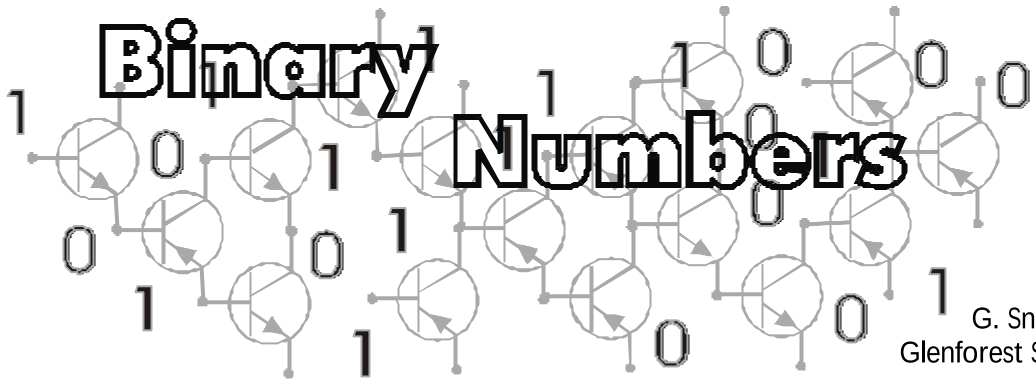


G. Snider
Glenforest S. S.

Numbering Systems

Although they are capable of many amazing feats, due to the ever increasing speed with which they can perform mathematical calculations, computers are fundamentally capable of understanding only two things: a high voltage and a low voltage. Consequently, computers are built to utilize a numbering system that uses only two numbers: binary, or base 2.

The numbering system which we work with on a daily basis, the decimal system, uses ten numerals (0 to 9), but it has something in common with the binary system: both are positional numbering systems. The decimal system is based on powers of 10, while the binary system is based on powers of 2.

If you are counting using the decimal system, we record the result using the numerals from 0 to 9. To save everyone the trouble of having to memorize more than ten numerals, we use a second digit to the left of the first, which has a weight ten times greater than the first digit: after 9 comes 10. The binary system works the same way, but since we run out of numerals very quickly, many more digits are needed to count in binary.

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Notice the pattern that develops as you go down each column!

The position of a digit in a decimal number determines its value.

Powers of 10	10^6	10^5	10^4	10^3	10^2	10^1	10^0
Positional Weight	1000000	100000	10000	1000	100	10	1
Example	6	5	4	3	2	1	0

The example number above can be written as:

```
6 x 1000000  =  6000000
 5 x 100000  =   500000
  4 x 10000  =    40000
   3 x 1000  =     3000
    2 x 100  =      200
     1 x 10  =       10
      0 x 0  = ________0
                6543210
```

The binary system works in similar fashion.

Powers of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Positional Weight	128	64	32	16	8	4	2	1
Example	1	0	1	0	0	1	0	1

A binary number can easily be converted to its decimal equivalent when we recognize that the value of each digit is determined by its position in the number. Taking the example above:

```
1 x 128  =  128
 0 x 64  =    0
 1 x 32  =   32
 0 x 16  =    0
  0 x 8  =    0
  1 x 4  =    4
  0 x 2  =    0
  1 x 1  =____1
            165
```

In practice, you can skip the columns with zeros and simply add the value for each column that has a 1 in it. For example:

1010 0101 = 128 + 32 + 4 + 1 = 165

To convert a decimal number to binary is a little more complex. Any decimal number can be written in binary as the sum of powers of two. Therefore, we can use the same table to make the conversion.

Powers of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Positional Weight	128	64	32	16	8	4	2	1
151 =								

For example, if you wish to convert the decimal number 151 to binary, look first for the largest power of two that is <u>less than</u> or <u>equal to</u> 151, which would be 128. Put a 1 in that column.

Powers of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Positional Weight	128	64	32	16	8	4	2	1
151 =	1							

Next, subtract 128 from 151, which leaves a remainder of 23. We look at each power of 2, and if it is more than the remainder, we put a 0 in that column, but if it is equal to or less than the remainder, we write a 1.

Powers of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Positional Weight	128	64	32	16	8	4	2	1
Example	1	0	0	1				

Then subtract 16 from the previous remainder (23 – 16 = 7) and repeat these steps until there is no remainder.

