link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=53 sentences=74, words=430

1.计算机中的数值表示

- 进位计数制两大要素
    - 基数R
    - 位权W

1.1 基数R

- R代表基本数码的个数
- R进制的主要特点是逢R进1
- 基数R的数制称为R进制数

R 进制 数码符号 进制规则 R=2 二进制数(2) 0、1 逢2进1 R=10 十进制数(10) 0 ~ 9 逢10进1

1.2 位权Wi

- 位权Wi是指第i位上的数码的 **权重值**,位权与数码所处的位置 **i**有关
- 例如 R=10(十进制数),各个数码的权为10^i, i表示数码所处的位置
- 个位i=0,位权是10^0=1
- 十位i=1,位权是10^1=10
- (22.22)10=2x101+2x100+2x10-1+2x10-2

R 进制 数码符号 进制规则 位权Wi 例子 R=2 二进制数(2) 0、1 逢2进1 2 (11.11)2 R=10 十进制数(10) 0 ~ 9 逢10进1 10 (99.99)10

2.进制转换

2.1 二进制转十进制

- 方法: 按权展开,加权求和,以(111.11)2为例

2.2 十进制转二进制

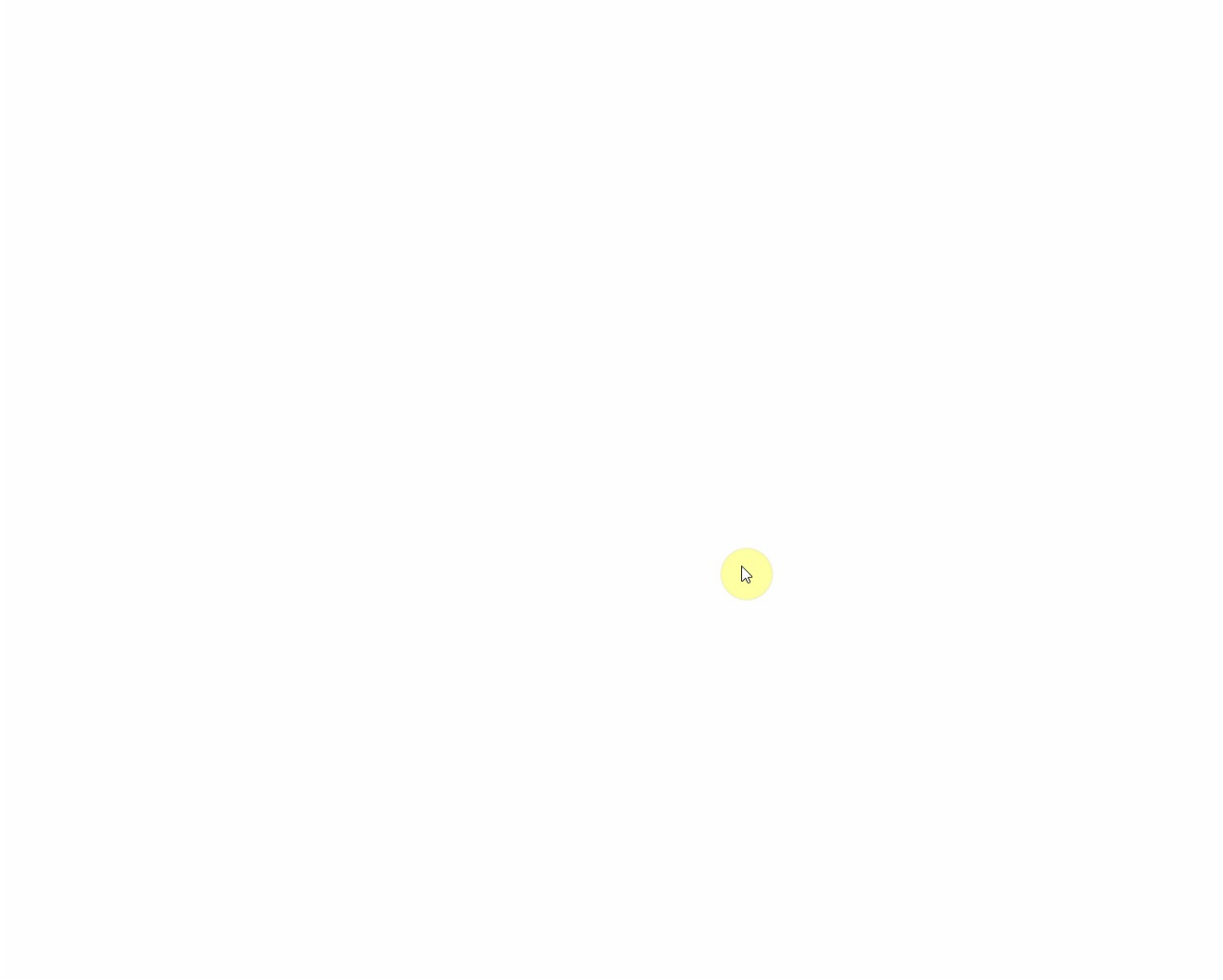
- 整数部分: 除2取余,直到商为0,最先得到的余数是最低位,最后得到的余数是最高位.
- 小数部分: 乘2取整,直到积为0或者达到精度要求为止,最先得到的整数是高位
- 例如 (7.75)10=(111.11)2

