位运算符基础

1. 基础规定

(1)位就是 2 进制数字中的每一个位，一个整数数字，有 32 个位构成！

(2)位运算符是仅仅针对整数进行的运算符

(3)位运算符有如下几个：&按位与、|按位或、~按位非(按位取反)、^按位异或

(4)位运算符的基本规则

按位与

1 & 1 ➔ 1

1 & 0 ➔ 0

0 & 1 ➔ 0

0 & 0 ➔ 0

按位或

1 | 1 ➔ 1

1 | 0 ➔ 1

0 | 1 ➔ 1

0 | 0 ➔ 0

按位非

~1 ➔ 0

~0 ➔ 1

按位异或

1 ^ 1 ➔ 0

1 ^ 0 ➔ 1

0 ^ 1 ➔ 1

0 ^ 0 ➔ 0

可见，按位异或规则是：相同为 0，不同为 1

2. 整数的按位与运算

形式：

$n1 &n2; //n1,n2 是 2 个任意整数

含义：

将该 2 个整数的 2 进制数字形式(都是 32 位)的每一个对应位上的数字进行基本按位与运算之后的结果

注意：

运算的结果，还是一个普通数字，只是内部运算用了二进制计算

图示：

10 & 20

10 的二进制	00	0	0	1	0	1	0
20 的二进制	00	0	1	0	1	0	0
&运算结果	00	0	0	0	0	0	0

结果为 0

3. 整数的按位或运算

形式：

$n1 | n2; //n1,n2 是 2 个任意整数

含义：

将该 2 个整数的 2 进制数字形式(都是 32 位)的每一个对应位上的数字进行基本按位或运算之后的结果

注意：

运算的结果，还是一个普通数字，只是内部运算用了二进制计算

图示：

15 | 18

15 的二进制	00	0	0	1	1	1	1

18 的二进制	00	0	1	0	0	1	0
&运算结果	00	0	1	1	1	1	1

结果为 31，最大是两个相加，最小为最小一个

4. 整数的按位左移动运算

形式：

&n1 << &m

含义

将数字 n1 的二进制形式的每一个位上的数字都一次性往左边移动 m 位，并将右边空出来的位置补 0，左边冒出去的不管，这样得到的结果

图示

$r1 = 10 << 2;

10 的二进制	0	0	0	0	1	0	1	0
左移 2 位后	0	0	1	0	1	0	0	0
<<运算结果			2^5		2^3		0	

可见结果为 32 + 8 = 40

5. 补充知识：原码，反码，补码

(1) 原码：

就是一个二进制数字，从“数学观念”上表达出的形式，其中我们规定，一个数字最左边位为符号位，0 表示整数 1 表示负数

比如 5 的原码

原码:00000000 0000000 0000000 00000101

(2) 反码：

正数的反码就是其本身，即不变

负数的反码为：符号为不变，其他位取反

如-3 的原码和反码

原码:10000000 00000000 00000000 00000011

反码:11111111 11111111 11111111 11111100

(3) 补码

正数的补码就是其本身，即不变

负数的补码为：符号为不变，其他位取反后+1，即补码+1

如-3 的三码

原码:10000000 00000000 00000000 00000011

反码:11111111 11111111 11111111 11111100

补码:11111111 11111111 11111111 11111101

注意：cpu 内部的所有运算都是用补码计算的，无法从直观简单地理解运算过程

6. 演示

(1)5+3 的 cpu 运算过程

5 的补码：00000000 00000000 00000000 00000101

3 的补码：00000000 00000000 00000000 00000011

运算结果：00000000 00000000 00000000 00001000，即 2^3=8

(2)5-3 的 cpu 运算过程

5 的补码：00000000 00000000 00000000 00000101

3 的补码：11111111 11111111 11111111 11111101

运算结果：00000000 00000000 00000000 00000010，即 2^1=2