THE DEFINITIVE GUIDE TO

HOW COMPUTERS DO MATH

Featuring

The Virtual DIY Calculator

Clive "MAX" Maxfield
Alvin Brown

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

010101001010111010101100111010101100000000010 01011000 0101011000010100011000

How Computers Do Math

THE DEFINITIVE GUIDE TO

HOW COMPUTERS DO MATH

Featuring

The Virtual DIY Calculator

Clive "MAX" Maxfield
Alvin Brown

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data is available.

ISBN-13 978-0471-73278-5
ISBN-10 0-471-73278-8

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

To all our friends
who make the world
such a wonderful place!

Contents

Laboratories

Do You Speak Martian?

There are an abundance of books on computer architectures, computer logic, and computer mathematics, and most of these works discuss various techniques for representing and manipulating numbers inside computers. Sadly, however, it appears that the majority of these tomes are written by visitors from the planet Mars, whose keen understanding of higher mathematics is somewhat offset by their limited grasp of the English language.

"*Say it's not so!*" you cry, but the proof is irrefutable. When was the last time you waded through a book on computer mathematics without your brain overheating? Much like reading *Being and Nothingness* by the famous French philosopher Jean Paul Sartre, one could mull over many of these cryptic masterpieces until the end of time without gaining so much as the faintest clue as to what was in their authors' minds.

This is why we have been moved to write this modest attempt at introducing the basics of computer arithmetic in words we can all understand. An impossible task, some may say, but we dare to fly in the face of conventional wisdom. Now, read on. . . .

CHAPTER 0

WHY THIS BOOK IS SO COOL

The simplest schoolboy is now familiar with facts for which Archimedes would have sacrificed his life.

Ernest Renan (1823–1892) in *Souvenirs D'enfance et de Jeunesse* (1887)

In this chapter we will learn

- The sock color of choice for the discerning Viking warrior
- Why this book is so cool
- That there are jobs awaiting time-travelers
- Some "stuff" about calculators
- Why this chapter is numbered "0"

Fearsome Warriors or Slaves to Fashion?

Many of us are used to thinking of the Vikings as fearsome warriors (this isn't your Mother's math book!) who descended from the northlands and rampaged and pillaged across Europe. These formidable fighters are popularly believed to have laughed at danger and scoffed at the elements, so the recent archeological discovery that many Vikings wore red woolly socks is, to many of us, somewhat disconcerting. However, we digress. . .

This Book Is Cool Because . . .

This book really is cool because, together, we are going to discover all sorts of interesting snippets of knowledge, tidbits of data, and nuggets of trivia, all bundled together with a tremendous amount useful information as to how computers and calculators perform their magic; and it isn't going to make our heads hurt at all!

Experts (the authors' mothers in this case) agree that one of the best ways to learn something and remember it afterward is by means of hands-on experience, which you are poised to gain in huge dollops by using the virtual microcomputer/calculator that you'll find on the CD-ROM accompanying this book.

As one rocket scientist[1] who reviewed this manuscript told the authors: "The combination of this book and its associated virtual computer is fantastic! Experience over the last 50 years has shown me that there's only one way to truly understand how computers work, and that is to learn one computer and its instruction set, no matter how simple or primitive, from the ground up. Once you fully comprehend how that simple computer functions, you can easily extrapolate to more complex machines." However, once again, we digress. . .

Jobs Abound for Time-Travelers

Today, most of us are extremely familiar with using numbers to perform simple tasks such as addition, subtraction, multiplication, and division.

[1]Honestly. Huntsville, Alabama, U.S.A. (where the authors live) is known as the Space Capital of America, and it's difficult to take even a short strolling without bumping into at least one rocket scientist.

Due to the fact that we are so intimate with these concepts, we tend to forget the tremendous amounts of mental effort that have been expended by so many folks over the millennia to raise us to our present level of understanding.

In the days of yore, when few people even knew how to count beyond the number of fingers on their hands,[2] anyone who was capable of performing relatively rudimentary mathematical operations could easily achieve a position of power and standing in the community.

