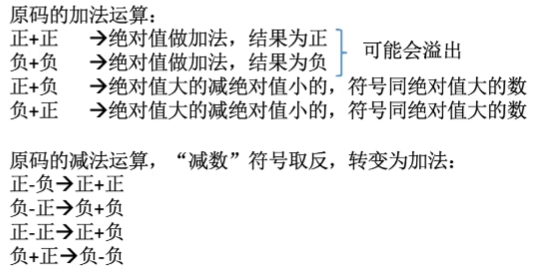
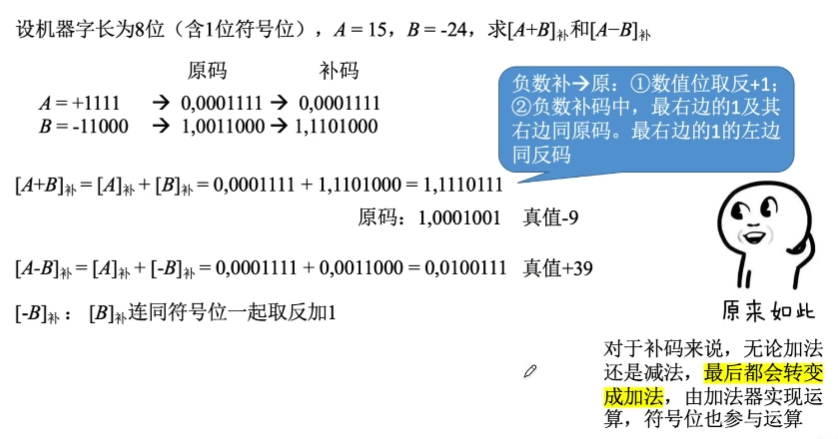
# 原码的加减运算



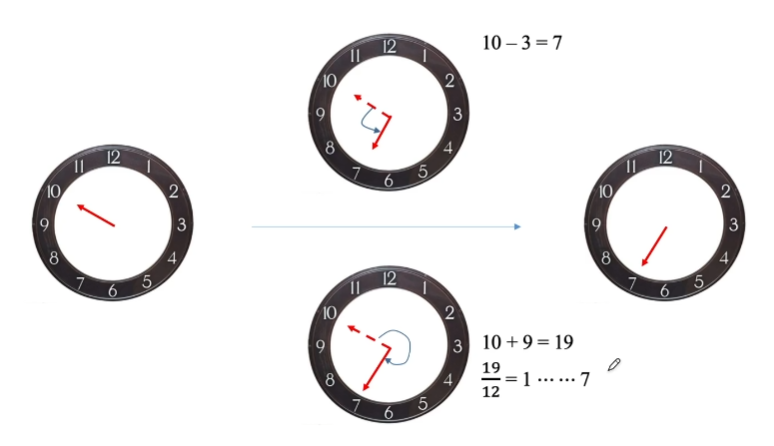
硬件实现加法容易，实现减法很难，通常使用补码的加法运算来等价替换原码的减法运算

# 补码的加法运算



# 补码的意义

## 用加法替代减法



在钟表这个范围内，7和19是等价的，因为19mod12=7

所以10-3这个减法可以用加法替代为10+9

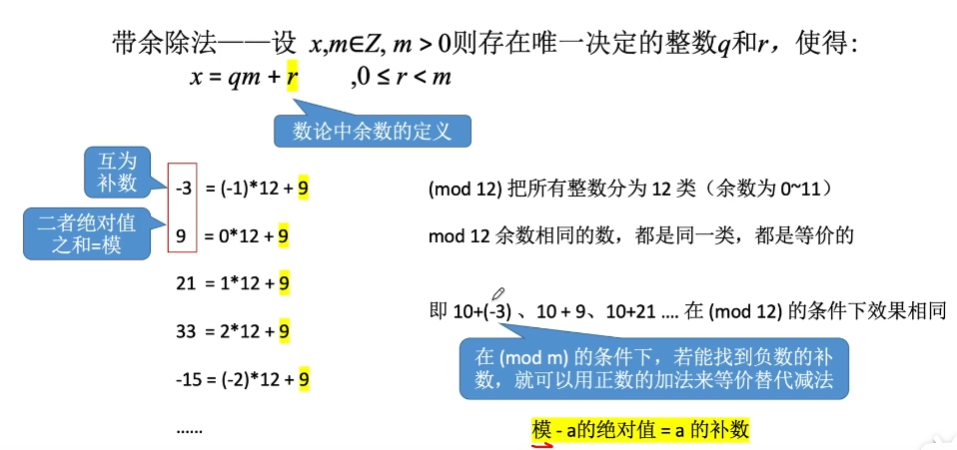
那么-3和+9就可以称为补码

所以：减去一个正数 => 加上一个负数 => 加上这个负数的补码，而这个补码在数值上是个正数（不考虑符号位）

这就可以解释为什么正数的补码是本身，而负数的补码要经过反码+1

因为，补码是为了将减法替换为加法，如果操作数本身就是正数，那就不需要进行转换，其补码取其本身就行了，而负数就得转换为正数了，即在计算机硬件中取反码+1（不考虑符号位，在数值上为正数）

# 模运算的性质



8位二进制数的范围是0000 0000 ~ 1111 1111，那么模就是1 0000 0000，根据取补数的公式：模 + 负数 = 负数的补数，那么-7（-00000111）的补码就是：

