Sergio A. Alvarez
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609, USA

web: http://www.cs.wpi.edu/∼alvarez/
e-mail: alvarez@cs.wpi.edu
phone: (508) 831-5118
fax: (508) 831-5776

Physical Data Representation

At the hardware level, computers represent data in terms of physical quantities, typically voltage or current signals. Since such signals may be modeled fairly accurately as being real-valued (capable of assuming any real number value in some continuous reference interval, to arbitrarily many decimal digits of accuracy), one could think of using such continuous-valued signals directly to represent numerical variables. For example, the number 3.79532 would be represented by a voltage difference of 3.79532 Volts between pins 12 and 17 of a certain circuit board. Computers that use such *analog* representations have actually been built for specialized purposes.

Why Digital?

Computers that use analog representations exclusively are extremely rare. This is partly because it is very difficult to reliably control physical signals to the desired arbitrarily high degree of accuracy. Many factors, including component aging, ambient temperature, and power source variation, contribute to changes in physical device characteristics that make their behavior vary over time.

Most computers rely mainly on *digital* data representations, in which fixed-length strings of symbols taken from a finite *alphabet* are used to represent numbers, characters, multimedia objects, etc. The symbols themselves are represented by values of physical variables, but the values need not be arbitrarily precise; they merely need to be close enough. For example, a fixed voltage range of 0 to 5 Volts may be split into 10 intervals of equal length, each consisting of allowable values for one of 10 possible symbols: any value between 0 and .5 Volts represents '0', any value greater than .5 but no greater than 1 represents '1', any value in the range $1 - 1.5$ represents '2', and so on. Default values for the various symbols would be those given by the midpoints of the corresponding intervals.

Why Binary?

Digital representations are resistant to environmental variation and noise. A base value of .25, representing '0', will still be interpreted as '0' if it is corrupted by noise, as long as the noise magnitude is less than the .25 distance to the adjacent symbol range $(.5, 1.]$. For a given physical signal range, say 0 to 5 Volts, noise resistance increases as the number of symbols decreases; for 5 symbols instead of 10, for example, the representational interval lengths are twice as long (one full Volt instead of half a Volt). Thus, one would prefer to use the smallest possible number of symbols, in order to maximize robustness of the representation. The smallest useful number of symbols is two. This is the usual choice, known as *binary* data representation.

The basic object in a binary data representation is the *binary digit*, or *bit*. This is the name for a variable whose value can be either of the two symbols of the binary alphabet. Normally, the allowable binary symbols are taken to be 0 and 1, although other choices would work equally well, e.g. *true* and *false*. Whatever the choice, a binary data representation uses strings of bits to represent objects of various types. Representational precision increases as the string lengths increase. Each computer has some default string length, called the *word length*, such that representations typically involve strings containing a whole number of words. Common word lengths are 16 bits and 32 bits. The details of some ways in which binary words can represent numerical and character data are given in the following notes.