位操作

位操作是程序设计中对位模式或二进制数的一元和二元操作。在许多古老的微处理器上，位运算比加减运算略快，通常位运算比乘除法运算要快很多。在现代架构中，情况并非如此：位运算的运算速度通常与加法运算相同（仍然快于乘法运算）。

目录

位运算符

取反（NOT）

取反是一元运算符，对一个二进制数的每一位执行逻辑反操作。使数字1成为0，0成为1。例如：

```
NOT 0111（十进制7）
    = 1000（十进制8）
```

许多程序设计语言（包括C程序设计语言family），取反操作符用波浪线"~"表示。值得注意的是此操作符与"逻辑非（!）"操作符不同。在C++中，逻

辑非将数字整体看做一个布尔类型--将真值转化为假，将假值转化为真；而C语言将0转化为1，将非零值转化为0。"逻辑非"并不是一个位操作。

按位或（OR）

按位或处理两个长度相同的二进制数，两个相应的二进位中只要有一个为1，该位的结果值为1。例如

```
   0101（十进制5）
OR 0011（十进制3）
  = 0111（十进制7）
```

在C类程序设计语言中，按位或操作符是"|"。这一操作符需要与逻辑或运算符（||）区别开来。

按位或能够将每一位看做旗帜；在二进制数中的每一位可以表示不同的布尔变量。应用按位或操作可以将二进制数的某一位设为1。例如

```
0010（十进制2）
```

能够看做包含4个旗帜的组合。第1，2，4旗帜为0；第3个旗帜为1。利用按位或可以将第1个旗帜设置为1，而其他旗帜不变。

```
   0010（十进制2）
OR 1000（十进制8）
  = 1010（十进制10）
```

这一技巧通常用来保存程序中的大量布尔变量。

按位异或（XOR）

按位异或运算，对等长二进制模式或二进制数的每一位执行逻辑异或操作。操作的结果是如果某位不同则该位为1，否则该位为0。例如

```
   0101
XOR 0011
  = 0110
```

在类C语言中，按位异或运算符是"^"。

汇编语言的程序员们有时使用按位异或运算作为将寄存器的值设为0的捷径。用值的自身对其执行按位异或运算将得到0。并且在许多架构中，与直接

加载0值并将它保存到寄存器相比，按位异或运算需要较少的中央处理单元时钟周期。

按位异或也可以用于在比特集合中切换旗帜。给出一个比特模式，

```
0010
```

第一和第三位能够通过按位异或运算使用同时切换。

```
    0010
XOR 1010
  = 1000
```

这一技巧可用于操作表示布尔变量的比特模式。

按位与（AND）

按位与处理两个长度相同的二进制数，两个相应的二进位都为1，该位的结果值才为1，否则为0。例如：

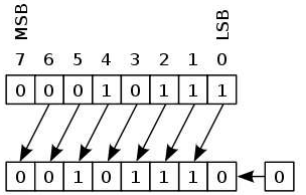
```
    0101
AND 0011
  = 0001
```

在类C语言中，按位与用'&'表示

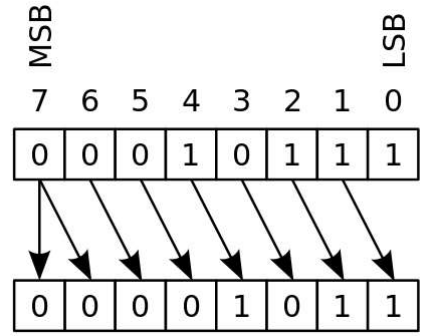
移位

移位是一个二元运算符，用来将一个二进制数中的每一位全部都向一个方向移动指定位，溢出的部分将被舍弃，而空缺的部分填入一定的值。在类C语言中，左移使用两个小于符号"<<"表示，右移使用两个大于符号">>"表示。

算术移位



Left arithmetic shift



150px arithmetic shift

逻辑移位

应用逻辑移位时，移位后空缺的部分全部填0。

```
      0001（十进制1）
<<       3（左移3位）
      = 1000（十进制8）
```

```
      1010（十进制10）
 >>       2（右移2位）
= 0010（十进制2）
```

C，C++和Java中的移位

JAVA中有一个特有的无符号右移操作符“>>>”。此操作将忽略操作数的符号 同样的还有>>>=

应用

- 本页面最后修订于2016年9月23日 (星期五) 11:58。