



第2章 数值表示及转换

何宾

2015.02

本章主要内容

- 常用码制
- 正数表示方法
- 正数码制转换
- 负数表示方法
- 负数补码的计算
- 定点数表示
- 浮点数表示

常用码制

--二进制码制

二进制是以2为基数的进位制，即：逢2进1。

- **在C/C++语言中，二进制数以0b开头**

- **比如：0b1011，0b010111**

- **在汇编语言中，二进制数以B/b结尾**

- **比如：1011B/1011b、010111B/01011b**

常用码制

--十进制码制

十进制是以10为基数的进位制，即：逢10进1。

- **在计算机系统中，对十进制数的表示没有特殊的要求。**

常用码制

--八进制码制

八进制是以8为基数的进位制，即：逢8进1。

- **在C/C++语言中，八进制数以0开头**

- **比如：0123，0675**

- **在汇编语言中，八进制数以O/o结尾**

- **比如：123O/123o、675O/675o**

常用码制

--十六进制码制

十六进制是以16为基数的进位制，即：逢16进1。

- 在16进制计数规则中，只使用数字0、1、2、3、4、5、6、7、8、9和字母A/a、B/b、C/c、D/d、E/e、F/f表示。
- 在C/C++语言中，十六进制数以0x开头
 - 比如：0x1234，0xE1DD
- 在汇编语言中，十六进制数以H结尾
 - 比如：1234H、E1DDH

常用码制

--不同进制数之间的对应关系

十进制数	二进制数	八进制数	十六进制数
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6

如需原始PPT文件请点击此处

<http://www.gpnewtech.com/ppt>

常用码制

--不同进制数之间的对应关系

7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D

常用码制

--不同进制数之间的对应关系

14	1110	16	E
15	1111	17	F
16	1,0000	20	10
17	1,0001	21	11
18	1,0010	22	12
19	1,0011	23	13
20	1,0100	24	14

常用码制

--BCD码

BCD码（ Binary-Coded Decimal ）亦称二进制十进数或二-十进制代码。用4位二进制数来表示1位十进制数中的0~9这10个数码。

■ BCD码可分为有权码和无权码两类：

□ 有权BCD码有8421码、2421码、5421码，其中8421码是最常用的；

□ 无权BCD码有余3码、格雷码（注：格雷码并不是BCD码）等。

正数表示方法

--正整数的表示

- 对于一个四位十进制数7531，用10的幂次方表示为：

$$7 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 1 \times 10^0$$

- 对于一个五位二进制数10101，用2的幂次方表示为：

$$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

- 对于一个三位八进制数327，用8的幂次方表示为：

$$3 \times 8^2 + 2 \times 8^1 + 7 \times 8^0$$

- 对于一个四位十六进制数13AF，用16的幂次方表示为：

$$1 \times 16^3 + 3 \times 16^2 + A \times 16^1 + F \times 16^0$$

正数表示方法

--正整数的表示

推广总结：

- 对于一个N位的2进制数，最低位为第0位，最高位为第N-1位。

其计算公式为：

$$Y = S_{N-1} \cdot 2^{N-1} + S_{N-2} \cdot 2^{N-2} + \dots + S_1 \cdot 2^1 + S_0 \cdot 2^0$$

- S_i 为第i位二进制数的值，取值为0或者1。
- 2^i 为第i位二进制数的权值。
- Y为等效的十进制数。

正数表示方法

--正整数的表示

- 对于一个N位的8进制数，最低位为第0位，最高位为第N-1位。

$$Y = S_{N-1} \cdot 8^{N-1} + S_{N-2} \cdot 8^{N-2} + \dots + S_1 \cdot 8^1 + S_0 \cdot 8^0$$

- S_i 为第i位八进制数的值，取值范围为0~7。
- 8^i 为第i位八进制数的权值。
- Y为等效的十进制数。

正数表示方法

--正整数的表示

- 对于一个N位的16进制数，最低位为第0位，最高位为第N-1位。

$$Y = S_{N-1} \cdot 16^{N-1} + S_{N-2} \cdot 16^{N-2} + \dots + S_1 \cdot 16^1 + S_0 \cdot 16^0$$