3. 计算机中的数据

- 数据
  - 数值数据
    - 无符号数据
    - 有符号数据
  - 数值数据
    - 文字
    - 图像

3.1 无符号数据的表示

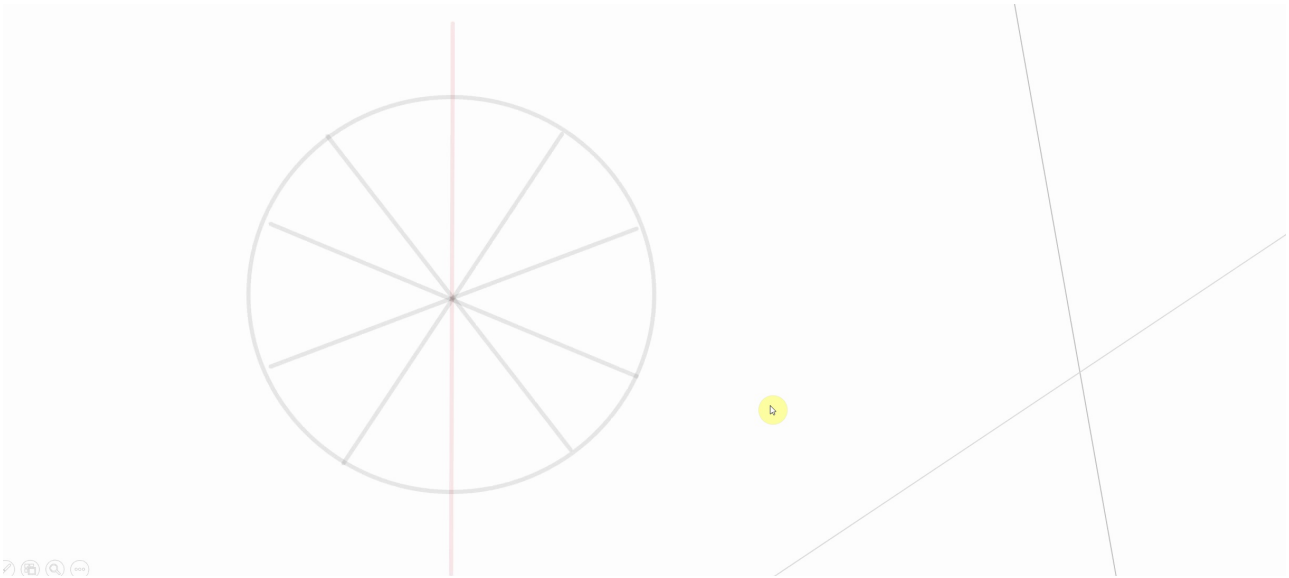
- 原码: 3个bit能表示8个数 0 1 2 3 4 5 6 7



3.2 有符号数据的表示

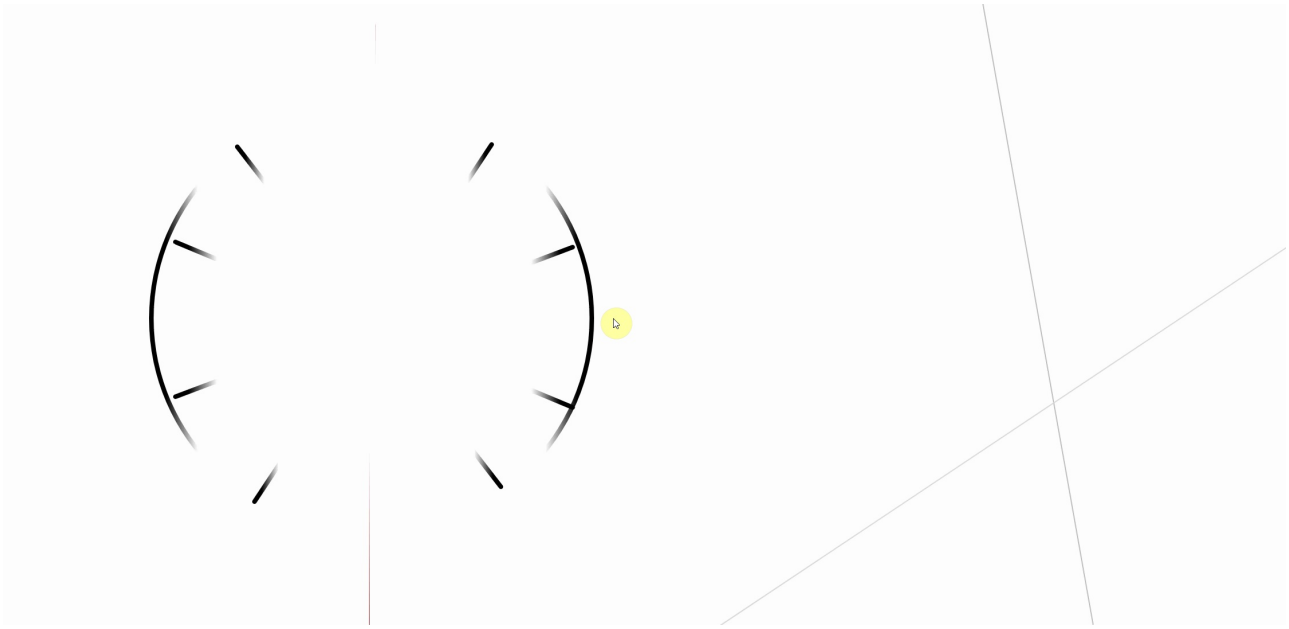
3.2.1 原码

- 原码: 3个bit能表示8个数 +0 +1 +2 +3 -0 -1 -2 -3
- 符号: 用0、1表示正负号,放在数值的最高位



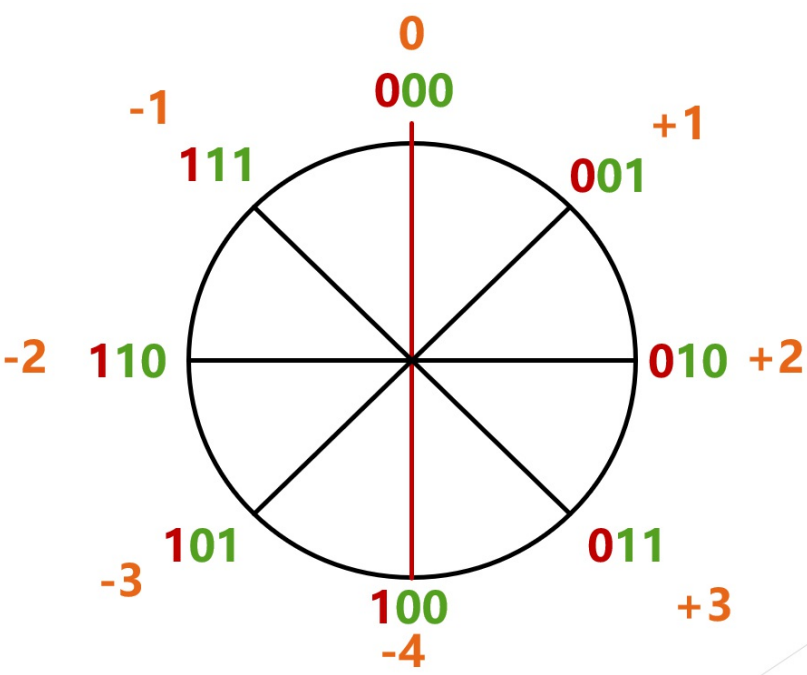
3.2.2 反码

- 反码:正数不变,负数的除符号位外取反



3.2.3 补码

- 补码:正数不变,负数在反码的基础上加1



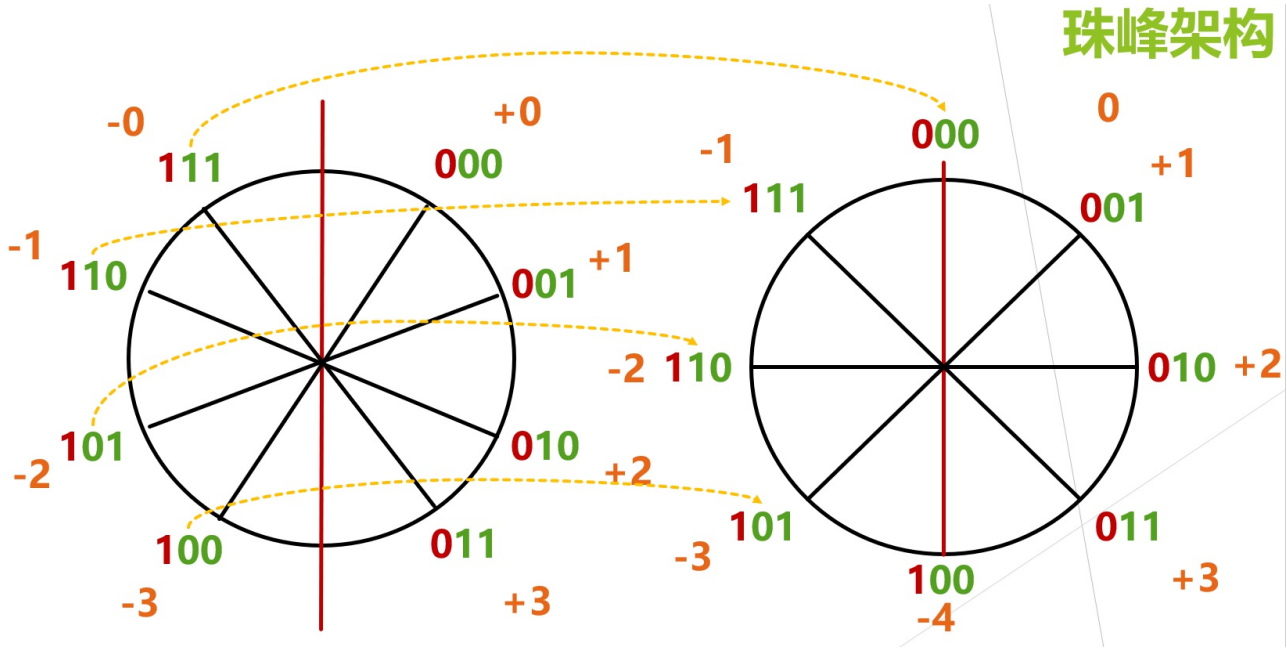
最高位溢出舍弃

001+111=~~1~~000

(+1)+(-1)=0

010+110=~~1~~000

(+2)+(-2)=0



