链滴

Java 基础 -- 位运算

作者：zeekling

原文链接：https://ld246.com/article/1577009925930

来源网站：链滴

<h2 id="简介">简介</h2>
程序中的所有数在计算机内存中都是以二进制的形式存储的。位运算(Bitwi
e operation)就是直接对整数在内存中的二进制位进行操作，因此其执行效率非常高。
<h2 id="详解">详解</h2>
Java 位运算细化划分可以分为按位运算和移位运算，见下表。
<table>
<thead>
<tr>
<th>符号</th>
<th>描述</th>
<th>运算规则</th>
<th>分类</th>
</tr>
</thead>
<tbody>
<tr>
<td>&</td>
<td>与</td>
<td>两位都为 1，那么结果为 1</td>
<td>按位运算</td>
</tr>
<tr>
<td>|</td>
<td>或</td>
<td>有一位为 1，那么结果为 1</td>
<td>按位运算</td>
</tr>
<tr>
<td>~</td>
<td>非</td>
<td>~0 = 1,~1 = 0</td>
<td>按位运算</td>
</tr>
<tr>
<td>^</td>
<td>亦或</td>
<td>两位不相同，结果为 1</td>
<td>按位运算</td>
</tr>
<tr>
<td><<</td>
<td>左移</td>
<td>各二进制位全部左移 N 位，高位丢弃，低位补 0</td>
<td>移位运算</td>
</tr>
<tr>
<td>>></td>
<td>右移</td>
<td>各二进制位全部右移 N 位，若值为正，则在高位插入 0，若值为负，则在高位插入 1</td>
<td>移位运算</td>
</tr>
<tr>
<td>>>></td>
<td>无符号右移</td>

<td>各二进制位全部右移 N 位，无论正负，都在高位插入 0</td>
<td>移位运算</td>
</tr>
</tbody>
</table>
在进行位运算详解之前，先来普及下计算机中数字的表示方法。对于计算机而言，万物皆 0、1 所有的数字最终都会转换成 0、1 的表示，有 3 种体现形式，分别是：原码、反码和补码。

原码：原码表示法在数字前面增加了一位符号位，即最高位为符号位，正位该位为 0，负数位该位为 1.比如十进制的 5 如果用 8 个二进制位来表示就是 00000101，-5 就是 1000101。
反码：正数的反码是其本身，负数的反码在其原码的基础上，符号位不变其余各个位取反。5 的反码就是 00000101，而-5 的则为 11111010。
补码：正数的补码是其本身，负数的补码在其原码的基础上，符号位不变其余各位取反，最后 +1。即在反码的基础上 +1。5 的反码就是 00000101，而-5 的则为 1111101。

了解了这几个概念后，我们现在先记住一个结论，那就是在计算机系统中，数字一律补码来表示、运算和存储，具体的原因可以看篇文章的讨论，这里不做更多讨论，因为不是本文的重点。
<h3 id="与运算---">与运算（&）</h3>
规则：转为二进制后，两位为 1，则结果为 1，否则结果为 0。
<h3 id="或运算---">或运算（|）</h3>
规则：转为二进制后，有一位为 1，则结果为 1，否则结果为 0。
<h3 id="非运算---">非运算（~）</h3>
规则：转为二进制后，~0 = 1,~1 = 0。
<h3 id="异或运算---">异或运算（^）</h3>
规则：转为二进制后，两位不相同，结果为 1，否则为 0。
<h3 id="左移运算----">左移运算（<<）</h3>
规则：转为二进制后，各二进制位全部左移 N 位，高位丢弃，低位补 0。
<h3 id="右移运算----">右移运算（>>）</h3>
规则：转为二进制后，各二进制位全部右移 N 位，若值为正，则在高位插入 0，若值为负，则高位插入 1。
<h3 id="无符号右移运算-----">无符号右移运算（>>>）</h3>
规则：转为二进制后，各二进制位全部右移 N 位，无论正负，都在高位插入 0。
<h2 id="应用">应用</h2>
<h3 id="不用额外的变量实现两个数字互换">不用额外的变量实现两个数字互换</h3>
见参考资料中的 BitOperationTest，方法 reverse 通过三次异或操作完成了两个变量值的替换。
证明很简单，我们只需要明白异或运算满足下面规律（实际不止如下规律）：
<pre><code class="highlight-chroma">0^a = a，a^a = 0;
a ^ b = b ^ a;
a ^ b ^ c = a ^ (b ^ c) = (a ^ b) ^ c;
a ^ b ^ a = b;
</code></pre>
假设 a,b 两个变量，经过如下步骤完成值交换：a=a^b,b=b^a,a=a^b。
证明如下：
　　因为 a ^ b = b ^ a，又 a=a^b,b=b^a。故 b=b^a= b^ (a^b)=a。
　　继续 a=a^b，a=（a^b) ^ b^ (a^b)，故 a=b。完成值交换。

<h3 id="不用判断语句实现求绝对值">不用判断语句实现求绝对值</h3>
公式如下：(a^(a>>31))-(a>>31)
先整理一下使用位运算取绝对值的思路：若 a 为正数，则不变，需要用异或 0 保持的特点；若 a 为负数，则其补码为原码翻转每一位后 +1，先求其原码，补码-1 后再翻转每一位，此时需要使用异或 1 具有翻转的特点。
任何正数右移 31 后只剩符号位 0，最终结果为 0，任何负数右移 31 后也只剩符号位 1，溢出的 1 位截断，空出的 31 位补符号位 1，最终结果为-1.右移 31 操作可以取得任何整数的符号位。
那么综合上面的步骤，可得到公式。a>>31 取得 a 的符号，若 a 为正数，a>>31 于 0，a^0=a，不变；若 a 为负数,a>>31 等于-1 ，a^-1 翻转每一位。
<h3 id="判断一个数的奇偶性">判断一个数的奇偶性</h3>
通过与运算判断奇偶数，伪代码如下：
<pre><code class="highlight-chroma">n&1 == 1?”奇数”:”偶数”
</code></pre>
奇数最低位肯定是 1，而 1 的二进制最低位也是 1，其他位都是 0，所以所有奇数和 1 与运算结肯定是 1。
<h3 id="查找落单的数">查找落单的数</h3>
将数组的数全部做异或，最后得到的数就是要找的数，因为和一个数做两次异或不会改变。
<blockquote>
参考文章: 一文搞懂位运算
</blockquote>