Binary Numbers

Harpeth Hall Winterim 2017

Medical Robotics

**Decimal Numbers**

When we write decimal numbers using Arabic numerals (as opposed to Roman numerals), we are using a “base 10” system. Each digit is multiplied by an appropriate power of 10 depending on its position in the number:

For example: 843

= 8 x 102 + 4 x 101 + 3 x 100

= 8 x 100 + 4 x 10 + 3 x 1

= 800 + 40 + 3

For whole numbers, the rightmost digit position is the one’s position (100 = 1). The number in that position indicates how many ones are present in the number. The next position to the left is ten’s, then hundred’s, thousand’s, and so on. Each digit position has a weight that is ten times the weight of the position to its right. In the decimal number system, there are ten possible values that can appear in each digit position, and so there are ten numerals required to represent the quantity in each digit position.

The decimal numerals are the familiar zero through nine (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Because of this, we call our system the base ten system.

**Binary Numbers**

The binary number system is similar, but in this case, the base is not ten, but is instead two. Each digit position in a binary number represents a power of two. So, when we write a binary number, each binary digit is multiplied by an appropriate power of 2 based on the position in the number:

For example: 101101

= 1 x 25 + 0 x 24 + 1 x 23 + 1 x 22 + 0 x 21 + 1 x 20

= 1 x 32 + 0 x 16 + 1 x 8 + 1 x 4 + 0 x 2 + 1 x 1

= 32 + 8 + 4 + 1

= 45

In the binary number system, there are only two possible values that can appear in each digit position rather than the ten that can appear in a decimal number. Only the numerals 0 and 1 are used in binary numbers. The term ‘bit’ is a contraction of the words ‘binary’ and ‘digit’, and when talking about binary numbers the terms bit and digit can be used interchangeably. When talking about binary numbers, it is often necessary to talk of the number of bits used to store or represent the number. This merely describes the number of binary digits that would be required to write the number. The number in the above example is a 6 bit number.

**Bits, Bytes, and Countable numbers**

In a computer, each digit is called a “bit”, and 8 of them put together make one “byte”. From these, you get measurements like kilobytes and gigabytes, which represent a large number of bytes.

One byte, which is 8 bits can represent values from 00000000 to 11111111, which is 0 to 127. For a string of x bits, you can represent numbers from 0 to 2x-1. So for 8, that is 28-1=128-1=127. This holds true for any number, for one bit you can count to 21-1=1, for 2 you can count to 22-1=3, etc.

**Converting from binary to decimal**

Converting a number from binary to decimal is quite easy. All that is required is to find the decimal value of each binary digit position containing a 1 and add them up.

For example: convert 10110 to decimal

1 x 21 = 2

1 x 22 = 4

1 x 24 = 16

2+4+16=22

**Converting from decimal to binary**

The most straightforward way is by subtraction. For example, take the number 327 and remember the powers of 2:

512 256 128 64 32 16 8 4 2 1

327-256=71

71-64=7

7-4=3

3-2=1

1-1=0

Therefore, 327=1+2+4+8+64+256

327=20+21+22+26+28

327=101000111

But there’s a more elegant way to do the same thing using division

327/2=161 remainder 1

163/2=81 remainder 1

81/2 = 40 remainder 1

40/2=20 remainder 0

20/2=10 remainder 0

10/2=5 remainder 0

5/2=2 remainder 1

2/2=1 remainder 0

1/2=0 remainder 1

So we put this together and get the same answer: 101000111

Cool!

**What else can we do with binary?**

Everything we can do with other numbers!

We can do addition and subtraction:

111001

+11001

1010010

or

1010010

- 111001

011001

We can verify that what that means, 57+25=82 and 82-57=25 is also true.

Multiplication and division also work as they would in “normal” numbers.

**But what does this have to do with computers and this class?**

Computers “count” using binary. The short explanation is that they store numbers in a series of transistors that act like switches that are either “on” or “off”, which is equivalent to a binary “1” and “0”.

The main takeaway for this class is when we are doing conversions between numbers in a computer and number in the physical world. We will call these digital and analog numbers, respectively. Digital numbers are represented using binary, whereas analog numbers are real numbers with decimals of infinite precision. Analog numbers are measured in volts, whereas digital numbers are represented in counts.

The conversion between digital and analog happens using a Digital to Analog Converter (DAC) or an Analog to Digital Converter (ADC). Both have a specific range that they work in: for example, you can have a 5 volt or a 10 volt converter. They also have a certain amount of precision, measured in bits. For example, there can be an 8-bit or 10-bit or 12-bit converter. This is how many counts are used to represent the range of voltages for any particular converter.

As an example, an 8-bit ADC with a range of 5 volts would take as an input a signal between 0 and 5 volts and convert it into a digital binary output to a computer as a number between 0 and 127. Remember that 127=28-1, which is how many numbers you can count to with 8 bits.

So now we have a relationship where 5 volts is equal to 127 counts, and 0 volts is equal to 0 counts. There is a linear relationship in between, so we can use ratios to find out what a given volt/count represents.

Example: a signal of 2.5 volts is converted to what number by 5 volt, 8 bit ADC?

2.5 volts/5volts = x counts/127 counts. x =127/2= 63.5, or by rounding 64 counts.

Using an 8-bit DAC with maximum output of 5 Volts, what voltage does 89 correspond to?

89 counts/127 counts = x volts/5 volts. x = 3.504 volts.

**What else?**

There are other things that binary can do: represent fractions/decimals or negative numbers, encode words/letters, etc. You’ll have to read more on your own time to find out!