- S_i 为第i位十六进制数的值，取值范围为0~9，A~F。
- 16^i 为第i位十六进制数的权值。
- Y为等效的十进制数。

正数表示方法

--正小数的表示

- 对于一个三位十进制小数0.714，用10的幂表示为：

$$7 \times 10^{-1} + 1 \times 10^{-2} + 4 \times 10^{-3}$$

- 对于一个五位二进制小数0.10101，用2的幂表示为：

$$1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5}$$

正数表示方法

--正小数的表示

推广总结：

- 对于一个N位的2进制小数，最高位为第0位，最低位为第N-1位。其计算公式为：

$$Y = S_0 \cdot 2^{-1} + S_1 \cdot 2^{-2} + \dots + S_{N-2} \cdot 2^{-(N-1)} + S_{N-1} \cdot 2^{-N}$$

- S_i 为第*i*位二进制小数的值，取值范围为0或者1。
- $2^{-(i+1)}$ 为第*i*位二进制小数的权值。
- Y为等效的十进制小数。

十进制整数转换成其它进制数

--十进制整数转换成二进制数

长除法

- 采用除法，除数始终为2，将十进制进行分解，直到商为0结束。然后，按顺序将最后得到的余数排在最高位，而最先得到的余数排在最低位。

比较法

- 让需要转换的正整数和不同的二进制权值进行比较。当：
 - 需要转换的正整数大于所对应的二进制权值时，得到1；并且转换的正整数减去二进制权值得到余数。然后，再用得到的余数与下一个二进制权值进行比较。
 - 需要转换的数小于所对应的二进制权值时，得到0。并且不做任何处理。

十进制整数转换成其它进制数

--十进制整数转换成二进制数

【例】使用长除法将十进制整数59转成所对应的二进制数

$$59 \div 2 = 29 \dots 1$$

$$29 \div 2 = 14 \dots 1$$

$$14 \div 2 = 7 \dots 0$$

$$7 \div 2 = 3 \dots 1$$

$$3 \div 2 = 1 \dots 1$$

$$1 \div 2 = 0 \dots 1$$

注：...前面的数字表示商，...后面表示的数字为余数。

所以，十进制正整数59所对应二进制数111011。

十进制整数转换成其它进制数

--十进制整数转换成二进制数

【例】使用比较法将十进制整数59转换所对应的二进制数

通过比较法，如下表所示，得到十进制正整数59所对应的二进制数为111011。

比较的数	59	59	27	11	3	3	1
二进制权值	2^6 (64)	2^5 (32)	2^4 (16)	2^3 (8)	2^2 (4)	2^1 (2)	2^0 (1)
对应的二进制值	0	1	1	1	0	1	1
余数	59	27	11	3	3	1	0

十进制整数转换成其它进制数

--十进制整数转换成十六进制数

长除法

- 采用除法，除数始终为16，将十进制进行分解，直到商为0结束。然后，按顺序将最后得到的余数排在最高位，而最先得到的余数排在最低位。

比较法

- 让需要转换的正整数和不同的十六进制权值进行比较，当：
 - 需要转换的正整数大于所对应的十六进制权值时，得到商；并且转换的正整数减去十六进制权值与商乘积后得到余数。然后，再用得到的余数与下一个十六进制权值进行比较。
 - 需要转换的数小于所对应的十六进制权值时，得到0。并且不做任何处理。

十进制整数转换成其它进制数

--十进制整数转换成十六进制数

【例】使用长除法将十进制整数4877转换成所对应的十六进制数

$$4877 \div 16 = 304 \dots 13(D)$$

$$304 \div 16 = 19 \dots 0$$

$$19 \div 16 = 1 \dots 3$$

$$1 \div 16 = 0 \dots 1$$

注：...前面的数字表示商，...后面表示的数字为余数。

所以，十进制正整数4877所对应十六进制数为130D。

十进制整数转换成其它进制数

--十进制整数转换成十六进制数

【例】使用比较法将十进制整数4877转换成所对应的十六进制数

通过比较法，如下表所示，得到十进制正整数4877所对应的十六进制数为130D。

比较的数	4877	781	13	13
十六进制权值	16^3 (4096)	16^2 (256)	16^1 (16)	16^0 (1)
对应的十六进制值	1	3	0	D
余数	781	13	13	0

正数码制转换

--十进制小数转换成二进制数

长乘法

- 将小数乘以2，取其整数部分的结果。然后，再用计算后的小数部分依此重复计算，算到小数部分全为0为止。在读取整数部分的结果时，最先得到的整数放在小数的最高有效位，而最后得到的整数放在小数的最低有效位。

比较法

- 让需要转换的数和不同的二进制权值进行比较，当：
 - 需要转换的小数大于所对应的二进制权值时，得到1；并且转换的小数减去二进制权值得到余数。然后，再用得到的余数与下一个二进制权值进行比较。
 - 需要转换的数小于所对应的二进制权值时，得到0。并且不做任何处理。

