Understanding Basic Binary Arithmetic

Working with IP addresses, especially for subnetting and supernetting is much simpler if you understand the basics of binary arithmetic. For the purposes of this text, you must master four different kinds of binary calculations:

- Converting decimal to binary
- Converting binary to decimal
- Understanding how setting increasing numbers of high-order bits to 1 in 8-bit binary numbers corresponds to specific decimal numbers
- Understanding how setting increasing low-order bits to 1 in 8-bit binary numbers corresponds to specific decimal numbers

Converting Decimal to Binary

This is extremely easy, if you don't mind thinking mathematically. We have two approaches to the problem.

The following approach has the advantage of working for any number, no matter how large or small. Simply divide the number by 2, write the remainder (which must be 0 or 1), then write the dividend, and repeat until the dividend is 0.

Let's convert the decimal number 125 to binary, as an example:

```
125 divided by 2 equals 62, remainder 1
62 divided by 2 equals 31, remainder 0
31 divided by 2 equals 15, remainder 1
15 divided by 2 equals 7, remainder 1
7 divided by 2 equals 3, remainder 1
3 divided by 2 equals 1, remainder 1
1 divided by 2 equals 0, remainder 1
```

To produce the binary number that corresponds to 125, you write the digits starting with the last remainder value, and work your way up: 1111101. Now, let's check the work involved. The exponential expansion of 1111101 is $1*2^6 + 1*2^5 + 1*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$ (1111101).

An alternate approach to convert the number depends on what mathematicians like to call a "step function." This particular approach depends on knowing the decimal values for various powers of two and then applying steps similarly to the first approach, except that each number must be positioned between the nearest power of two that's greater than or equal to the number in question, and less than or equal to the number in question. Here's an example:

```
125 is less than 128 (2^7) and more than 64 (2^6)
125 minus 64 is 61
61 is less than 64 (2^6) and more than 32 (2^5)
61 minus 32 is 29
29 is less than 32 (2^5) and more than 16 (2^4)
29 minus 16 is 13
13 is less than 16 (2^4) and more than 8 (2^3)
```

```
13 minus 8 is 5
5 is less than 8 (2³) and more than 4 (2²) 5
minus 4 is 1
```

Note that although there's an entry for 2^6, 2^5, 2^4, 2^3, 2^2, and 2^0, there is no entry for 2^1, so it shows up as 0 when converted to binary (any number you subtract from a previous result produces a 1 for the corresponding power of two; a missing number produces a 0). One is always 1, even in binary. Thus, $125 = 1{*}2^6 + 1{*}2^5 + 1{*}2^4 + 1{*}2^3 + 1{*}2^2 + 0{*}2^1 + 1{*}2^0$. Read the multipliers from left to right to convert to binary, and you end up with 1111101; for octet notation, add a leading 0 to make that number 01111101.

You should practice on some numbers of your own to make sure you understand one method or the other.

Converting Binary to Decimal

This is extremely easy, if you know your powers of two. Follow these steps, using 11011011 as the example number:

1. Count the total number of digits in the number (11011011 has eight digits).
2. 1 from the total (8 – 1 = 7). That is the power of two to associate with the highest exponent for two in the exponential notation for that number.
3. Convert to exponential notation, using all the digits as multipliers. 11011011, therefore, converts as shown:

$$11011011 = 1 * 2^7 + 1 * 2^6 + 0{*}2^5 + 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$$

$$= 128 + 64 + 0 + 16 + 8 + 0 + 2 + 1 = 219$$

Practice this on some numbers of your own to make sure you understand the method.

High-Order Bit Patterns

Sometimes, we block off bits in 8-bit numbers from the top down. (These are called the most significant bits because they represent the highest possible numeric values.) In an 8-bit number, there's little or no value in blocking fewer than 2 bits, or more than 6 bits, so the bit patterns you care about most appear in the second through the sixth positions in this list of possibilities:

Binary	Decimal
10000000	128
11000000	192
11100000	224
11110000	240
11111000	248
11111100	252
11111110	254
11111111	255

If you simply memorize these correspondences, then you'll be well equipped to deal with subnet masking problems.

Low-Order Bit Patterns

Here, we stand the previous example on its head and start counting up through the bit positions in 8-bit numbers from right to left, adding 1s as we increment. Note that each of these numbers is the same as 2 raised to the power of the number of bits showing, minus 1. If you memorize the values of the powers of two from 1 through 8, you can calculate this table in your head in a flash!

Binary	**Decimal**	**Exponent**
00000001	1	21 - 1
00000011	3	22 - 1
00000111	7	23 - 1
00001111	15	24 - 1
00011111	31	25 - 1
00111111	63	26 - 1
01111111	127	27 - 1
11111111	255	28 - 1

Memorizing these numbers, or learning how to calculate them, will prepare you to deal with supernet masking problems. As with subnet masks, you seldom deal with supernet masks of more than 4 to 6 bits, but these numbers are easy enough to calculate on demand, and you should learn how to do so.