Powers of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Positional Weight	128	64	32	16	8	4	2	1
Example	1	0	0	1	0	1	1	1

You can check your result by adding up the values for each column which has a one in it: 1001 0111 = 128 + 16 + 4 + 2 + 1 = 151

Bits and Bytes

In the binary system, a 0 or 1—a binary digit—is called a bit.

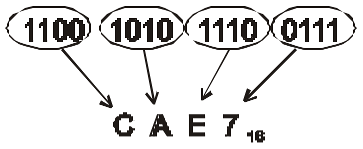
8 bits	=	1 byte
2^{10} bits	=	1024 bits = 1 kilobit (1Kb)
2^{20} bits	=	1 048 576 bits = 1 Megabit (1Mb)
2^{10} bytes	=	1kilobyte (1KB)
2^{20} bytes	=	1 Megabyte (1MB)
2^{30} bytes	=	1 Gigabyte (1GB)
2^{40} bytes	=	1 Terabyte (1TB)

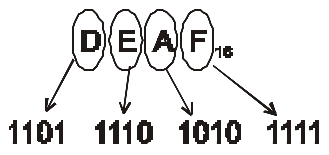
Hexadecimal

Computers routinely handle binary codes of 32 bits or more, but numbers that long are unwieldy to work with. Consequently it is common practice to use a numbering system known as hexadecimal, or base 16, simply because it is easy to convert from binary to hex and back. The hexadecimal system requires 16 different numerals, so the first six letters of the alphabet are used for the decimal numbers 10 to 15. As you can see on the table, there is a simple correspondence between 4-bit binary numbers and hexadecimal numbers.

Decimal	Binary	Hexadecimal	Decimal	Binary	Hexadecimal
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

To convert a binary number to hex, simply take one group of four bits at a time and write the hexadecimal equivalent.





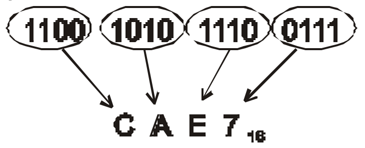
Hex to binary conversions work the same way in reverse.

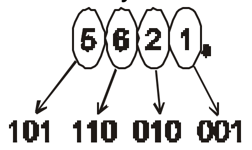
Octal

Since 2 cubed is eight, the base 8 numbering system, or octal, can also be used as a form of shorthand for binary numbers. The numbers from 0 to 7 can be written in binary using three bits.

Decimal	Binary	Octal	Decimal	Binary	Octal
0	000	0	4	100	4
1	001	1	5	101	5
2	010	2	6	110	6
3	011	3	7	111	7

These are the same numerals used in octal, so the conversion from binary to octal can be performed in much the same way as the conversion from binary to hexadecimal.





Octal to binary conversions can be done by reversing the process, as shown at right.

Applications of Binary Numbers

If you work with computers you will often hear phrases like “64-bit processor”, “32-bit operating system”, “8-bit colour”, or “16-bit sound”. In each case, the size (number of bits) of the binary number determines the range of possibilities for the task being performed. For example, early computer systems were often designed to display only 16 colours on the screen. This was because memory was expensive and very limited, so only 4-bits of memory could be allocated to each pixel on the screen. With four bits, you can write the numbers from 0 to 15, or define 16 different colours. If the colour palette is increased to 5 bits, however, the number of possible colours increases to 32, because the number of different combinations of 0s and 1s doubles each time you add another bit.

In general terms, the number of colours, memory addresses, characters, or other quantities which can be defined by a binary number of ‘n’ bits (where ‘n’ is a variable indicating the number of bits) is 2^n. So, an Intel 8088 microprocessor, with a 20-bit address bus, could address 1 048 576 bytes (1 Megabyte) of RAM. The 80286 processor, with a 24-bit address bus, could address 2^{24} bytes, or 16 Megabytes of RAM. The “High Color” setting in Windows, uses 2 bytes (16 bits) of memory for the colour of each pixel, providing 65 536 (2^{16}) different colours.

Binary Codes

As noted above, it is not only numerical quantities that can be represented using binary. All the characters and control codes on the standard computer keyboard are transmitted from the keyboard to the central processor as binary codes. There are at least two standards for encoding keyboard characters: ASCII (American Standard Code for Information Interchange) which is used in DOS and .txt files; and the ANSI (American National Standards Institute) code which Microsoft adopted for Windows. The Gray Code is a binary code used for sending positional from robotic arms and other automated machinery to the controller. Musical instrumentation, colours, graphics, and audio are examples of other concepts which can be represented by binary codes.