正数码制转换

--十进制小数转换成二进制数

【例】使用长乘法将一个十进制小数0.8125转换成所对应的二进制小数

$$0.8125 \times 2 = 1.625 \quad \text{取整是1}$$

$$0.625 \times 2 = 1.25 \quad \text{取整是1}$$

$$0.25 \times 2 = 0.5 \quad \text{取整是0}$$

$$0.5 \times 2 = 1.0 \quad \text{取整是1}$$

即：十进制小数0.8125所对应的二进制小数表示为0.1101。

正数码制转换

--十进制小数转换成二进制数

【例】使用比较法将一个十进制小数0.8125转换成所对应的二进制小数

通过比较法，得到十进制小数0.8125所对应的二进制小数为0.1101。

比较的数	0.8125	0.3125	0.0625	0.0625
二进制权值	2^{-1} (0.5)	2^{-2} (0.25)	2^{-3} (0.125)	2^{-4} (0.0625)
对应的二进制值	1	1	0	1
余数	0.3125	0.0625	0.0625	0

负数表示方法

一个N位的系统总共可以表示 2^N 个数，因此一个有用的编码就是使用一半可用的编码（ $2^N/2$ ）表示正数，另一半表示负数。

■ 可以将一个比特位设计成一个符号位，用于区分正数和负数。

在这种表示方法中，最高有效位（Most Significant Bit，MSB）可以作为符号位。如果：

- 符号位为1，所表示的数为负数。

- 符号位为0，所表示的数为正数。

■ 在所有可能的负数编码方案中，经常使用两种：

- 符号幅度

- 二进制补码

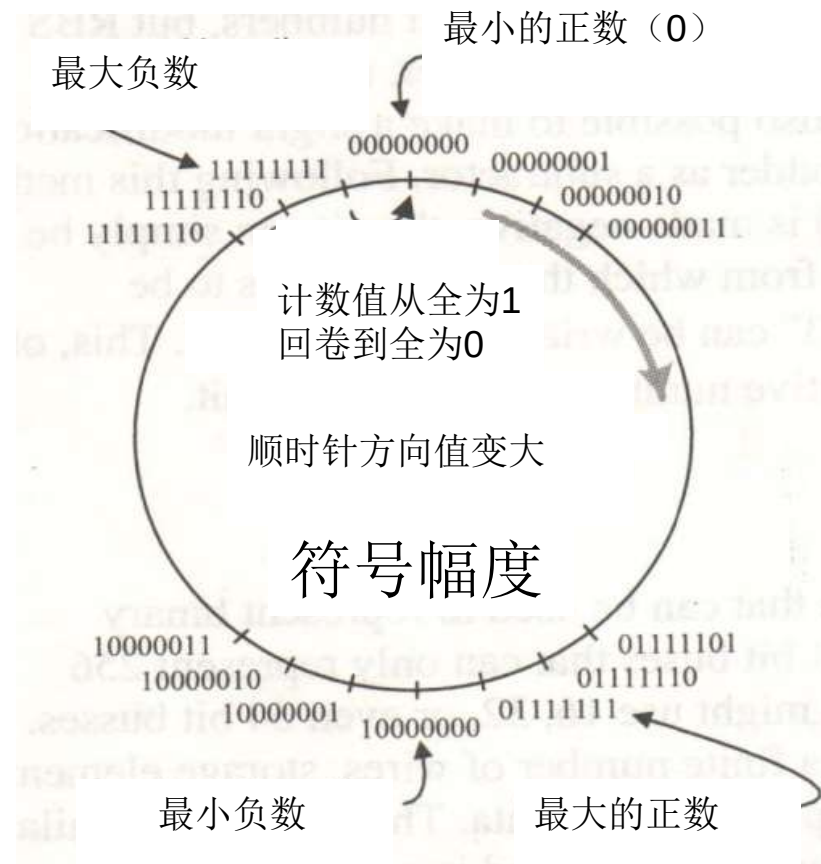
负数表示方法

--符号幅度表示法

■ 就是用MSB表示符号位，剩下的位表示幅度，如右图所示。在一个8位的符号幅度系统中：

□ 十进制数16表示为00010000

□ 而十进制数-16表示为10010000



符号幅度表示法

负数表示方法

--符号幅度表示法

用于数字系统中最不利的方面表现在：

- 如果0到 $2N$ 的计数范围从最小到最大，则最大的正数将出现在所表示范围的一半的地方，然后，跟随最小的负数。
- 最大的负数出现在范围的末尾，更多的计数将出现回卷。这是由于不能表示 $2N+1$ 。
- 因此，在计数范围内， $2N$ 后面跟着0，这样最大的负数就立即调整到最小的正数。