1 0000 0000 – 00000111 = 11111001

概括出补码的规则：最高位符号位取1，剩余位取反+1

# 例题一

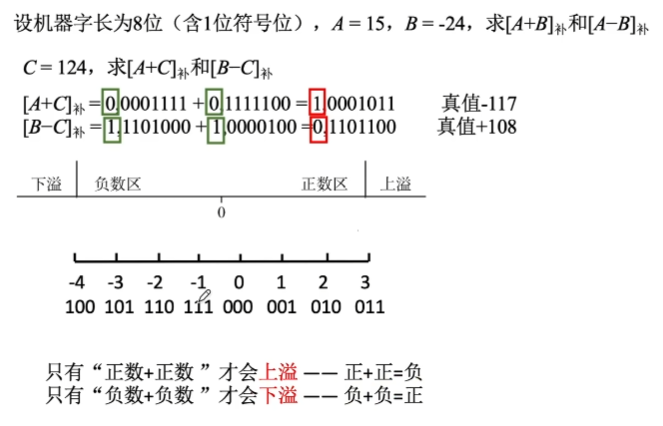
例题：A=15，B=7，求A-B，B-A，-A-B

总结：

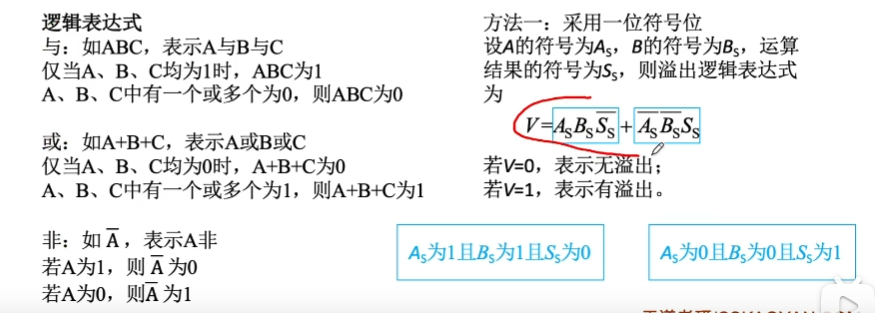
在不发生溢出的情况下，结果为正即为最终结果，结果为负取其补码；

如果结果超过了最高位，直接丢弃最高位。

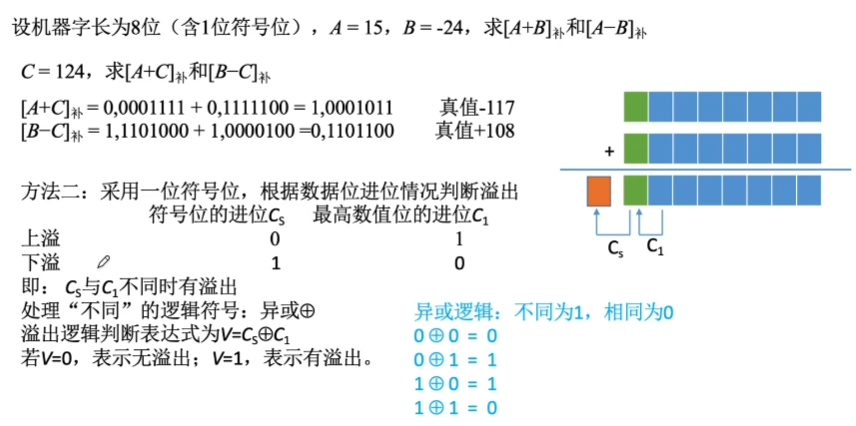
# 溢出判断



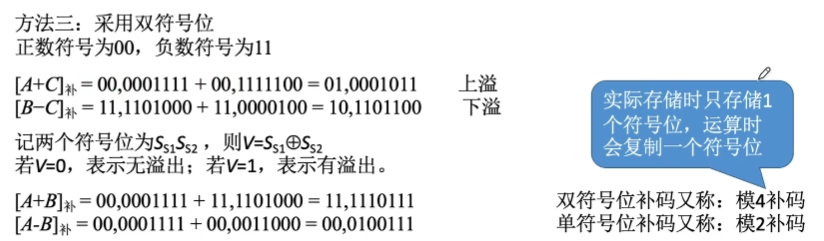
## 方法一



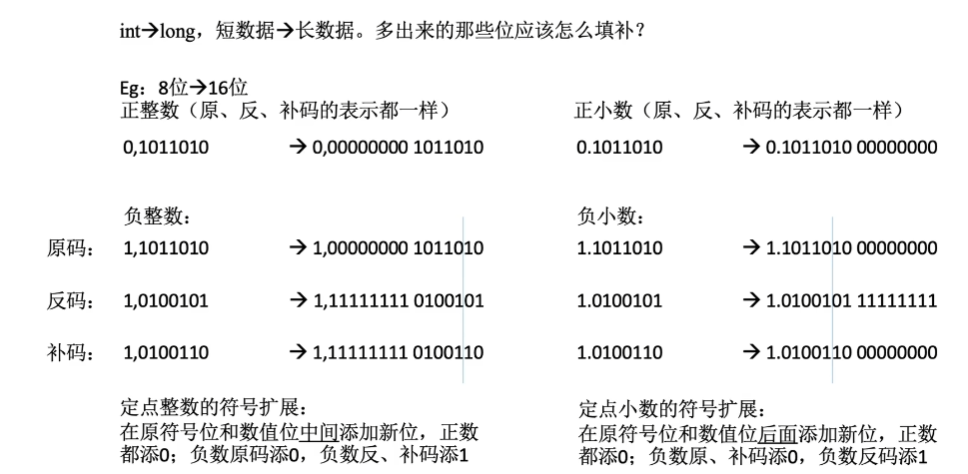
## 方法二



## 方法三（最常考）



# 符号扩展



# 例题二

A=125，B=9，求-A-B，A+B