换方法是：以小数点为界，左右按四位一划分，不足四位补0，然后按十六进制数的代码读出。

例如，将$(10110101011.0100111)_2$转换成十六进制。

$$\underset{5}{\underline{0101}}\ \underset{A}{\underline{1010}}\ \underset{B}{\underline{1011}}.\ \underset{4}{\underline{0100}}\ \underset{E}{\underline{1110}}$$

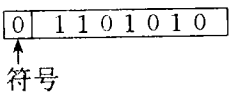
所以$(10110101011.0100111)_2=(5AB.4E)_{16}$。

反之，十六进制转换成二进制时，将十六进制数各位用四位二进制数书写出来即可。

若需要将十进制数转化成八进制或十六进制时，可以先将十进制数化成二进制，再转化成八进制或十六进制。

§2 信息编码

计算机中的数据，又称为信息，可以分为两大类：数值数据和非数值数据。数值数据有确定的值，而且有正负之分。计算机中的正负号也是用数码来表示的，通常用一位二进制位代表符号位，当该位为0时，表示正；为1时，表示负。符号位一般放在数的最高位，例如：



表示$(+1101010)_2$。计算机中连同符号位一起存放的一串数称为机器数，如上述01101010；而+1101010为机器数的真值。

在不同的机器中，机器数的表示方法不完全相同，常用的编码

有原码、反码、补码和移码。

一、数的原码、反码、补码和移码

1. 原码

原码是机器数中最简单的一种编码。用原码表示时，最高位为符号位，数值部分为原数的绝对值。

例如：

$x=+0110101 \quad [x]_{原}=00110101$

$x=-0110101 \quad [x]_{原}=10110101$

$x=0.1101011 \quad [x]_{原}=0.1101011$

$x=-0.1101011 \quad [x]_{原}=1.1101011$

设 x 为二进制小数，则 x 的原码：

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

机器数中的 0，在原码中有正 0，负 0 之分，设 $x=0$，则：

$[x]_{原}=0.0000000$

$[x]_{原}=1.0000000$

原码表示数的范围，以 8 位二进制数为例：

最大小数：0.1111111，即为 $(+\frac{127}{128})_{10}$。

最小小数：1.1111111，即为 $(-\frac{127}{128})_{10}$。

最大整数：01111111，即为 $(+127)_{10}$。

最小整数：11111111，即为 $(-127)_{10}$。

原码表示数的个数，n 位二进制位有 2^n 个不同的状态，但由于 0 有正、负之分，占去了两个编码，所以 n 位二进制位能表示 2^n-1 个原码数。例如 $n=8$，则 $2^8-1=255$，即能表示 255 个原码整数或原码小数。

2. 反码

反码表示时，符号位与原码相同，正数的反码与正数的原码相同，负数的反码符号位用 1，数值部分逐位取反。例如：

$x=+1010101$　　$[x]_{反}=01010101$

$x=-1010101$　　$[x]_{反}=10101010$

$x=0.1101101$　　$[x]_{反}=0.1101010$

$x=-0.1101101$　　$[x]_{反}=1.0010010$

$x=0.0000000$　　$[x]_{反}=0.0000000$

$x=-0.0000000$　　$[x]_{反}=1.1111111$

所以反码表示时，0 也是不唯一的。

反码表示数的范围和个数与原码相同。

3. 补码

在计算机中，表示一个数的位数（或称字长）是有限的。对于 n 位字长的机器来说，若 n 位全为 1 时，再加上 1，已经超过了机器所能表示的数的范围，此时，n 位要变为全 0，而高位的进位机器自动丢掉。这就是模数的概念。如果有 n 位整数（包括一位符号位），则它的模为 2^n，若是 n 位小数（包括一位符号位），它的模为 2，模数在计算机中是表示不出来的。

补码就是利用了模的概念。例如字长为 4 位的二进制数，做十进制 8－4 运算，用二进制表示为：1000－0100，做减法运算：

$$\begin{array}{r} 1000 \\ -0100 \\ \hline 0100 \end{array} \longrightarrow \text{结果为 4}$$

或者做运算：

$$\begin{array}{r} 1000 \\ +1100 \\ \hline 10100 \end{array}$$

↑——自动丢掉

自动丢掉后剩下的结果 0100 也是 4，结果是一样的。而上式中的 1100 就是－0100（即 4）的补码形式。因为在 4 位二进制的机器中，模为 $2^4=16$，$16+(-4)=12$，12 的二进制形式为 1100。所以一般补码的定义为：

若 n 位二进制表示整数 x，则

$$[x]_{补}=\begin{cases}x & x\geqslant 0\\ 2^n+x & x<0\end{cases}$$

若 n 位二进制表示小数 x，$x=x_0x_1x_2\cdots\cdots x_{n-1}$，其中 x_0 为符号位，小数点在 x_0 与 x_1 之间，因为小数的模为 2，因此小数 x 的补码可写为：