If you could predict an eclipse, for example, you were obviously someone to be reckoned with (especially if it actually came to pass). Similarly, if you were a warrior chieftain, it would be advantageous to know how many fighting men and women you had at your command, and the person who could provide you with this vital information would obviously rank high on your summer-solstice card list.[3]

So, we can all rest easy in our beds at night, secure in the knowledge that, should we ever be presented with the occasion to travel back through time, there would be numerous job opportunities awaiting our arrival. However (you guessed it), we digress. . .

Calculators Then and Now

Every now and then, strange and wonderful mechanisms from antiquity are discovered. In 1900, for example, a device of unknown purpose containing numerous gear wheels forming a sophisticated mechanism dating from 2200 BC was discovered in a shipwreck close to the tiny Greek island of Antikythera. This contraption, which is now known as the *Antikythera Mechanism* or the *Antikythera Calculator,* was created during the early years of the Hellenistic Period, a golden age during which science and art flourished in ancient Greece.

In many cases, objects like this prompt speculation that our antediluvian ancestors were the creators of complex mechanical calculators with which they could perform mathematical operations such as addition, subtraction, multiplication, and division. In reality, however, the concept of zero (0) as representing a true quantity in a place-value num-

[2]Sometimes, fingers and toes in the case of societies that employed vigesimal (base-20) numbering systems.
[3]You wouldn't have a Christmas card list, because Christmas cards weren't invented until 1843. (Try finding this tasty morsel of trivia in another computer book!)

Note **The invention of zero and negative numbers, the use of tally sticks and the abacus, the origin of logic machines, calculators, and computers; and a wealth of other topics are discussed in more detail in "The History of Calculators, Computers, and Other Stuff" provided on the CD-ROM accompanying this book. See Appendix D for more details on additional resources, including this history.**

ber system didn't appear until around 600 AD in India. Without the notion of zero in this context, it is really not possible to create a mechanical calculator in any form we would recognize.

This is not to say that these ancient mechanisms were not incredibly cunning and refined. However, such instruments were probably designed to measure things or to track time in one way or another; for example, to help in predicting the seasons and the activities of celestial objects like the sun, moon, planets, and constellations.

On the other hand, we might well wonder why, after the idea of zero as an actual number entered the scene, it took so long to actually invent a true mechanical calculator. Of course it's fair to say that Europe was undergoing a period of stagnation in art, literature, and science called the Dark Ages.[4] However, there were many brilliant minds in the Byzantine Empire and the Arabic, Chinese, Indian, and Persian cultures (to name but a few) that would almost certainly have been up to the task.

Be this as it may, the first true mechanical calculators of which we are aware were the Calculating Clock (1623), which was created by the German astronomer and mathematician Wilhelm Schickard (1592–1635); the Pascaline or Arithmetic Machine (1642), which was the inspiration of the French mathematician, physicist, and theologian Blaise Pascal (1623–1662); and the Step Reckoner (1694), which was the brainchild of a German Baron called Gottfried von Leibniz (1646–1716).

Countless mechanical calculating machines emerged to see the light of day over the next few hundred years, but these were all largely based on the underlying principles established by Schickard, Pascal, and Leibniz. However, "the times they were a'changin," as they say. The invention of the transistor in 1947 and the integrated circuit (silicon chip) in 1958 paved the way for an entirely new class of calculating devices.

[4]Some pundits equate the Dark Ages with the Middle Ages (the period in European history between Antiquity and the Renaissance, often dated from 476 AD to 1453 AD), whereas others regard the Dark Ages as encompassing only the early portion of the Middle Ages.

The first experimental model of an electronic pocket calculator was created by Texas Instruments in 1966.[5] This was followed in 1970/1971 by the first commercially available unit, a portable (hand-held) printing calculator called the Pocketronic. Created as a joint effort by Texas Instruments and Cannon, and priced at $150 (which was a *lot* of money at the time), this four-function device could add, subtract, multiply, and divide.

Today, of course, it's possible to purchase pocket calculators that boast enough computing power to guide a rocket to the moon, and cheap and cheerful versions are now so ubiquitous that it's not uncommon to find them as giveaways in boxes of breakfast cereal. However (believe it or not), we digress. . .

Déjà Vu (Didn't Someone Just Say That?)

