2.1.2 Words

Every computer has a *word size*, indicating the nominal size of integer and pointer data. Since a virtual address is encoded by such a word, the most important system parameter determined by the word size is the maximum size of the virtual address space. That is, for a machine with a w-bit word size, the virtual addresses can range from 0 to $2^w - 1$, giving the program access to at most 2^w bytes.

Most personal computers today have a 32-bit word size. This limits the virtual address space to 4 gigabytes (written 4 GB), that is, just over 4×10^9 bytes. Although this is ample space for most applications, we have reached the point where many large-scale scientific and database applications require larger amounts of storage. Consequently, high-end machines with 64-bit word sizes are becoming increasingly common as storage costs decrease. As hardware costs drop over time, even desktop and laptop machines will switch to 64-bit word sizes, and so we will consider the general case of a w-bit word size, as well as the special cases of $w = 32$ and $w = 64$.

2.1.3 Data Sizes

Computers and compilers support multiple data formats using different ways to encode data, such as integers and floating point, as well as different lengths. For example, many machines have instructions for manipulating single bytes, as well as integers represented as 2-, 4-, and 8-byte quantities. They also support floating-point numbers represented as 4- and 8-byte quantities.

The C language supports multiple data formats for both integer and floating-point data. The C data type `char` represents a single byte. Although the name "`char`" derives from the fact that it is used to store a single character in a text string, it can also be used to store integer values. The C data type `int` can also be prefixed by the qualifiers `short`, `long`, and recently `long long`, providing integer representations of various sizes. Figure 2.3 shows the number of bytes allocated

C declaration	32-bit	64-bit
`char`	1	1
`short int`	2	2
`int`	4	4
`long int`	4	8
`long long int`	8	8
`char *`	4	8
`float`	4	4
`double`	8	8

Figure 2.3 **Sizes (in bytes) of C numeric data types.** The number of bytes allocated varies with machine and compiler. This chart shows the values typical of 32-bit and 64-bit machines.

C data type	Minimum	Maximum
`char`	−128	127
`unsigned char`	0	255
`short` [`int`]	−32,768	32,767
`unsigned short` [`int`]	0	65,535
`int`	−2,147,483,648	2,147,483,647
`unsigned` [`int`]	0	4,294,967,295
`long` [`int`]	−2,147,483,648	2,147,483,647
`unsigned long` [`int`]	0	4,294,967,295
`long long` [`int`]	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
`unsigned long long` [`int`]	0	18,446,744,073,709,551,615

Figure 2.8 **Typical ranges for C integral data types on a 32-bit machine.** Text in square brackets is optional.

C data type	Minimum	Maximum
`char`	−128	127
`unsigned char`	0	255
`short` [`int`]	−32,768	32,767
`unsigned short` [`int`]	0	65,535
`int`	−2,147,483,648	2,147,483,647
`unsigned` [`int`]	0	4,294,967,295
`long` [`int`]	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
`unsigned long` [`int`]	0	18,446,744,073,709,551,615
`long long` [`int`]	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
`unsigned long long` [`int`]	0	18,446,744,073,709,551,615

Figure 2.9 **Typical ranges for C integral data types on a 64-bit machine.** Text in square brackets is optional.

negative, zero, and positive numbers. We will see later that they are strongly related both in their mathematical properties and their machine-level implementations. We also investigate the effect of expanding or shrinking an encoded integer to fit a representation with a different length.

2.2.1 Integral Data Types

C supports a variety of *integral* data types—ones that represent finite ranges of integers. These are shown in Figures 2.8 and 2.9, along with the ranges of values they can have for "typical" 32- and 64-bit machines. Each type can specify a size with keyword `char`, `short`, `long`, or `long long`, as well as an indication of whether the represented numbers are all nonnegative (declared as `unsigned`), or possibly