Bit Operations

|  |  |  |
| --- | --- | --- |
| Name | Operator | Purpose |
| Bitwise Not | ~x |  |
| Bitwise And | x & y | Intersection of 2 bit strings |
| Bitwise Or | x | y | Union of 2 bit strings |
| Bitwise XOR | x ^ y | 1) *Comparison of 2 bit strings, 0 -> same, 1 - different*  2) *p (+) q, either one is true*  3) *a (+) b (+) c +... n,*  *true if odd number of bits set if true*  4) *a (+) b (+) c +... n, true if odd number of set if true=> ( a + b + c +... +n) %2 <-- summing up all values then see even/odd*   5) With it, you can do parity checks and swaps without temp variables |
| Bitwise shift LEFT by K bit | x << K  x: an int or a char,  K: an int | *Move the bits of x to the left y positions* |
| Bitwise shift Right by K | x >> K | *Move the bits of x to the right y positions* |
| Bitwise unsigned >> by | x >>> K | *Move the bits of x to the right y positions* |

Number to Base N

**Definition** A **digit** is a single symbol used to represent a value in base K.

<dN,dn-1,dn-2....d3,d2,d1,d0> of base K



To convert a number written in base K, with N digits left of the radix point and M digits right of the radix point, use the following summation: 



If there are no values right of the radix point, that is, if M=0, then the number of numeric "patterns" (i.e., the number of possible values) of an N digit base-K number is **KN**.

Big Endian

Little Endian