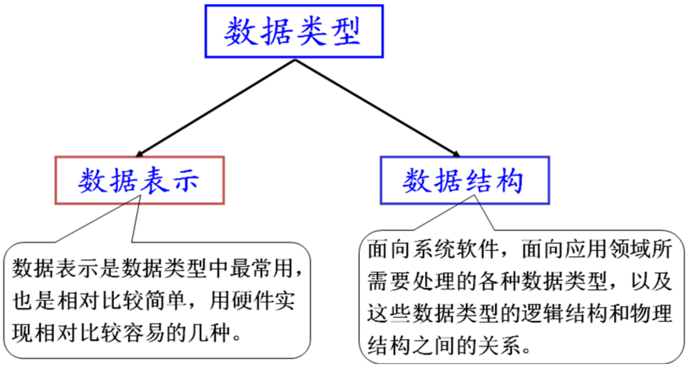
第 2 章 计算机的信息表示

(一)计算机中表示信息的数据类型



为了表示不同类型的信息，需要利用不同的二进制数码的方法和数制。

- 无符号数
- 符号数
- 二进制位串

(二) 无符号定点数表示

（1)进位计数制

进位计数制是指按照进位制的方法表示数。

不同的数制均涉及两个基本概念：基数和权。

- 基数：进位计数制中所拥有数字的个数.
- 权：每位数字的值等于数字乘以所在位数的相关常数，这个常数就是权。

任意一个 R 进制数 X，设整数部分为 n 位，小数部分为 m 位，则 X 可表示为：

$$X=a_{n-1}r^{n-1}+a_{n-2}r^{n-2}+\cdots+a_0r^0+a_{-1}r^{-1}+a_{-2}r^{-2}+\cdots+a_{-m}r^{-m}$$

$$(X)r=\sum_{i=n-1}^{-m}K_ir^i$$

（2)不同数制间的数据转换

①二、八、十六进制数转换成十进制数

利用上面讲到的公式：

- $(N)_2=\sum Di \cdot 2^i$
- $(N)_8=\sum Di \cdot 8^i$
- $(N)_{16}=\sum Di \cdot 16^i$

进行计算。

②十进制数转换成二进制数

通常要对一个数的整数部分和小数部分分别进行处理，各自得出结果后再合并。

- 对整数部分，一般采用除 2 取余数法
- 对小数部分，一般用乘 2 取整数法

(3)二进制数、八进制数和十六进制数之间的转换

八进制数和十六进制数是从二进制数演变而来的：

由 3 位二进制数组成 1 位八进制数；

由 4 位二进制数组成 1 位十六进制数。

对一个兼有整数和小数部分的数以小数点为界，小数点前后的数分别分组进行处理，不足的位数用 0 补足。

对整数部分将 0 补在数的左侧，对小数部分将 0 补在数的右侧，这样数值不会发生差错。

(三) 有符号定点数表示

（1） 真值和机器数

真值：数据的数值通常以正(+)负(-)号后跟绝对值来表示，称之为“真值”。

机器数：在计算机中正负号也需要数字化，一般用 0 表示正号，1 表示负号。把符号数字化的数成为机器数。

（2）根据符号位和数值位的编码方法不同，机器数分为原码、补码和反码

① 原码表示法

机器数的最高位为符号位，0 表示正数，1 表示负数，数值跟随其后，并以绝对值形式给出。这是与真值最接近的一种表示形式。

原码的定义：

$$[X]_{原}=\begin{cases}X; & 0\le X<1\\ 1-X=1+|X|; & -1<X\le 0\end{cases}$$

② 补码表示法

机器数的最高位为符号位，0 表示正数，1 表示负数，其定义如下：

$$[X]_{补} = \begin{cases} X; & 0 \le X < 1 \\ 2 + X = 2 - |X|; & -1 < X \le 0 \end{cases}$$

③反码表示法

机器数的最高位为符号，0 表示正数，1 表示负数。反码的定义：

$$[X]_{反} = \begin{cases} X; & 0 \le X < 1 \\ 2 - 2^{-n} + X = 2 - |X|; & -1 < X \le 0 \end{cases}$$

	原码	补码	反码
整数	$[x]_{原} = \begin{cases} 0,x & 2^n > x \ge 0 \\ 2^n - x & 0 \ge x > -2^n \end{cases}$	$[x]_{补} = \begin{cases} 0,x & 2^n > x \ge 0 \\ 2^{n+1} + x & 0 \ge x > -2^n \end{cases}$ (mod 2^{n+1})	$[x]_{反} \begin{cases} 0,x & 2^n > x \ge 0 \\ (2^{n+1} - 1) + x & 0 \ge x > -2^n \end{cases}$ (mod($2^{n+1} - 1$))
小数	$[x]_{原} \begin{cases} x & 1 > x \ge 0 \\ 1 - x & 0 > x \ge -1 \end{cases}$	$[x]_{补} = \begin{cases} x & 1 > x \ge 0 \\ 2 + x & 0 > x \ge -1 \end{cases}$ (mod 2)	$[x]_{反} = \begin{cases} x & 1 > x \ge 0 \\ (2 - 2^{-n}) + x & 0 > x \ge -1 \end{cases}$ (mod(2-2^{-n}))
0	$[+0]_{原}$=0.0000≠ $[-0]_{原}$=1.0000	$[+0]_{补} = [-0]_{补}$=0.0000	$[+0]_{反}$=0.0000≠ $[-0]_{反}$=1.1111
		负数原码求反+1	负数每位求反

