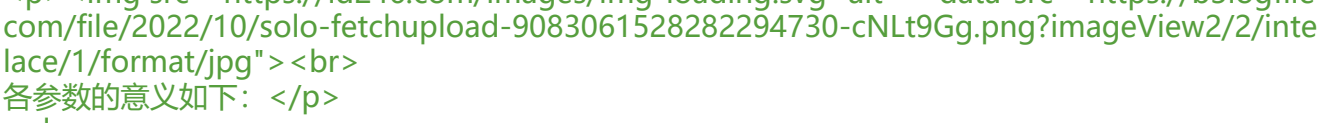
计算机组成原理 - 第 6 章 - 计算机的运算方法

作者：Anileh
原文链接：https://ld246.com/article/1667348727041
来源网站：链滴

本章主要介绍参与运算的各类数据（无符号数和有符号数、定点数和浮点数等），以及它们在计机中的算术运算方法。
<h2 id="6-1-无符号数和有符号数">6.1 无符号数和有符号数</h2>
1.码制比较
<table>
<thead>
<tr>
<th>类型</th>
<th>定义</th>
<th>表示范围(n+1 位，1 位符号位)</th>
<th>作用</th>
</tr>
</thead>
<tbody>
<tr>
<td>原码</td>
<td></td>
<td></td>
<td>带符号的绝对值表示</td>
</tr>
<tr>
<td>补码</td>
<td></td>
<td></td>
<td>消除减法运算</td>
</tr>
<tr>
<td>反码</td>
<td></td>
<td></td>
<td>由原码求补码或由补码求原码的中间过渡</td>
</tr>
</tbody>
</table>
2.相互转换

<h2 id="6-2-定点数和浮点数">6.2 定点数和浮点数</h2>
<h2 id="1-定点数的表示">1.定点数的表示</h2>
小数点按约定的方式标出，小数点位于

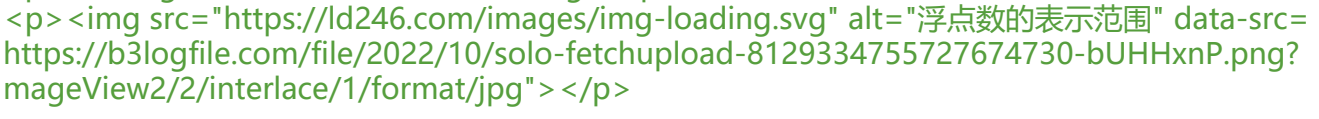
符号位和第一数值位之间：纯小数
数值位之后：纯整数

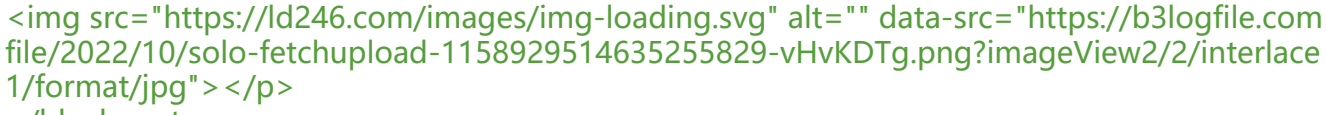

原码、补码、反码在定点数中的表示范围：
<table>
<thead>
<tr>
<th>定点机</th>
<th>小数定点机</th>
<th>整数定点机</th>
</tr>
</thead>
<tbody>
<tr>
<td>原码</td>
<td>-(1-2⁻ⁿ)~+(1-2⁻ⁿ)</td>
<td>-(2ⁿ-1)~+(2ⁿ-1)</td>
</tr>
<tr>
<td>补码</td>
<td>-1~+(1-2⁻ⁿ)</td>
<td>-2ⁿ~+(2ⁿ-1)</td>
</tr>
<tr>
<td>反码</td>
<td>-(1-2⁻ⁿ)~+(1-2⁻ⁿ)</td>
<td>-(2ⁿ-1)~+(2ⁿ-1)</td>
</tr>
</tbody>
</table>
<h2 id="2-浮点数的表示">2.浮点数的表示</h2>
1.浮点数的表示形式

