进制及数值数据的编码

- What?Why?How?

目录

- 进制
- 数值数据的编码和表示

进制

- 十进制
  - 基数是10，有10个不同的数学符号，即0-9
- 二进制
  - 基数是2，有2个不同的数学符号，即0和1
- 八进制
  - 基数是8，有8个不同的数学符号，即0-7
- 十六进制
  - 基数是16，有16个不同的数学符号，即0-9,A,B,C,D,E,F

不同进制的数的大小计算

- 某一进制数的大小由系数项和权的乘积决定
- $(123.45)_{10}=1*10^{2}+2*10^{1}+3*10^{0}+4*10^{-1}+5*10^{-2}$
- $(10101.11)_{2}=1*2^{4}+1*2^{2}+1*2^{0}+1*2^{-1}+1*2^{-2}$
- $(375.4)_{8}=3*8^{2}+7*8^{1}+5*8^{0}+4*8^{-1}$

计算机中数值数据的编码和表示

- 机器数和真值
  - 实际运算中，数是有正负的，计算机中数也有正负，通过用一个数的最高位表示符号，如果字长为8位，分别为D7~~D0，那么D7为符号位，0表示正数，1表示负数，D6~D0为数值位
  - 例如：
    - 11010111 = -87
- 这样，在计算机中，连同符号一起数码化的数，就被称为机器数，如上面的11010111；而使用正负号加其绝对值的表示方法，称为该数的真值，如-87

原码

- 原码是指将最高位作为符号位（0表示正，1表示负），其它数字位代表数值本身的绝对值的数字表示方式
  - 数字6在计算机中原码表示为：0000 0110
  - 数字-6在计算机中原码表示为：1000 0110
- 以上是在8位计算机中的原码表示，如果在32位或16位计算机中，表示方法是一样的，只是多了几个0而已

原码

- 有了数值的表示方法就可以对数进行算数运算，但是很快就发现用带符号位的原码进行乘除运算时结果正确，而在加减运算的时候回出现问题，如下：
- $(1)_{10} - (1)_{10} = (1)_{10} + (-1)_{10} = (0)_{10}$
- $(00000001)_{原} + (10000001)_{原} = (10000010)_{原} = (-2)$
- 显然是不正确的

反码

- 反码表示规则为：如果是正数，则表示方法和原码一样，如果是负数，则保留符号位1，然后将这个数字的原码按照每位取反，则得到这个数的反码表示形式
  - 数字6在计算机中反码就是它的原码：0000 0110
  - 数字-6在计算机中反码为：1111 1001

反码

- 因为在两个正数的加法运算中是没有问题的，于是就发现问题出现在带符号位的负数身上，对除符号位外的其余各位逐位取反就产生了反码，反码的取值空间和原码相同且一一对应，下面是反码的减法运算：
  - (1)10-(1)10=(1)10+(-1)10=(0)10
  - (00000001)反+(11111110)反=(11111111)反=（-0）有问题
  - (1)10-(2)10=(1)10+(-2)10=(-1)10
  - (00000001)反+(11111101)反=(11111110)反=(-1)正确

补码

- 问题出现在(+0)和(-0)上，在人们的计算概念中0是没有正负之分的
- 于是就引入了补码的概念，负数的补码就是对反码加一，而正数不变，正数的原码反码补码都是一样的，在补码中用(-128)代替了(-0),所以补码的表示范围为
  - (-128~0~127),共256个

补码

- 补码是计算机表示数据的一般方式，其规则为：如果是正数，则表示方法和原码一样，如果是负数，则将数字的反码加上1（相当于将原码数值按位取反然后在对地位加1）

补码

- 注意:(-128)没有相对应的原码和反码，(-128)=(10000000)的补码的加减运算如下:
- $(1)_{10}-(1)_{10}=(1)_{10}+(-1)_{10}=(0)_{10}$
- $(00000001)_{补}+(11111111)_{补}=(00000000)_{补}=(0)$ 正确
- $(1)_{10}-(2)_{10}=(1)_{10}+(-2)_{10}=(-1)_{10}$
- $(00000001)_{补}+(11111110)_{补}=(11111111)_{补}=(-1)$ 正确

补码

- 所以补码的设计目的是：
  - 使符号位能与有效值部分一起参加运算，从而简化运算规则
  - 使减法运算转换为加法运算，进一步简化计算机中运算器的线路设计
- 所有这些转换都是在计算机的最底层进行的，而在我们使用的汇编、C等其他高级语言中使用的都是原码