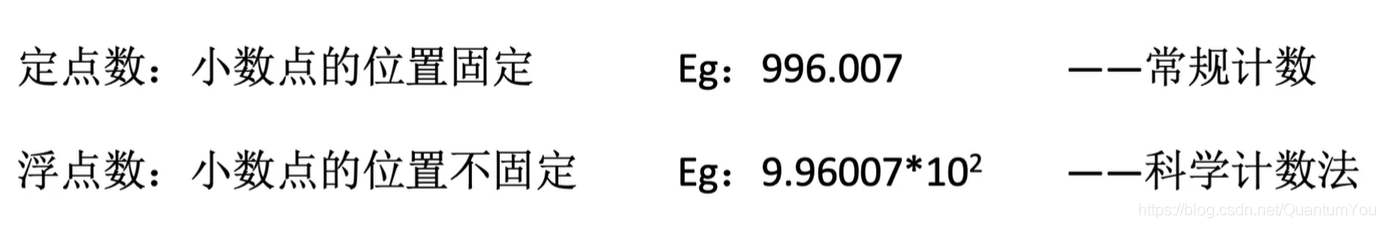
我很弱，但是我要坚强！绝不让那些为我付出过的人失望！

@[toc]

# 引入

定点数与浮点数的区别



# 定点数的表示

## 无符号数

无符号数：整个机器字长的全部二进制位均为数值位，没有符号位，相当于数的绝对值。

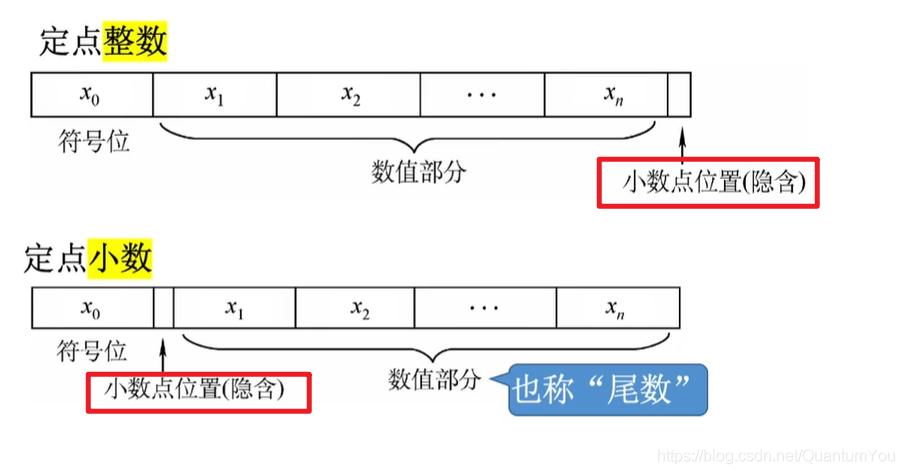
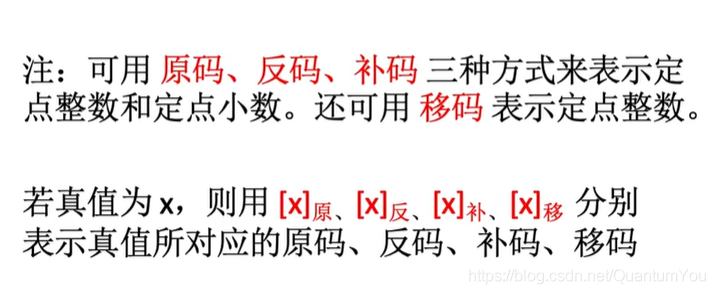