Whether they are aware of it or not, the average person in the developed world comes into contact with dozens or hundreds of electronic calculating, computing, and controlling machines every day. These devices toil away, performing countless mathematical operations, but very few of us actually know what they are doing or, perhaps more importantly, how they are doing it.

Consider your own pocket calculator, for example. When you multiply two numbers like 36.984562 and 79.386431 together and you are presented with the result, you may well assume that the answer is correct, but just *how correct is it?* and *is it correct enough?* The accuracy and precision (as we will discover, these are two different things) required by an accountant and a rocket scientist may be poles apart.

As we noted earlier, in the not-so-distant past, anyone who was capable of performing relatively rudimentary mathematical operations could easily achieve a position of power and standing in the community. Well, we are in danger of finding ourselves in a déjà vu situation, because knowledge is power, and very few folks actually have any clue as to what is going on behind the scenes with regard to the way in which computers and calculators perform even simple operations.

[5]This machine is now preserved at the National Museum of American History (a part of the Smithsonian Institution) in Washington, DC, U.S.A.

But fear not, because there's nothing to fear but fear itself. You can turn that frown upside down into a smile, because we are going to explain just how computers and calculators perform their magic. The really "cool beans" part of all of this is the virtual microcomputer/calculator on the CD-ROM accompanying this book. When you first launch this DIY Calculator application on your home computer (PC) and click the buttons on the calculator interface nothing will happen. But wait, there's more! We are going to guide you through the process of creating your very own program to make the calculator function as required. On the way, we will discover all sorts of interesting things that will leave us grinning with delight and gasping for more. However, we digress. . .

Why Is this Chapter Numbered "0"?

Computer programmers and engineers typically start counting, indexing, and referencing things from zero.[6] Thus, in order to keep in the spirit of things, we decided to follow the same convention with our chapter numbers.

However, let us digress no more. You are poised on the brink of discovering all manner of weird and wonderful things, so proceed immediately to Chapter 1 and let the fun begin!

Note **Except where such interpretation is inconsistent with the context, the singular shall be deemed to include the plural, the masculine shall be deemed to include the feminine, and the spelling and punctuation shall be deemed to be correct!**

[6]The reasons for this—and related conventions—will become apparent as we wend our way through the topics in this book.

CHAPTER 1

INTRODUCING BINARY AND HEXADECIMAL NUMBERS

I am ill at these numbers.

William Shakespeare
(1564–1616) in *Hamlet* (1601)

In this chapter we will learn about:

- Counting on fingers and toes
- Place-value number systems
- Using powers or exponents
- The binary number system
- The hexadecimal number system
- Counting in the binary and hexadecimal systems
- Using wires to represent numbers

Why Do We Need to Know this Stuff?

The number system with which we are most familiar is the *decimal system,* which is based on ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. As we shall soon discover, however, it's easier for electronic systems to work with data that is represented using the *binary* number system, which comprises only two digits: 0 and 1.

Unfortunately, it's difficult for humans to visualize large values presented as strings of 0s and 1s. Thus, as an alternative, we often use the *hexadecimal* number system, which is based on sixteen digits that we represent by using the numbers 0 through 9 and the letters A through F.

Familiarity with the binary and hexadecimal number systems is necessary in order to truly understand how computers and calculators perform their magic. In this chapter, we will discover just enough to make us dangerous, and then we'll return to consider number systems and representations in more detail in Chapters 4, 5, and 6.

Counting on Fingers and Toes

The first tools used as aids to calculation were almost certainly man's own fingers. It is no coincidence, therefore, that the word "digit" is used to refer to a finger (or toe) as well as a numerical quantity. As the need grew to represent greater quantities, small stones or pebbles could be used to represent larger numbers than could fingers and toes. These had the added advantage of being able to store intermediate results for later use. Thus, it is also no coincidence that the word "calculate" is derived from the Latin word for pebble.

Throughout history, humans have experimented with a variety of different number systems. For example, you might use one of your thumbs to count the finger joints on the same hand (1, 2, 3 on the index finger; 4, 5, 6 on the next finger; up to 10, 11, 12 on the little finger). Based on this technique, some of our ancestors experimented with base-12 systems. This explains why we have special words like *dozen,* meaning "twelve," and *gross,* meaning "one hundred and forty-four" (12 × 12 = 144). The fact that we have 24 hours in a day (2 × 12) is also related to these base-12 systems.