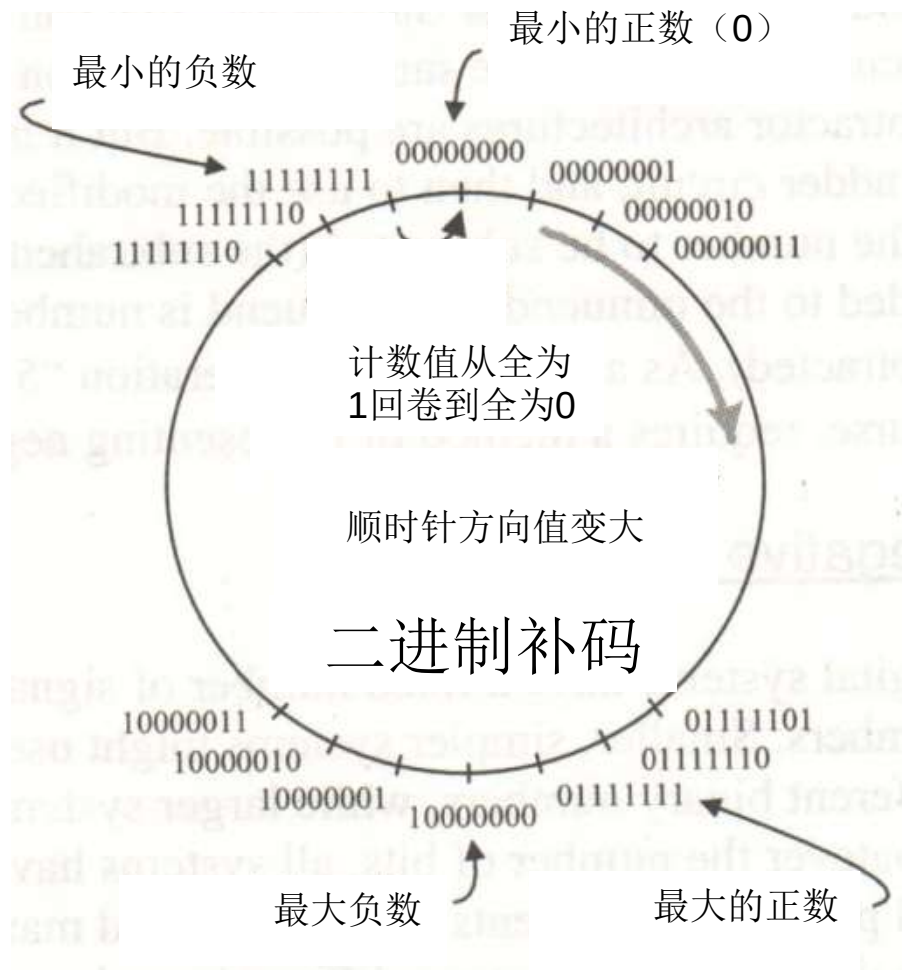
负数表示方法

--补码表示法



补码表示法

- 由于上面的原因，引入了二进制补码的概念，如右图所示。



负数表示方法

--补码表示法



- 在二进制补码编码中，MSB仍是符号位，1表示负数，0表示正数。
- 在二进制补码中，0由一个包含所有0的比特模式所定义。其余的 2^N-1 个数表示非零的正数和负数。
 - 由于 2^N-1 是奇数， $(2^N-1)/2$ 个编码表示负数， $[(2^N-1)/2]-1$ 个编码表示正数。
 - 换句话说，可以表示的负数比正数要多一个。
 - 最大负数的幅度要比最大正数的幅度个数要多一个。

负数表示方法

--补码表示法

二进制补码编码的不利的地方是，不容易理解负数。

- 对于一个N位字长的二进制补码来说，其可以表示的有符号数（包括正数、负数和0）的范围是：

$$- 2^{N-1} \sim 2^{N-1} - 1$$

负数补码的计算

--负整数补码的计算

■ 原码转补码

- 将该负数所对应的正数按位全部取反。
- 将取反后的结果加1。

【例】将+17转换-17的二进制补码

- 对应的17的原码为00010001
- 按位取反后变成11101110
- 结果加1，变成11101111

负数补码的计算

--负整数补码的计算

【例】将-35转换为+35的二进制补码

- 对应的-35的补码为11011101
- 按位取反后变成00100010
- 结果加1，变成00100011

【例】将-127转换为+127的二进制补码

- 对应的-127的补码为10000001
- 按位取反后变成01111110
- 结果加1，变成01111111

负数补码的计算

--负整数补码的计算

■ 比较法

- 得到需要转换负数的最小权值，该权值为负数，以 -2^i 表示，使得其满足：

$$-2^i < \text{需要转换的负数} ;$$

- 取比该权值绝对值 2^i 小的权值，以 2^{i-1} ， 2^{i-2} ，...， 2^0 的幂次方表示；
- 需要转换的负数 $+2^i$ ，得到了正数，以后的权值 2^{i-1} ， 2^{i-2} ，...， 2^0 按照前面的方法和这个正数进行比较。

