二、存储数字

<1>整数（给定一个整数，存储方式决定了对其进行操作时的表现形式，如用补码方式，则对整数用补码存为二进制，对二进制用补码还原为整数）

* 定点表示法：小数点位置固定，规定了小数点前用多少位存，小数点后用多少位存

1. 存储无符号整数

计算机会分配一个存储无符号整数的最大二进制位数n，超过2的n次方-1的无符号整数存储时会溢出

溢出：n个二进制位表示最大数为2的n次方-1，若某个数超过最大值，那么其二进制会舍弃掉超过n位的部分，即该数减去k个2的n次方，直到最后落在0~2的n次方-1的范围内

2. 符号加绝对值存储法

对分配n位二进制的有符号整数，计算机将其最左端解释为符号位，则其正数范围为0~2的n-1次方-1；负数范围为-（2的n-1次方-1），但形如1000和0000造成0的双标

溢出：

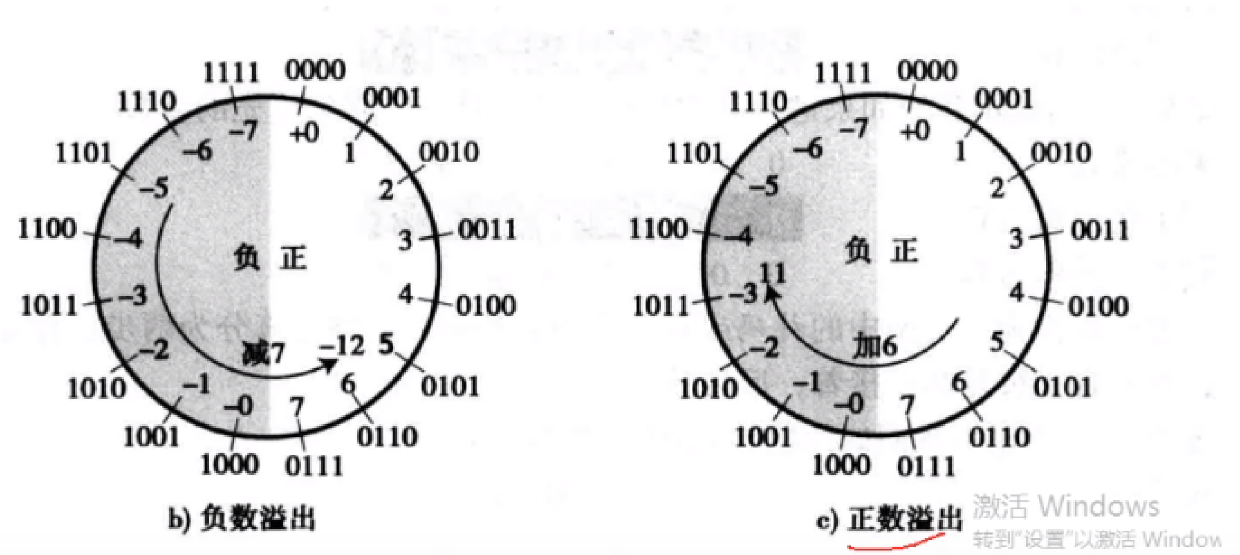


图 3-7 符号加绝对值表示法的溢出

3. 补码存储法（任意二进制数计算机将其看成补码）

（1）以二进制补码形式存储整数

1.将整数转为二进制形式，如+7转为0111，-7转为1111（最左边为符号位）

2.若原整数是正数，则其补码为上述二进制，若原整数是负数，补码为上述二进制除符号位外每位取反后再加一

（2）将用二进制补码表示的数转换为整数

1.若该二进制数最左端为0，则直接还原整数

2.若最左端为1，表示原数为负数，将其二进制除符号位外每位取反，最后+1；

3.注意特殊情况，若用二进制补码表示法存储在n位存储单元中的数位1000..00，则按上述方法还原为整数时符号位会被进一，此时该数变为“2”00000...000，此时该数其实是-0,

即-(2的n次方)

（3）补码表示法的溢出

对n位二进制的补码表示法，表示数的范围是-(2的n-1次方)~2的n-1次方-1，因为其中-0表示为2的n-1次方

在补码表示法中，运算的操作数和结果计算机都看作补码，减法转化为加上负数

4.整数三种表示法的比较

（续）

存储单元的内容	无 符 号	符号加绝对值	二进制补码	存储单元的内容	无 符 号	符号加绝对值	二进制补码
1000	8	−0	−8	1100	12	−4	−4
1001	9	−1	−7	1101	13	−5	−3
1010	10	−2	−6	1110	14	−6	−2
1011	11	−3	−5	1111	15	−7	−1

<2>实数

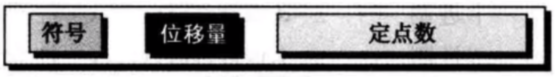
*定点表示法规定了小数点前后的数码位数，对于整数部分很大或者小数部分很小的实数来说，超过规定位数的部分会被舍弃，造成实数的精度损失