各参数的意义如下：

<code>Sf</code>： 浮点数的符号位
<code>n</code>：尾数的位数，反映浮点数的精度
<code>m</code>：阶码的位数，反映浮点数的表示范围
<code>jf和m</code>：共同表示小数点的实际位置，~~这条没什么用~~

机器零：

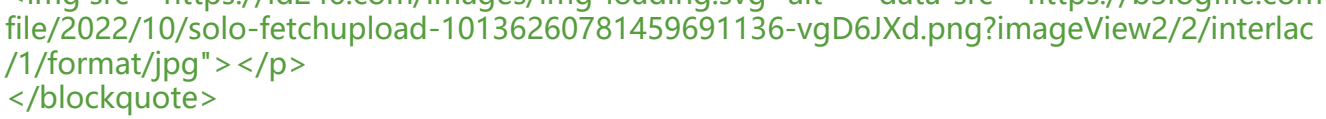
当浮点数尾数为 0 时，不论阶码为何值，按机器零处理
当浮点数阶码等于或小于所能表示的最小值时，不论尾数为何值，按机器零处理

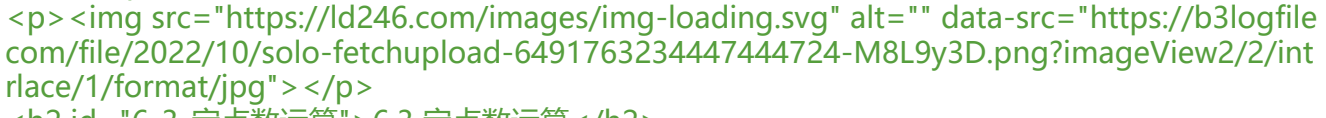
2.浮点数的表示范围


<blockquote>
练习：设机器字长为 24 位，与表示正负 3 万的十进制数，试问在保证数的最大精度的前提下，阶符、数符各取 1 位外，阶码、尾数各取几位？


</blockquote>
3.浮点数的规格化表示

基数 r 越大，可表示的浮点数的范围越大
基数 r 越大，浮点数的精度降低

<blockquote>
例 6.15


</blockquote>

<h2 id="6-3-定点数运算">6.3 定点数运算</h2>
<h2 id="1-移位运算">1.移位运算</h2>
1. 算术移位
有符号数，无论是正数还是负数，移位时符号位保持不变
算术移位规则：

正数：原码、补码、反码移位均添 0
负数：

原码：都添 0
补码：左移添 0，右移添 1
反码：都添 1

2. 逻辑移位
无符号数，移位规则：

逻辑左移，低位添 0，高位移丢；
逻辑右移，高位添 0，低位移丢.

<h2 id="2-加减运算">2.加减运算</h2>
1. 加减法运算

减法运算可以看做被减数加上一个减数的负值，即 A -B = A + (-B)

因此若机器数采用补码，当求 A- B 时，只需先求[-B]_补 就可按补码加法规则进运算

求 [-B]_补 ： 将[-B]_补 连同符号位在内每位取反。末位 +1

减法运算连同符号位一起运算，符号位产生的进位直接丢弃
<table>
<thead>
<tr>
<th>类型</th>
<th>加法</th>
<th>减法</th>
</tr>
</thead>
<tbody>
<tr>
<td>整数</td>
<td>[A]_补+[B]_补=[A+B]_补(mod 2^{n+1})</td>
<td>[A-B]_补 = [A]_补 + [-B]_补(mod 2^{n+1})</td>
</tr>
<tr>
<td>小数</td>
<td>[A]_补+[B]_补=[A+B]_补(mod 2)</td>
<td>[A-B]_补 = [A]_补 + [-B]_补(mod 2)</td>
</tr>
</tbody>
</table>