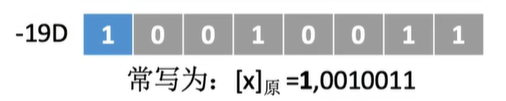
注意：

* 我们通常只有无符号整数，没有无符号小数
* eg: C 语言 unsigned 只能修饰 int long 等 不能修饰 float

## 有符号数

0 表示为正 1 表示为 负

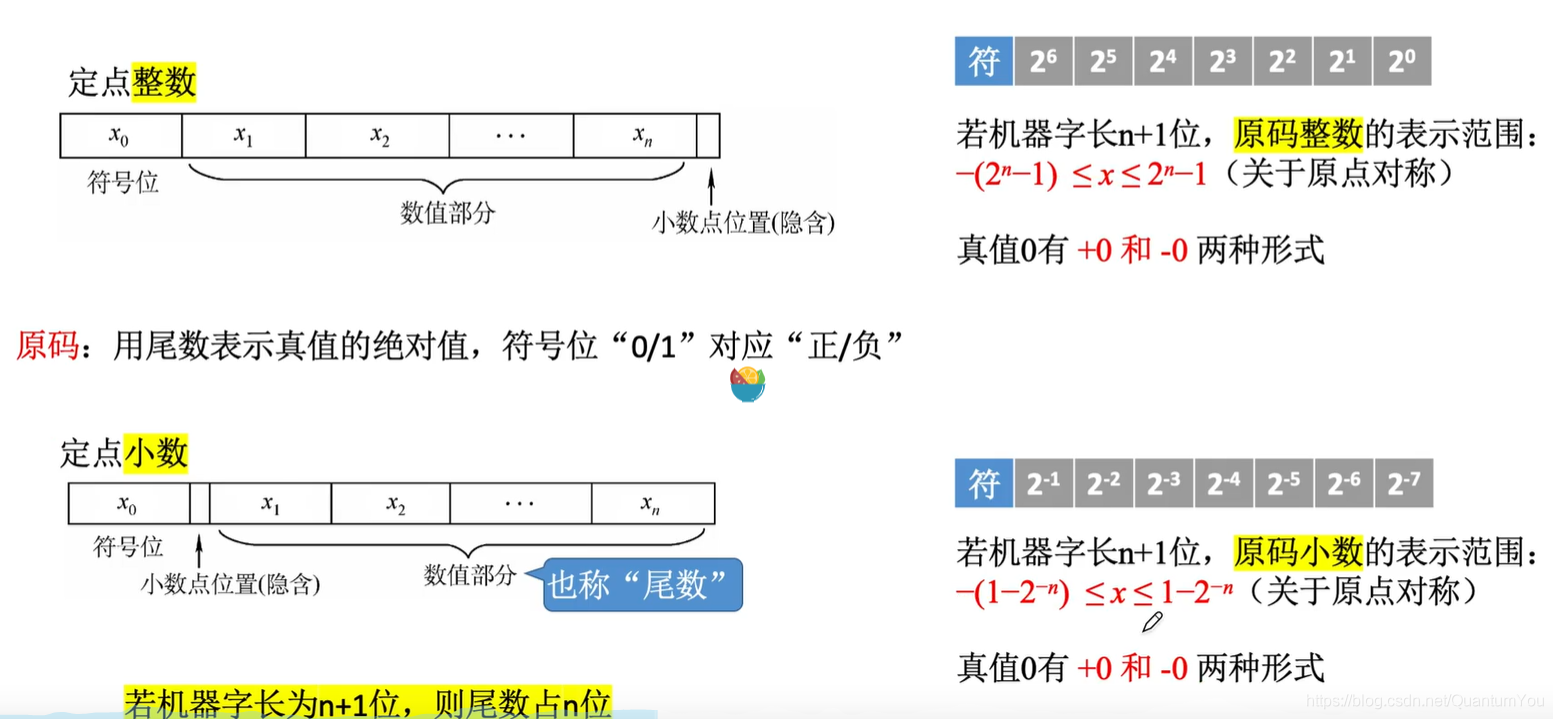
  
  
注意： 其中的 逗号 表示前面为 符号位



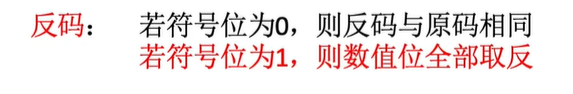
  
点号 前面 为符号

### 原码

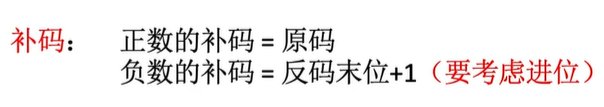
* 原码：用尾数表示真值的绝对值，符号位“0/1”对应“正/负”