移码 $[x]_{移} = 2^n + x \quad (2^n > x \ge -2^n)$ 移码表示中零也是唯一的

真值的移码和补码仅差一个符号位。若将补码的符号位由 0 改为 1 或从 1 改为 0 即可得到真值的移码

乘法运算可用移码和加法来实现，两个 n 位数相乘，总共要进行 n 次加法运算和 n 次移位运算

三种机器数的特点可以归纳为：

- 三种机器数的最高位均为符号位。符号位和数值位之间可用“.”(对于小数)或“，”(对于整数)隔开
- 当真值为正时，原码，补码和反码的表示形式均相同，即符号位用“0”表示，数值部分与真值部分相同
- 当真值为负时，原码，补码和反码的表示形式不同，其它符号位都用“1”表示，而数值部分有这样的关系，即补码是原码的“求反加 1”，反码是原码的“每位求反”。

(四) 定点数的运算

（1）定点数的位移运算

- 左移，绝对值扩大；右移，绝对值缩小。

- 算术移位和逻辑移位的区别：
  - 算术移位：带符号数移位；
  - 逻辑移位：无符号数移位。

（2）原码定点数的加/减运算;

对原码表示的两个操作数进行加减运算时，计算机的实际操作是加还是减，不仅取决指令中的操作码，还取决于两个操作数的符号。而且运算结果的符号判断也较复杂。

例如，加法指令指示做(＋A)＋(－B)由于一操作数为负，实际操作是做减法(＋A)-(+B)，结果符号与绝对值大的符号相同。同理，在减法指令中指示做(＋A)－(－B)实际操作做加法(＋A)＋(＋B)，结果与被减数符号相同。由于原码加减法比较繁琐，相应地需要由复杂的硬件逻辑才能实现，因此在计算机中很少被采用。

（3）补码定点数的加/减运算;

① 加法

整数 [A]补 + [B]补= [A+B]补(mod 2n+1)

小数 [A]补 + [B]补= [A+B]补(mod 2)

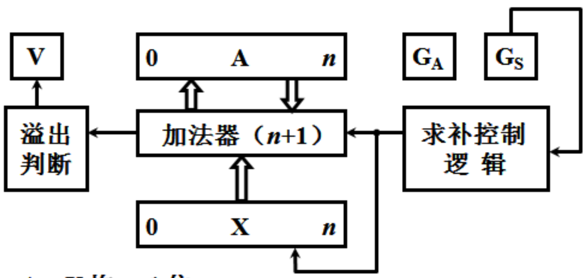
② 减法

整数 [A]补 - [B]补= [A+(-B)]补=[A]补 + [-B]补(mod 2n+1)

小数 [A]补 - [B]补= [A+(-B)]补=[A]补 + [-B]补(mod 2)

无需符号判定，连同符号位一起相加，符号位产生的进位自然丢掉

③补码加减法的硬件配置



A、X 均 n+1 位
用减法标记 G_S 控制求补逻辑

（4）溢出概念和判别方法

当运算结果超出机器数所能表示的范围时，称为溢出。显然，两个异号数相加

或两个同号数相减，其结果是不会溢出的。仅当两个同号数相加或者两个异号数相减时，才有可能发溢出的情况，一旦溢出，运算结果就不正确了，因此必须将溢出的情况检查出来。判别方法有三种：

① 当符号相同的两数相加时，如果结果的符号与加数(或被加数)不相同，则为溢出。

② 当任意符号两数相加时，如果 C=Cf，运算结果正确，其中 C 为数值最高位的进位，Cf 为符号位的进位。如果 C≠Cf ，则为溢出，所以溢出条件=$C \oplus Cf$ 。

③ 采用双符号 fs2fs1。正数的双符号位为 00，负数的双符号位为 11。符号位参与运算，当结果的两个符号位甲和乙不相同时，为溢出。所以溢出条件= $fs2 \oplus fs1$ ，或者溢出条件= fs2fs1 + fs2fs1

(五) 其他编码

（1）BCD 码(Binary Coded Decimal 以二进制编码的十进制码)

在计算机中采用 4 位二进制码对每个十进制数位进行编码。4 位二进制码有 16 种不同的组合，从中选出 10 种来表示十进制数位的 0～9，用 0000，0001，…，1001 分别表示 0，1，…，9，每个数位内部满足二进制规则，而数位之间满足十进制规则，故称这种编码为“以二进制编码的十进制(binary coded decima1，简称 BCD)码”。

在计算机内部实现 BCD 码算术运算，要对运算结果进行修正，对加法运算的修正规则是：

- 如果两个一位 BCD 码相加之和小于或等于$(1001)_2$，即$(9)_{10}$，不需要修正；
- 如相加之和大于或等于$(1010)_2$，或者产生进位，要进行加 6 修正，如果有进位，要向高位进位。

（2）字符与字符串

在计算机中要对字符进行识别和处理，必须通过编码的方法，按照一定的规则将字符用一组二进制数编码表示。字符的编码方式有多种，常见的编码有 ASCII 码，EBCDIC 码等。

① ASCII 码(American Standard Code for Information Interchange 美国信息交换标准码)

ASCII 码用 7 位二进制表示一个字符，总共 128 个字符元素，包括 10 个十进制

数字(0-9)，52 个英文字母(A-Z 和 a-z)，34 专用符号和 32 控制符号。

② EBCDIC 码为 Extended Binary Coded Decimal Interchange Code 的简称，它采用 8 位来表示一个字符。

③ 字符串的存放

向量存储法：字符串存储时，字符串中的所有元素在物理上是邻接的。

串表存储法：字符串的每个字符代码后面设置一个链接字，用于指出下一个字符的存储单元的地址。