*实数通常用浮点表示法

1. 浮点表示法

用于维持正确度或精度的解决方法是使用**浮点表示法**。该表示法允许小数点浮动：我们可以在小数点的左右有不同数量的数码。使用这种方法极大地增加了可存储的实数范围：带有很大的整数部分或很小的小数部分的实数可以存储在内存中了。在浮点表示法中，无论十进制还是二进制，一个数字都由 3 部分组成，如图 3-10 所示。

第一部分是符号，可正可负。第二部分显示小数点应该左右移动构成实际数字的位移量。第三部分是小数点位置固定的定点表示法。



浮点表示法

图 3-10 在浮点表示法中的实数的 3 个部分

一个数字的浮点表示法由 3 部分组成：符号、位移量和定点数。

类似方法可用于表示很小或很大的存储于计算机中的二进制数字（整数和实数皆可）。

例 3.20 用浮点格式表示数字 $(1010010000000000000000000000000000.00)_2$。

解 使用前例同样的方法，小数点前只保留一位数字，如下所示：

实际数字 → + $(1010010000000000000000000000000000.00)_2$

科学计数法 → + 1.01001×2^{32}

例 3.21 用浮点格式表示数字 $-(0.00000000000000000000000101)_2$。

解 使用前例同样的方法，小数点左边只留一个非零数码：

实际数字 → − $(0.00000000000000000000000101)_2$

科学计数法 → − 1.01×2^{-24}

e.g计算机中存储二进制实数10011.1000

（1）规范化：保证小数点左边只有一位非零数码，在二进制表示下该数码为1

原数变为1.00111000乘以10的4次方

（2）规范化确定后，存储一个实数只需要知道其三部分信息：符号，指数，尾数，在上述例子中，符号位为1，指数为4，尾数为小数点后面的部分

（3）符号位用0或1存储，尾数位用无符号整数存储，指数位用余码系统存储

余码系统：对于用n位存储单元存储的数，该数存储时加上2的n-1次方-1，保证其为正数，再用无符号表示法存储这个正数

5. IEEE 标准

电气和电子工程师协会（IEEE）已定义了几种存储浮点数的标准。这里我们讨论其中两种最常用的——单精度和双精度。该格式如图 3-12 所示。方框上方的数就是每一项的位数。

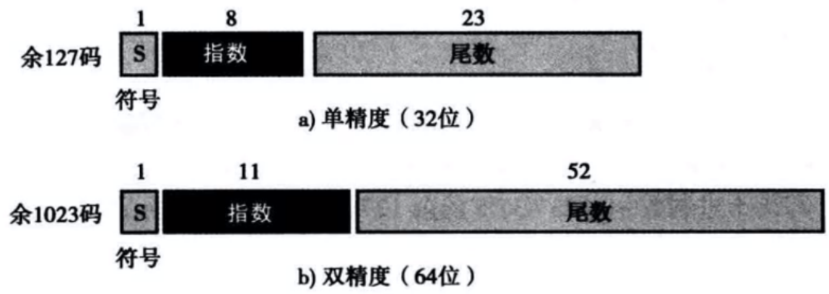


图 3-12　浮点数表示法的 IEEE 标准

其中余127码指的是2的（8-1）次方再减一

将实数存储为标准浮点格式

参照图 3-12，使用以下步骤，一个实数可以存储为 IEEE 标准浮点数格式：

1）在 S 中存储符号（0 或 1）。

2）将数字转换为二进制。

3）规范化。

4）找到 E 和 M 的值。

5）连接 S、E 和 M。

e.g

例 3.24　写出十进制数 –161.875 的余 127 码（单精度）表示法。

解

1）符号为负，所以 S=1。

2）十进制转换为二进制：161.875=$(10100001.111)_2$。

3）规范化：$(10100001.111)_2 = (1.0100001111)_2 \times 2^7$。

4）E=7+127=134= $(10000110)_2$，而 M= $(0100001111)_2$。

5）该表示法如下所示：

S	E	M
1	10000110	01000011110000000000000

存储在计算机中的数字是 11000011010000111100000000000000。

*其中余127码指的是上述8位指数，23位尾数的存储方式

*注意M不足23位时是向右补0，与传统无符号整数补0方式不同

将标准浮点格式还原为实数

一个以 IEEE 浮点格式之一存储的数字可以用以下步骤方法还原：

1）找到 S、E 和 M 的值。

2）如果 S=0，将符号设为正号，否则设为负号。

3）找到位移量（E–127）。

4）对尾数去规范化。

5）将去规范化的数字变为二进制以求出绝对值。

6）加上符号。

e.g

例 3.26　位模式 $(11001010000000000111000100001111)_2$ 以余 127 码格式存储于内存中，求该数字十进制计数法的值。

解

1）首位表示 S，后 8 位是 E，剩下 23 位是 M。

S	E	M
1	10010100	00000000111000100001111

2）符号为负号。

3）位移量 =E–127=148–127=21。

4）将 $(1.00000000111000100001111)_2 \times 2^{21}$ 去规范化。

5）二进制数是 $(1000000001110001000011.11)_2$。

6）绝对值是 2 104 378.75。

7）该数字是 –2 104 378.75。