**原码的表示范围：**  


### 反码

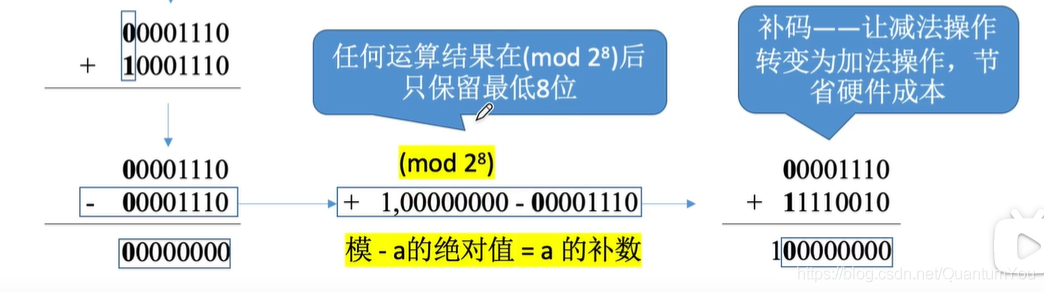


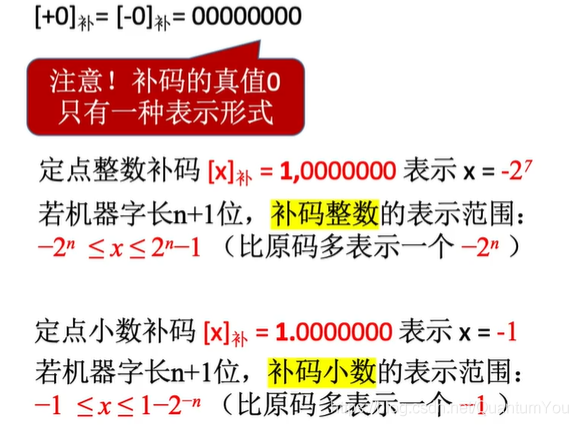
### 补码

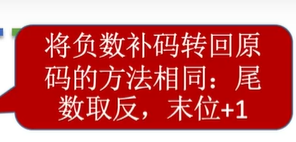


**补码的原理 ：**

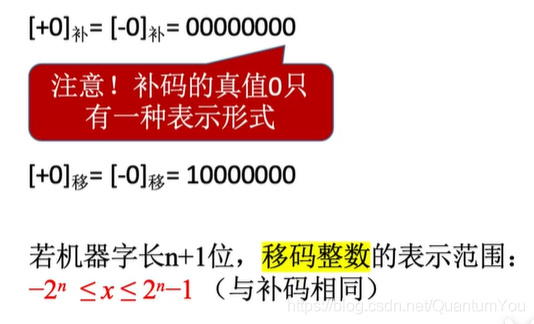
实际为 mod 运算



**补码的表示范围：**  


注意： 由补码 求 原码 --> 补码的补码 即为 原码  


### 移码

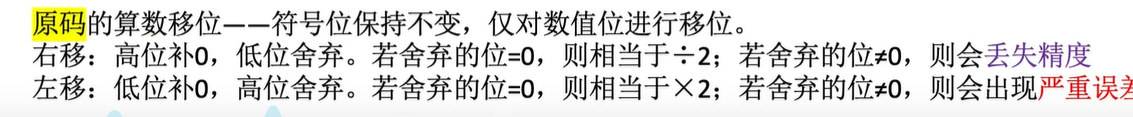
作用：  
移码表示的整数很方便对比大小

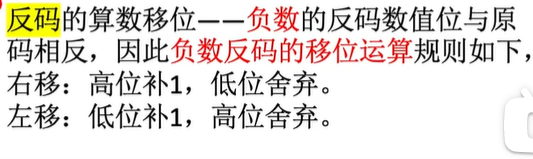
--> 从最高位开始 依次比较 先出现 1 的 表示的整数较大

## 定点数移位

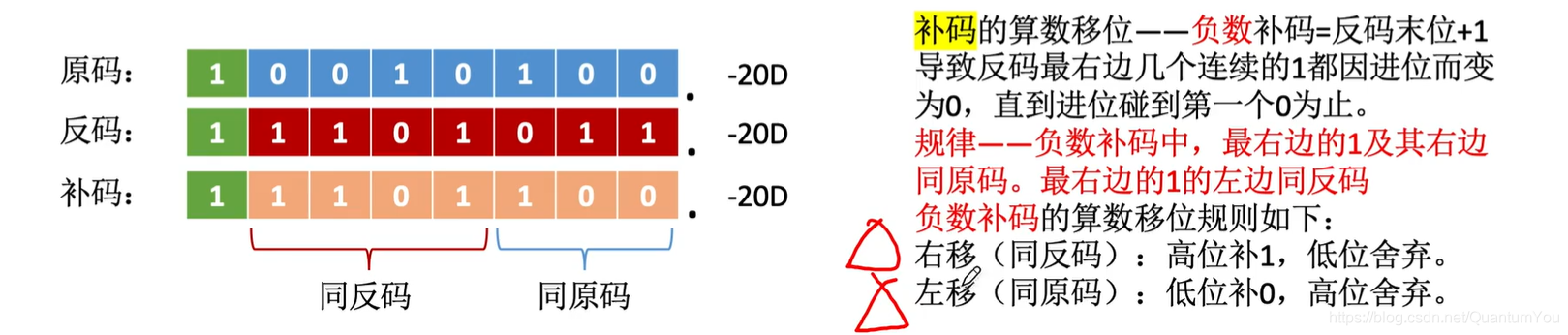
* **移位**：通过改变各个数仍位和小数点的相对位置，从而改变各数码位的位权。可用移位运算实现乘法、除法