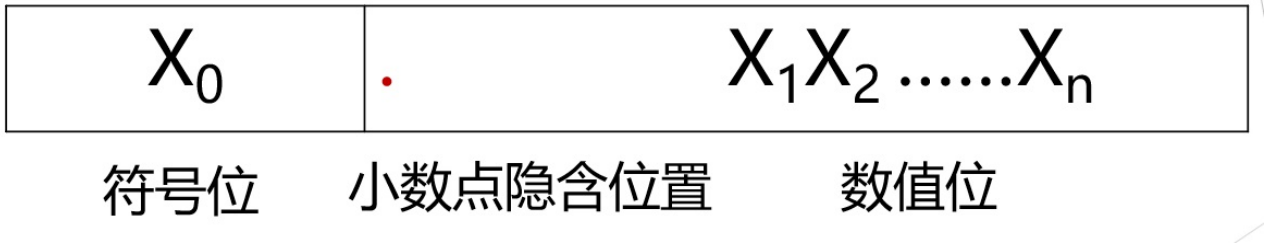
4. 小数

4.1 小数点表示

- 在计算机中,小数点及其位置是隐含规定的,小数点并不占用存储空间
- 定点数: 小数点的位置是固定不变的
- 浮点数:小数点的位置是会变化的

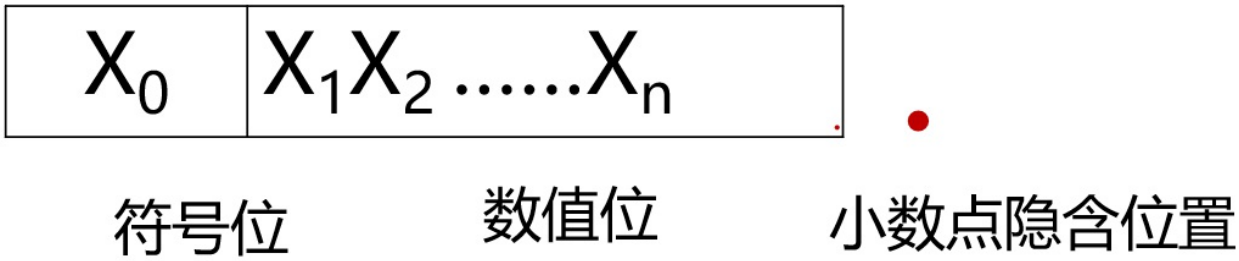
4.2 定点小数

- 定点小数: 小数点隐含固定在最高数据位的左边,整数位则用于表示符号位,用于表示纯小数. 例如: (0.110011)2



4.3 定点整数

- 定点整数: 小数点位置隐含固定在最低位之后,最高位为符号位,用于表示纯整数.
- 例如: (0110011.)2

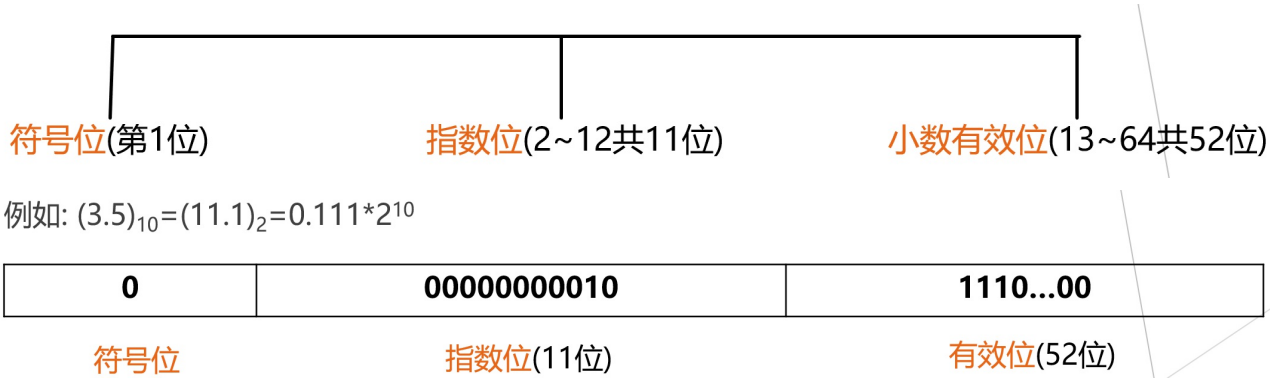


4.3 浮点数

- 对于既不是定点整数,也不是定点小数的数用浮点数表示
- 在计算机中通常把浮点数N分成 **阶码** 和 **尾数** 两部分来表示
- 小数点位置由 **阶码** 规定,因此是浮动的
- N=尾数*基数阶码,基中尾数是一个规格化的纯小数
- 例如8位系统中,阶码占3位,尾数占3位,数符和阶符各占1位