负数补码的计算

--负整数补码的计算

【例】使用比较法得到负整数-97所对应的二进制补码

对于负的十进制整数-97来说，假设使用8位二进制数进行表示，则其所对应的二进制补码为10011111B，如下表所示。

转换的数	-97	31	31	31	15	7	3	1
权值	-2^7 (-128)	2^6 (64)	2^5 (32)	2^4 (16)	2^3 (8)	2^2 (4)	2^1 (2)	2^0 (1)
二进制数	1	0	0	1	1	1	1	1
余数	31	31	31	15	7	3	1	0

负数补码的计算

--负小数补码的计算

比较法

- 得到需要转换负数的最小权值，该权值为负数，以 -2^0 表示
- 取比该权值绝对值 2^i 小的权值，以 2^{-1} ， 2^{-2} ，...， 2^{-N} 的幂次方表示。
- 需要转换的负数+1，得到了正数，以后的权值 2^{-1} ， 2^{-2} ，...， 2^{-N} 按照前面的方法和这个正数进行比较。

负数补码的计算

--负小数补码的计算

【例】使用比较法得到负小数-0.03125所对应的二进制补码。
通过比较法，得到十进制负小数-0.03125所对应的二进制小数为1.11111。

转换的数	-0.03125	0.96875	0.46875	0.21875	0.09375	0.03125
权值	-2^0 (-1)	2^{-1} (0.5)	2^{-2} (0.25)	2^{-3} (0.125)	2^{-4} (0.0625)	2^{-5} (0.03125)
二进制数	1	1	1	1	1	1
余数	0.96875	0.46875	0.21875	0.09375	0.03125	0

定点数表示

定点数就是二进制小数点在固定位置的数。二进制小数点左边部分的位被定义为整数位，而该点右边部分的位被定义成小数位。通常表示为 $Q_{m.n}$ 格式，即：

- m 为整数部分的二进制的位数。

- m 越大，表示数的动态范围越大；
- m 越小，表示数的范围越小；

- n 为小数部分的二进制的位数。

- n 越大，表示数的精度越高；
- n 越小，表示数的精度就越低；

定点数表示

【例】将十进制数-28.65625用定点二进制的形式表示。
使用前面所介绍的比较法，将-28.65625表示成Q7.5定点二进制数1100010.01011B。

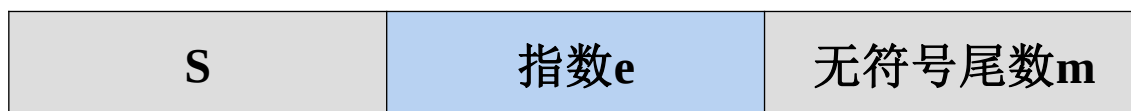
转换的数	-28.65625	35.34375	3.34375	3.34375	3.34375	3.34375
权值	-2^6	2^5	2^4	2^3	2^2	2^1
二进制数	1	1	0	0	0	1
余数	35.34375	3.34375	3.34375	3.34375	3.34375	1.34375

转换的数	1.34375	0.34375	0.34375	0.09375	0.09375	0.03125
权值	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
二进制数	0	0	1	0	1	1
余数	0.34375	0.34375	0.09375	0.09375	0.03125	0

浮点数表示

大多数的浮点数都遵循单精度或双精度的IEEE浮点标准。

- 标准浮点数字长由一个符号位S，指数e和无符号（小数）的规格化尾数m构成，如下所示。



- 浮点数可以用下式描述：

$$X = (-1)^S 1.m \cdot 2^{e-bias}$$

- 的表示能力，取阶码 P = -126，尾数 M = (0.00000000000000000000000001)₂。此时该数的二进制表示为：0 00000000 000000000000000000000001。**

浮点数表示

IEEE的单精度和双精度格式的参数

	单精度	双精度
字长	32	64
尾数	23	52
指数	8	11
偏置	127	1023