2进制与运算

- 计算机的最小组成单位的IC
  因其1个引脚只有2种直流电压状态
  所以计算机只能用2进制处理信息
  - 比特 bit
    - 1位2进制数
  - 字节Byte
    - 8位2进制数
  - 小贴士：处理器的位数即为IC的引脚数
    例如：64位微处理器就具有64个引脚
  - 小贴士：计算机不会区分2进制数的信息究竟是数字/文字/还是图片。至于如何处理一个2进制数，则是由编写好的程序决定的
  - 小贴士：电流通过引脚时数值为1
    没有电流通过引脚时数值为0
- 进制的相关概念
  - 基数
    - 几进制的基数就是几
      - 例如：2进制的基数是2
        8进制的基数是8
        10进制的基数是10
  - 位权
    - 基数的n-1次幂
      - 例如：8位2进制的第1位的位权是2的0次幂
        第8位的位权则是2的7次幂
  - 加（位）权
    - 将[N进制中各个数位的值 × 该位的位权]加总
      - 得到该数的真实10进制数值
- 2进制的移位
  - 左移
    - 乘法规则：无论是几位2进制，向左移动n位，就比移位前扩大2的n次幂倍
      ※ 但是当位溢出时，此规则失效
      - 例如：00100000向左移动2位，得到10000000
        其10进制分别为32和128
    - 左移的补位原则
      - 移动的位一律补0
        - 例如：左移4位，原值扩大2的4次方（16）倍
          00001111 → 11110000
          （15 × 16 = 240）
      - 溢出的位一律丢弃
        - 例如：左移4位，原值理应扩大2的4次方（16）倍
          00011110 → 11110000
          （30 × 16 = 256 + 128 + 64 + 32 = 480）
          如果是9位2进制，通过上述等式可以看出，左移依然满足乘法规则。但由于受到位数限制，移位后的值实际上是（128 + 64 + 32 = 224）
  - 右移的先决条件：
    用'正数'来表示'负数'
    - 符号位
      - 无论是几位的2进制数，其最高位也是符号位
        - 符号位为0时该2进制数只能表示正数
          - 00000001只能表示1
        - 符号位为1时该2进制数可以表示负数或正数
          - 10000001为正数时，表示129
            而其为负数时，表示-127
            （01111111 = 127）
    - 补数
      - 2进制数 + 他的补数 = 0
        - 一个2进制负数的补数的补数，是该2进制数本身
      - 补数的计算方式：将该2进制数所有数位先取反，再+1
        - 例如：00000001是2进制的1
          其补数为（11111110 + 1 = 11111111）
          故11111111表示负数时，其为-1
    - 2进制中的负数
      - 在上例中，11111111为正数时，其值为255
        而11111111为负数时，其补数为（00000000 + 1 = 00000001），故其值为-1
      - 小贴士：正数的补数是其负数，而负数的补数则是具体的负值
        - 例如：负数10100101，我们向知道其值是多少
          先找到他的补数是（01011010 + 1 = 01011011）
          补数值为64 + 16 + 8 + 2 + 1 = 91
          所以该负数的值为-91
  - 右移
    - 逻辑右移的补位原则
      - 移位后，空出的所有高位一律补0
      - 溢出的位一律丢弃
    - 算数右移的补位原则
      - ※ 只有算数右移才具备除法规则
        除法规则：无论是几位2进制，向右移动n位，就比移位前缩小2的n次幂分之1
        ※ 但是当位溢出时，此规则失效
      - 移位后，空出的所有高位根据原2进制数的正负决定
        - 正数的例子：正数全部补0
          右移3位
          10101000 → 00010101
          168 × 1/8 = 21
        - 负数的例子：负数全部补1
          右移3位
          10101000 → 11110101
          先计算补数求出负数值：
          01011000 = -88
          00001011 = -11
          -88 × 1/8 = -11
  - 小贴士：笔者认为，当通过移位进行算数运算（成倍扩大或者等比例缩小）的时候，一般不会涉及到位数溢出的问题。因为无论是左移还是右移，一旦位数溢出，算数规则便失效了
  - 符号扩充
    - 保持值不变的情况下
      将低位2进制数转化成高位2进制数
      例如：8位2进制数转16位/32位2进制数
      - 方法：只需用符号位的值填充高位即可
        - 正数的例子
          用0填充
          - 8位的2进制数127
            01111111
            - 16位的2进制数127
              0000000001111111
        - 负数的例子
          用1填充
          - 8位的2进制数-1
            11111111
            - 16位2进制数-1
              1111111111111111
- 基于2进制的4种逻辑运算
  - AND
    - 0 AND 0 = 0
    - 1 AND 0 = 0
    - 0 AND 1 = 0
    - 1 AND 1 = 1
  - OR
    - 0 OR 0 = 0
    - 1 OR 0 = 1
    - 0 OR 1 = 1
    - 1 OR 1 = 1
  - XOR
    - 0 XOR 0 = 0
    - 1 XOR 0 = 1
    - 0 XOR 1 = 1
    - 1 XOR 1 = 1
  - NOT
    - NOT 0 = 1
    - NOT 1 = 0