### 算术右移（左移）





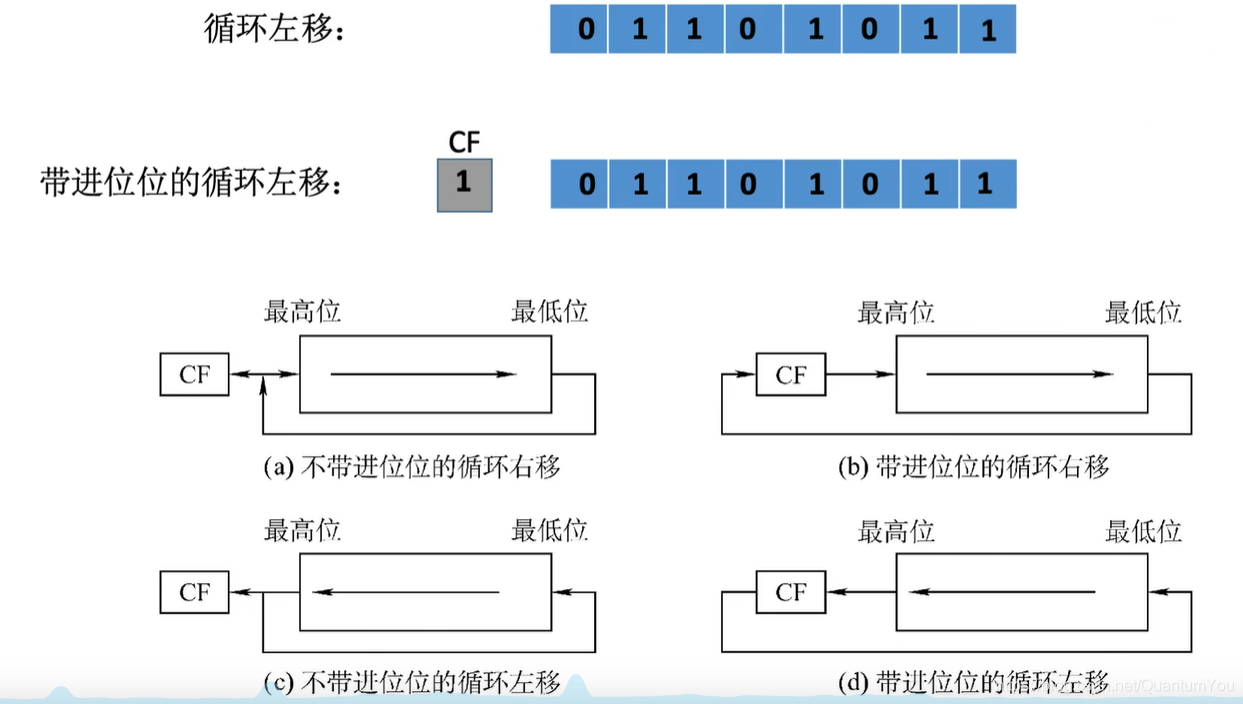
### 补码的算术右移



### 逻辑移位的应用

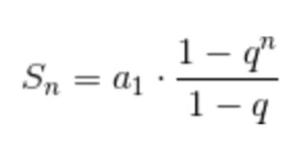


### 循环移位

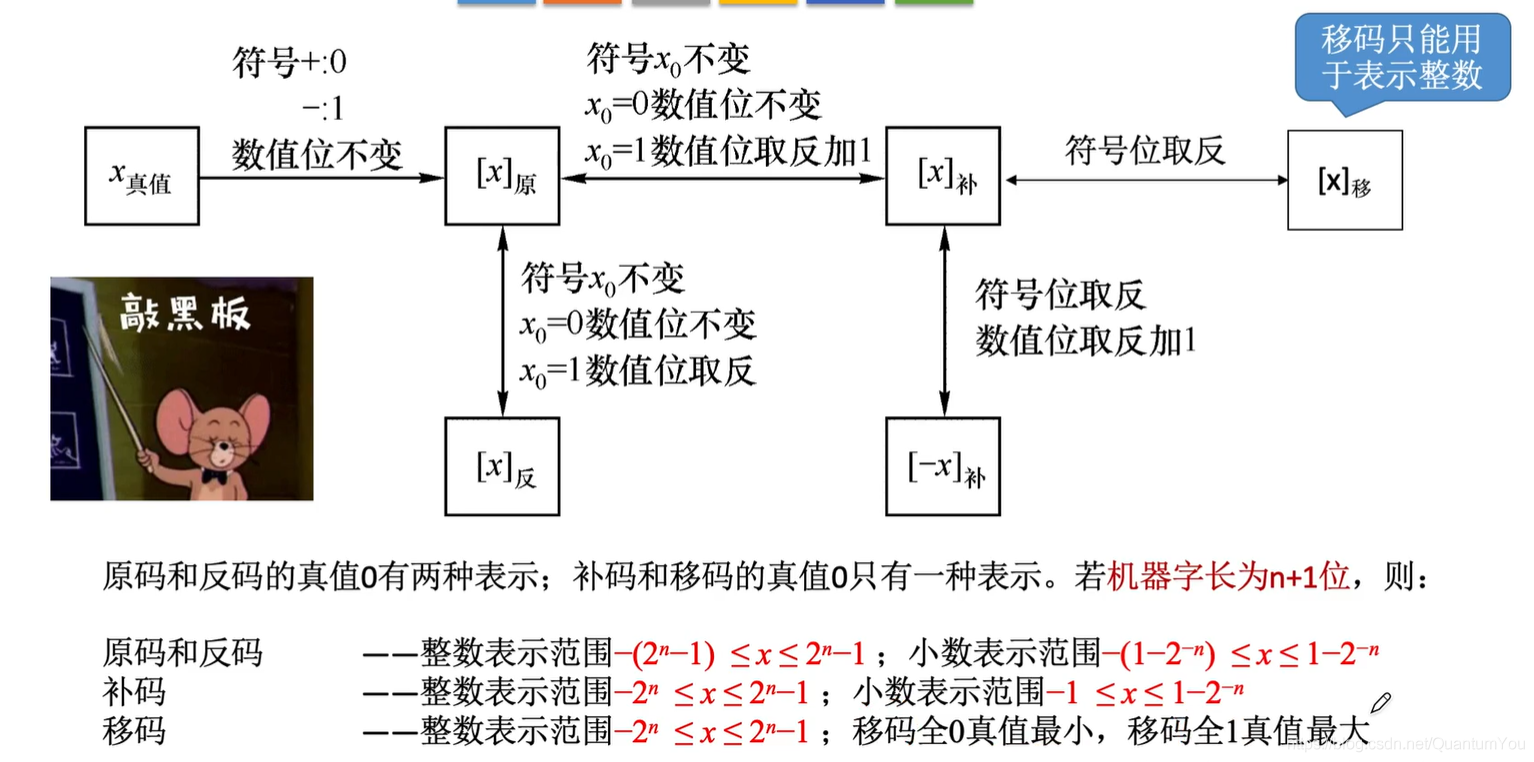