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

正数的补码形式上与原码相同，就是其本身。而负数的补码用模加上该负数。例如设 n 为 4 位，则：

$x=+1100\quad [x]_{补}=01100$

$x=-1100\quad [x]_{补}=2^4+(-1100)=10000-1100=0100$

$x=-0.1010\quad [x]_{补}=2+(0.1010)=10-0.1010=1.0110$

对于 0，由于 $[+0]_{补}=[-0]_{补}=0000\cdots0$，所以 0 在补码表示中是唯一的。

在实际应用中若求一个负数的补码时，常用将符号位置成 1，数值各位求反，在末位加 1 的方法实现，与用公式求完全相同。

例如，$x=-\frac{5}{16}$，求 $[x]_{补}$。

$x=-\frac{5}{16}=-0.3125$，对应的二进制数为 -0.0101。

$[x]_{反}=1.1010$

$[x]_{补}=1.1011$

也可以用 $[x]_{补}=2+(0.0101)=10-0.0101=1.1011$。

补码表示数的范围，以 n=8 位为例：

最大整数：$01111111=(+127)_{10}$。

最小整数：$10000000=(-128)_{10}$。

最大小数：$0.1111111=(+\frac{127}{128})_{10}$。

最小小数：$1.0000000=(-1)_{10}$。

在补码系统中，由于 0 是唯一的，n 位二进制数能表示 2^n 个

补码数。

在补码表示法中减法可以用加法实现，因此在计算机中用得较多。

4. 移码

移码又称增码，多用于表示浮点数中的阶码。

对于一个 n 位的二进制整数 x(包括符号位)，其移码为：

$[x]_{移}=2^{n-1}+x \qquad -2^{n-1}\leqslant x<2^{n-1}$

即当 $x>0$ 时，将 x 的最高位加 1，作为符号位，数值部分为 x 的绝对值。当 $x<0$ 时，用上式将 2^{n-1} 减去 x 的绝对值，符号位为 0，即得到 $[x]_{移}$。

例如，设 n=4 位，求以下 x 的移码。

$x=+0110 \qquad [x]_{移}=2^{4-1}+0110=1110$

$x=-0110 \qquad [x]_{移}=2^{4-1}+(-0110)=0010$

当 $x=0$ 时，$[x]_{移}=1000$，表示是唯一的。

在移码表示法中，正数的符号位为 1，负数的符号位为 0，与前边介绍的原码、反码和补码对符号位的表示正相反。通过上边的求法可以看出，实际上移码与补码只相差一个符号位，其表示的数码个数与范围均与补码相同。

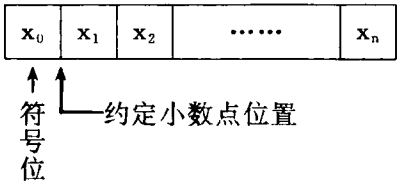
二、数的定点表示和浮点表示

在计算机中的数有两种表示方法：定点表示法和浮点表示法。

1. 数的定点表示法

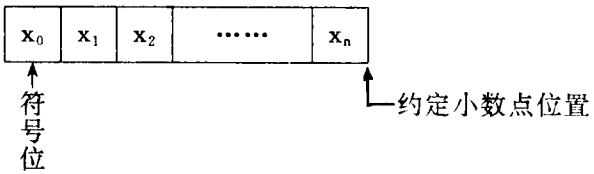
定点表示法规定，数据小数点的位置是固定不变的。在计算机中常采用两种定点形式：一是小数点的位置约定在数据最高位之前，表示的是定点小数；另一种是小数点的位置固定在最低位之后，表示的是定点整数。

定点小数在计算机中表示形式为：



小数点的位置是隐的，并非机器中有个小数点，小数点位置一经确定，机器中线路即相应确定。上述 $x_1 \sim x_n$ 是有效数值部分，参加运算的数都是小数。

定点整数在计算机中表示形式为：



以这种形式表示时参加运算的数都是整数。

通常参加运算的数既有整数部分，也有小数部分，为了将它们表示成约定的定点数形式，一般设一个比例因子。根据将原数扩大或缩小的倍数置比例因子，计算后再还原成实际数值。

无论是定点小数还是定点整数，参加运算的数和运算的结果都必须在定点数所能表示的范围内，否则就会产生溢出。

2. 数的浮点表示法

浮点表示就是小数点的位置是浮动的，不是固定的。对任意一个二进制数 N，可以写成：

$$N = \pm M \times 2^{\pm E}$$

其中 M 称为数 N 的尾数，E 为数 N 的指数，在浮点表示中称为阶码，即 2 的次幂，而基数 2 在浮点表示中并不出现。浮点数的表示形式如下：