Binary Number System Notes

Introduction

Most of us have worked in the deca system our entire life. Computers however use an entirely different number system to operate.

Why is it Important?

Computers at their most basic level use electricity to operate. Electricity only has two reliable states, on(1) and off(0). For this reason, computers use the binary number system, which only operate off two numbers, 0 and 1. Computers don't have the ability to use the deca system. Since computer science is the study of computers, it is helpful that we understand the math that computers use, so we can better understand their logic.

Lesson Notes:

The deca system is the number system we typically use. It runs off something called "Base 10". This means after 10 of each number's place, we increase the number of the next greatest magnitude. So we have 1, 10, 100, 1000, 10000, and so on.

If we add 1 to 9, the one's place gets reset to 0, and then the next magnitude is increased. So we are left with 10. If we have 99 and increase by 1, the 1's spot is reset to 0, the 10's spot is increased by one. There is already a 9 here, so it gets reset to 0 as well and then increments the next spot by 1. So we are left with 100. This makes it easy to do math, as each spot just adds an additional zero.

It may help to think of the numbers with leading 0's. So 00000998. If we add 1, it's now 00000999. Add one more, and we have to increment each 9. They get reset, and then the zero to the left is set to 1, 00001000.

The binary system however is different than this. Instead of having 10 numbers it can use, it only has 2. It has a 0 and a 1. This is due to how computers work at a basic level. They are only capable of reading whether a switch is on, or if it is off. They can't reliably read anything in between. This means all the math a computer does must be based off of this "on" (1) and this "off" (0).

A CPU processor has millions of these little switches all working in unison, combining and reorganizing the 1's and 0's to perform operations.

This binary system means that the number system works off of "Base 2" instead of "Base 10". This means it goes up in order of 2's.

Let's compare the deca and binary system with some tables. Below we have the number 1,578,483 broken out into it's magnitudes (number's places).

1	5	7	8	4	8	3
10^6	10^5	10^4	10^3	10^2	10^1	10^0
1,000,000	100,000	10,000	1,000	100	10	1

Each new position is an order of 10. So the 100's position is 10^2, the 1,000's position is 10^3 and so on. Since it is the deca system, we just move up by an order of 10 each time we run out of space in the current number. To get our final number, we just add up the top number multiplied by it's spot.

So in this situation we have 1,000,000 + 500,000 + 70,000 + 8,000 + 400 + 80 + 3 = 1,578,483

The Binary system however works a little bit differently.

1	0	1	1	1	1	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1

The number above is 1011111, which is the binary representation of the deca number 95. Notice the differences between the first table and this table. Instead of 10^x, it is 2^x. This makes for a much smaller transition between each number. Where the 6th place represents 1,000,000 in the deca system, it only represents 64 in the binary system.

We can get the number above from just adding like with the previous table. So we have 64+0+16+8+4+2+1 = 95.

Converting Binary to Decimal

To convert from a binary number back in to a deca number, all you have to do is what we just did above. Just add up the magnitudes with a 1 in them. In this case it would 64 +0 +16+8+4+2+1 = 95. The number 1011 would be 8 + 0 + 2 + 1 = 11.

If you don't want to draw a table. You can just add the magnitudes into the equation. So 1011 would be:

(1*(2^3)) + (0*(2^2)) + (1*(2^1)) + (1*(2^0)) => (1 * 8) + (0 *4) + (1 *2) + (1 *1) => 8 + 0 + 2 + 1 = 11

Converting Decimal to Binary

Converting from a deca number to a binary number is a little tricky, but not too hard. To do this, you just subtract the largest number you can from the deca number, and add that to your binary number. It works a little like this.

Deca Number = **55**

First we look for the column that will take out the most from our number without going over. In this case it is the 32. We put a one in this column and subtract this number from our original.

55 – 32 = **23**

	1					
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1

We then repeat this step over and over until our number hits zero. Any numbers that don't have a 1, get filled in as a 0.

23 – 16 = **7**

	1	1				
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1

Note our largest number here isn't the 8's spot. That would go over our 7, so we go to the next spot of 4.

7 – 4 = **3**

	1	1	0	1		
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1

3 – 2 = **1**

	1	1	0	1	1	
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1

1 – 1 **= 0**

	1	1	0	1	1	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1

So our final number comes out to be 55 = 110111.