- (3.5)10=(11.1)2=0.111*210
- 科学记数法是一种记数的方法。把一个数表示成a与10的n次幂相乘的形式,可以节约空间和时间
- 例如:（1≤|a|

5.IEEE754标准

- JavaScript采用的是双精度(64位)
- 符号位决定了一个数的正负,指数部分决定了数值的大小,小数有效位部分决定了数值的精度
- 一个数在 JavaScript 内部实际的表示形式 (-1)符号位 _1.有效位_2指数位
- 精度最多53个二进制位, -(253-1)到253-1
- 指数部分最大值是 2017(211-1),分一半表示负数,JavaScript能够表示的数值范围是21024~2-102



6. 0.1+0.2 != 0.3 #

7.JS大数相加 #

- 列竖式方法 从低位向最高位计算的
  - 1.把原始数字进行倒序
  - 2.从个位起开始依次相加

```
let numA = "1234567890",numB = "123456789";
let numAArray = numA.split("").map((item) => parseInt(item)).reverse();
let numBArray = numB.split("").map((item) => parseInt(item)).reverse();
let sum = [].fill(0,0,(numA.length >= numB.length ? numA.length : numB.length) + 1);
for (let i = 0; i < numAArray.length; i++) {
  sum[i] = numAArray[i];
}
let up = 0;
for (let i = 0; i < numBArray.length; i++) {
  sum[i] = sum[i] + numBArray[i] + up;
  if (sum[i] > 9) {
    sum[i] = sum[i] % 10;
    up = 1;
  } else {
    up = 0;
  }
}
if (sum[sum.length - 1] == 0) {sum.pop()}
let result = sum.reverse().join("");
console.log(Number(numA) + Number(numB));
console.log(result);
```