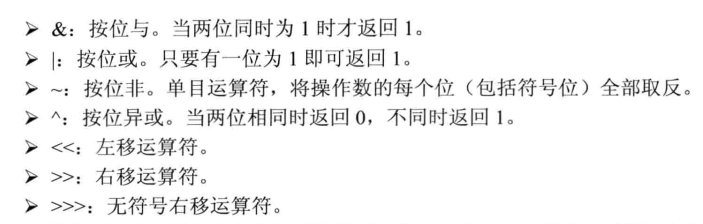
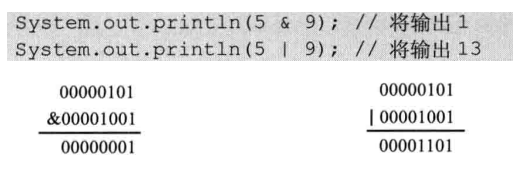
# 位运算符种类



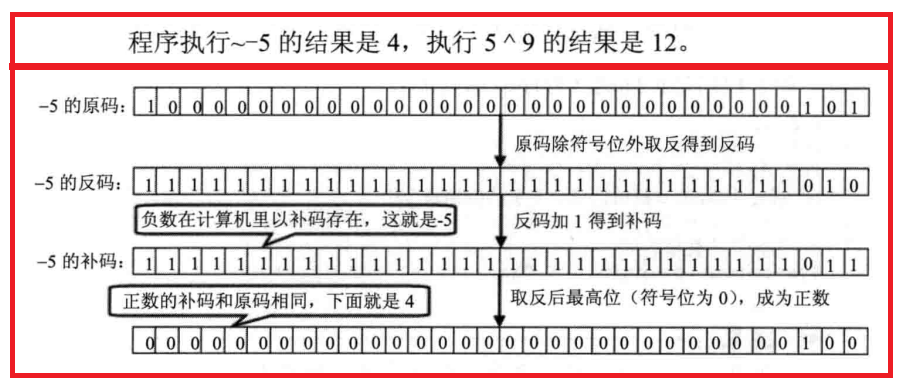


## 案例

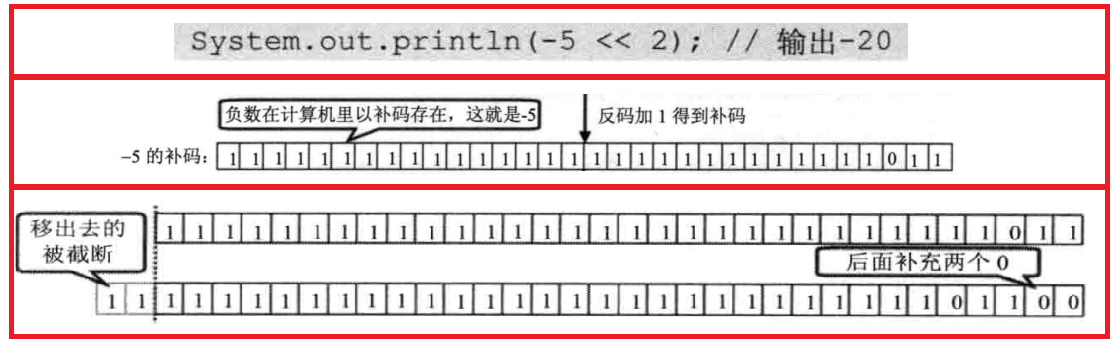
### 1.与或计算



### 2.非计算

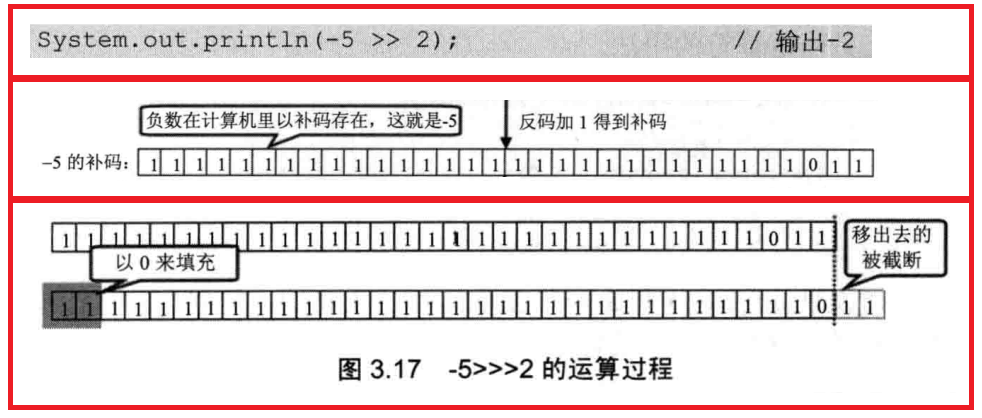


### 3.左移位补0

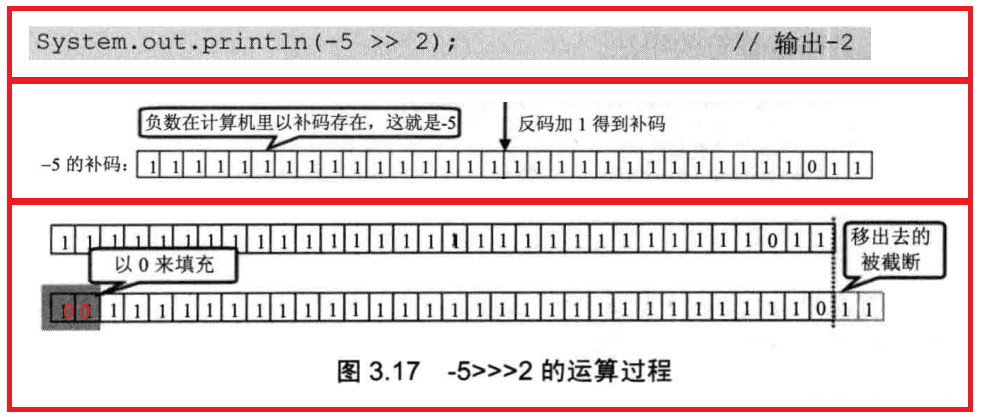


### 4.右移

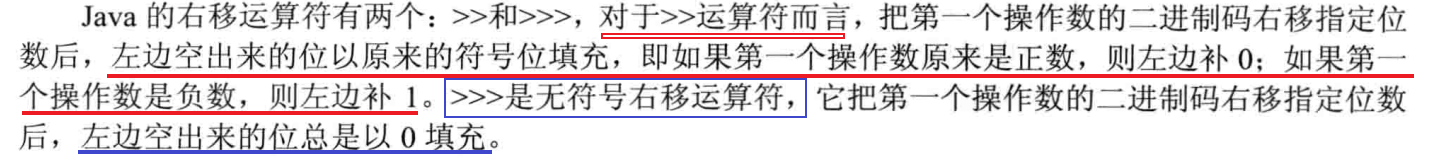
#### 右移位(>>)补符号位



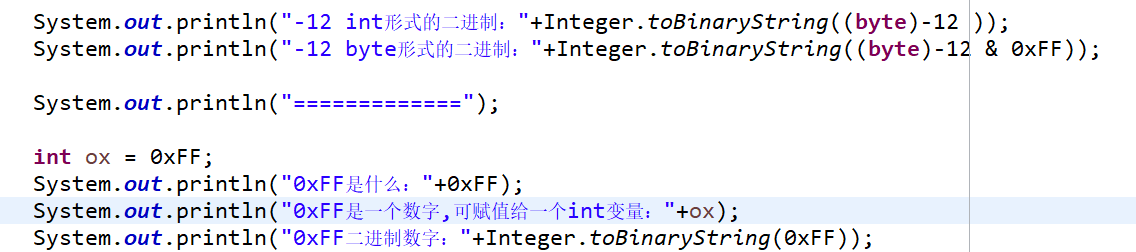
#### 无符号右移位(>>>)补0



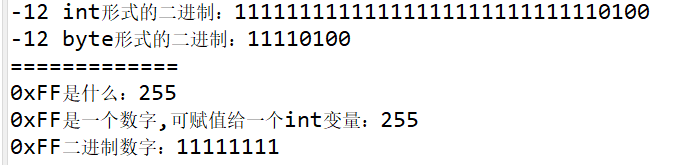
注意



## & 0xFF 表示什么



打印结果：



### byte&0xff的本质

byte类型的数字要&0xff再赋值给int类型，其本质原因就是想保持二进制补码的一致性。

### 产生不一致的原因

因为在计算机中正数存储的二进制原码, 负数存储的是二进制的补码（补码是负数的绝对值反码加1）。 也就是说正数与负数存储方式不一样。

而且在java中, byte要转化为int的时候会自动将byte转换为int，必然导致高的24位必然会补1，这样，其二进制补码其实已经不一致了。

### byte&0xff的作用

&0xff可以将高的24位置为0，低8位保持原样。这样做的目的就是为了保证二进制数据的一致性。

java中byte自动转化成int类型时，会将byte的内存空间高位补1（也就是按符号位补位）扩充到32位，再参与运算。

## 参考文章

1. [byte 为什么要 & 0xff？-重点看](https://blog.csdn.net/iblade/article/details/73289831)

2. [Java中byte做&0xff运算的原因及解析](https://blog.csdn.net/zhaominpro/article/details/79602381)