2. 溢出判断
补码定点加减法运算判断溢出有两种方法
<table>
<thead>
<tr>
<th>方法</th>
<th>溢出判断</th>
<th>硬件实现</th>
</tr>
</thead>
<tbody>
<tr>
<td>一位符号位</td>
<td>参加操作的两个数符号相同，而运算结果的符号与其不同</td>
<td>最高有效位的进位\bigoplus符号位的进位=1</td>
</tr>
<tr>
<td>两位符号位（变形补码）</td>
<td>2 位符号位不同即溢出，而无论是否溢出，高位符号位代表真实符号</td>
<td>符号位产生的进位\bigoplus最高有效位产生的进位=1</spa
></td>
</tr>
</tbody>
</table>
<h2 id="3-乘法运算">3.乘法运算</h2>
<h3 id="原码乘法">原码乘法</h3>
1.原码一位乘和原码两位乘

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile.com file/2022/10/solo-fetchupload-17432098625025846989-JHtzhly.png?imageView2/2/interlace 1/format/jpg">
原码一位乘运算规则：
<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile.com file/2022/10/solo-fetchupload-8251157085256209786-vPJShmc.png?imageView2/2/interlace 1/format/jpg">
2.原码两位乘
<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile com/file/2022/10/solo-fetchupload-11907853119215191682-CLsPBdO.png?imageView2/2/in erlace/1/format/jpg">
原码两位乘运算规则、
<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile.com file/2022/10/solo-fetchupload-1311086158614959036-bB2wYBG.png?imageView2/2/interlac /1/format/jpg">
原码一位乘和原码两位乘比较
<table>
<thead>
<tr>
<th>科目</th>
<th>原码一位乘</th>
<th>原码两位乘</th>
</tr>
</thead>
<tbody>
<tr>
<td>符号位</td>
<td>x_0\bigoplus y_0</td>
<td>x_0\bigoplus y_0</td>
</tr>
<tr>
<td>操作数</td>
<td>绝对值</td>
<td>绝对值的补码</td>
</tr>
<tr>
<td>移位</td>
<td>逻辑右移</td>
<td>算术右移</td>
</tr>
<tr>
<td>移位次数</td>
<td>n</td>
<td>\frac{n}{2},n 为偶数</td>
</tr>
<tr>
<td>最多加法次数</td>
<td>n</td>
<td>\frac{n}{2}+ 1，n 为偶数</td>
</tr>
</tbody>
</table>
<h3 id="补码乘法">补码乘法</h3>
1.补码一位乘

补码一位乘运算公式：

Booth 算法递推公式

补码乘法小结

<h2 id="4-除法运算">4.除法运算</h2>
<h3 id="原码除法">原码除法</h3>

特点：

上商 n + 1 次
第一次上商判溢出
移 n 次，加 n + 1 次
用移位的次数判断除法是否已经结束

<h2 id="6-4浮点数运算">6.4 浮点数运算</h2>
<h2 id="1-加减运算">1.加减运算</h2>
<blockquote>
步骤： 对阶、尾数求和、规格化、舍入
</blockquote>
1.对阶
先求阶差 j_{x}-j_{y} 判断正负，然后对阶
对阶原则：小阶向大阶看齐
尾数要根据阶码的移动而移动
2.尾数求和
补码求和，~~没啥好说的~~
3.规格化
规格化数的定义：r=2, \frac{1}{2}<=|S|<1
规格化数的判断：

原码：无论正负，第一数值位需要为 1
补码：符号位和第一数值位不同

<blockquote>
例：字长 12 位， 用定点补码规格化小数表示时,所能表示的正数范围是( )。
\frac{1}{2} ~ (1- 2^{-1})
</blockquote>
特例：[-\frac{1}{2}]_补 不是规格化的数，[-1]_补 是规格化的数
左规：尾数左移一位，阶码减 1，直到数符合第一数位不同为止
右规：当尾数溢出（>1）时，尾数右移一位，阶码加 1
4.舍入

在 对阶 和 右规 过程中，可能出现 尾数末位丢失 引起误差，需考虑舍入
0 舍 1 入法，恒置 “1” 法</p>