# Tips & Summary

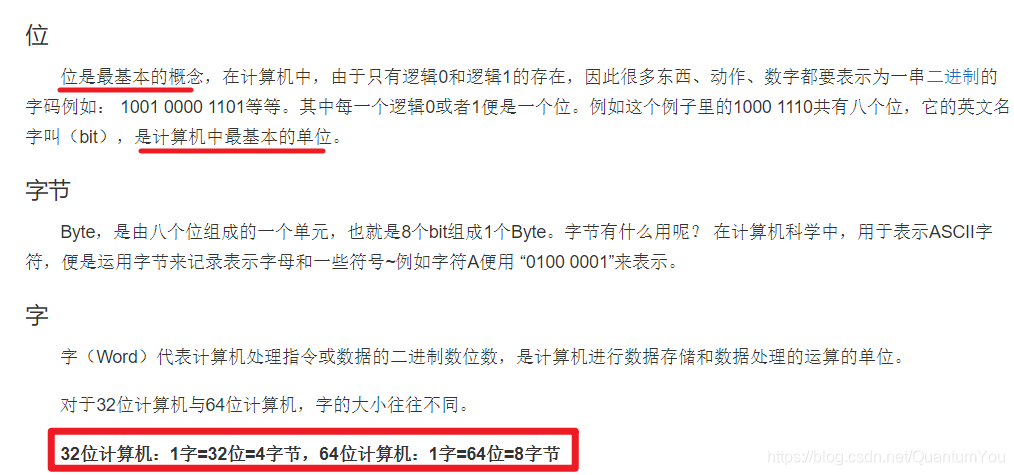
* 等比数列求和



* 原码数值范围：- (2n-1 ) ~ ( 2n-1 )
* 补码数值范围：- 2n ~ (2n-1)
* 反码数值范围：- (2n-1 ) ~ ( 2n-1 )



## 字 字节 位



* 1 Byte = 8 bit 1字节等于8比特