1 Binary and Hexadecimal Numbers

To understand binary and hexadecimal numbers it's useful to begin with a quick refresher on the standard decimal system. The decimal number system is a *positional*, base-10 system. This means that the value of a digit depends on it's place (position) in the number and the places correspond to multiples of 10. Table 1 illustrates the idea, digits go in the 1's place, 10's place, 100's place, etc. So, the number 243 is 2*100 + 4*10 + 3*1.

Although in elementary school we're taught about the one's place, the ten's place, the hundred's place, etc., the places actually correspond to powers of 10. The 1's place is 10^0, the 10's place is 10^1, the 100's place is 10^2, etc.[1]

A final important fact is that the base of a number system determines how many digits are necessary. In a base 10 system there must be 10 digits: 0 through 9, to keep track of 0 through 9 objects. For more than 9 objects the other places are used.

The binary system (or base 2 system) is used in computer science because at the lowest level computers store two digits: 0 and 1, which correspond to off and on.[2] With only two digits available the first three numbers (starting from zero) must be: 0, 1, 10 because there's no digit to represent a 2. [3] Thus, the second place in the binary system is the 2's place (or more properly the 2^1 place), the third place is the 2^2 or 4's place, etc. Table 2 illustrates the binary system.

The binary system has two problems. The numbers end up being long and difficult for people to use (a "simple" number like 165 in decimal is 10100101 in binary) and converting back and forth between binary and decimal is awkward because the places don't line up. A decimal 7 requires three binary digits: 111, but a decimal 9 requires four binary digits: 1001.

To avoid these problems computer scientists often use a hexadecimal (base 16) system. In a base 16 system the places are 16^0 (1's place), 16^1 (16's place), 16^2 (256's place), etc. (see Table 3). This introduces a digit problem. Consider counting in hexadecimal:
0, 1, 2, 3, ..., 9, what come next in hexadecimal?

[1]Although there is no definitive proof, it is widely assumed that the base 10 system developed because people have 10 fingers. Originally people counted in sets of 10, e.g. 3 sets of 10 fingers, plus 4 more, which became 3*10 + 4 or 34.

[2]It is possible to design a computer that uses different voltages for different values, but difficulties in getting the necessary precision makes the binary system more practical.

[3]This also leads to the joke: "There are 10 types of people in the world, those who understand binary and those who don't.".

10^3	10^2	10^1	10^0	
1000's	100's	10's	1's	
8	2	4	3	8*1000 + 2*100 + 4*10 + 3*1 = 8243
			1	1*1 = 1
			...	...
			9	9*1 = 9
		1	0	1*10 = 10

Table 1: The decimal system. The places (columns) are multiples of 10. In a base 10 system 10 digits: 0 through 9 are required, after 9 the other places are used and no more digits are necessary.

2^3	2^2	2^1	2^0	
8's	4's	2's	1's	
1	1	0	1	1*8 + 1*4 + 0*2 + 1*1 = 13
			0	0*1 = 0
			1	1*1 = 1
		1	0	1*2 + 0*1 = 2
		1	1	1*2 + 1*1 = 3
	1	0	0	1*4 + 0*2 + 0*1 = 4

Table 2: The binary system. The places (columns) are multiples of 2. Only the digits 0 and 1 are used.

16^2 256's	16^1 16's	16^0 1's	
2	4	3	2*256 + 4*16 + 3*1 = 579
		1	1*1 = 1
		...	...
		9	9*1 = 9
		a	10*1 = 10
		...	...
		f	15*1 = 15
	1	0	1*16 + 0*1 = 16
	3	b	3*16 + 11*1 = 59

Table 3: The hexadecimal system. The places (columns) are multiples of 16. Sixteen digits are needed: 0 through 15 (decimal). The characters *a* through *f* are used for the numbers 10 through 15.

binary	1011 1100 1010 1110	1011	1100	1010	1110
hexadecimal	b 9 a e	b	9	a	e
decimal		11	9	10	14

Table 4: Conversion between binary and hexadecimal is relatively easy because four binary digits correspond to one hexadecimal digit.

It can't be 10 because in hexadecimal the number 10 represents 1*16 + 0*1 = 16 objects. The only solution is to introduce new digits for 10 (decimal) through 15 (decimal). Rather than introducing brand new symbols computer scientists simply borrowed letters. So, 10 (decimal) is written as *a* in hexadecimal, 11 (decimal) is written as *b* in hexadecimal, up to 15 (decimal) written as *f* in hexadecimal, which brings us to 16 (decimal) written as 10 in hexadecimal.

It's fairly easy to get decimal and hexadecimal numbers confused: is 23 equal to 2*10 + 3*1 (decimal) or 2*16 + 3*1 (hexadecimal)? To solve this problem C++ puts a *oX* in front of hexadecimal numbers when they are printed. This shows up whenever a memory address, which are always stored as hexadecimal numbers, is printed.

Hexadecimal numbers have two advantages. First, they are fairly compact and, with practice, not too hard for humans to use. Second, they are easily converted to and from binary, because four binary places correspond to one hexadecimal place (see Table 4).