Similarly, some groups used their fingers *and* toes for counting, so they ended up with base-20 systems. This is why we still have special

Note Truth to tell, the first people to employ their fingers as counting aids didn't use the digits 0, 1, 2, and 3. Instead, they started counting at one, as in 1, 2, 3, 4, because if all one is doing is counting goats, for example, then the concepts of "zero goats" (and "negative goats") are relatively unimportant. In fact, it was not until around 600 AD that the use of zero as an actual value, along with the concept of negative numbers, first appeared in India.

words like *score,* meaning "twenty." However, due to the fact that we have ten fingers, the number system with which we are most familiar is the decimal system, which is based on ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 (see note). The word *decimal* is derived from the Latin *decam,* meaning "ten." As this system uses ten digits, it is said to be *base-10* or *radix-10;* the term *radix* comes from the Latin word meaning "root."

Note Dates in Western books have traditionally been shown in the form 1000 BC or 2001 AD, where BC stands for "Before Christ" and AD is an abbreviation of the Latin *Ano Domini,* meaning "Year of our Lord." In many cases, modern historians now prefer to use BCE, meaning "Before Common Era," instead of BC; and CE, meaning "Common Era," instead of AD. This preference is based on the fact that BCE and CE aren't associated with any particular religion. However, the BC/AD nomenclature is more familiar to nonhistorians, so that is what we are using in this book.

Place-Value Number Systems

Consider the concept of Roman numerals, in which I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1,000, and so forth. Using this scheme, XXXV represents 35 (three tens and a five). One problem with this type of number system is that over time, as a civilization develops, it tends to become necessary to represent larger and larger quantities. This means that mathematicians either have to keep on inventing new symbols or start using lots and lots of their old ones. But the biggest disadvantage of this approach is that it's painfully difficult to work with (try multiplying CLXXX by DDCV and it won't take you long to discover what we mean).

An alternative technique is known as a *place-value system,* in which the value of a particular digit depends both on itself and on its position within the number. This is the way in which the decimal number system works. In this case, each column in the number has a "weight" associated with it, and the value of the number is determined by combining each digit with the weight of its column (Figure 1-1).

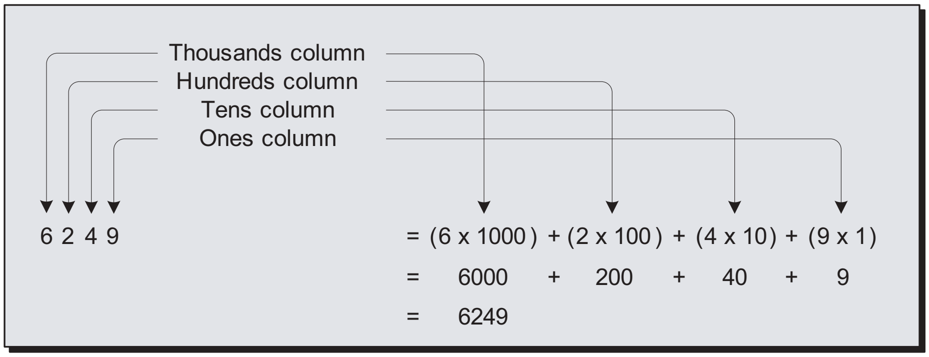

Figure 1-1. Combining digits and column weights in decimal.

Using Powers or Exponents

Another way of thinking about this is to use the concept of *powers;* for example, $100 = 10 \times 10$. This can be written as 10^2, meaning "ten to the power of two" or "ten multiplied by itself two times." Similarly, $1000 = 10 \times 10 \times 10 = 10^3$, $10{,}000 = 10 \times 10 \times 10 \times 10 = 10^4$, and so forth (Figure 1-2).

Rather than talking about using powers, some mathematicians prefer to refer to this type of representation as an *exponential form.* In the case of a number like 10^3, the number being multiplied (10) is known as the *base,* whereas the *exponent* (3) specifies how many times the base is to be multiplied by itself.

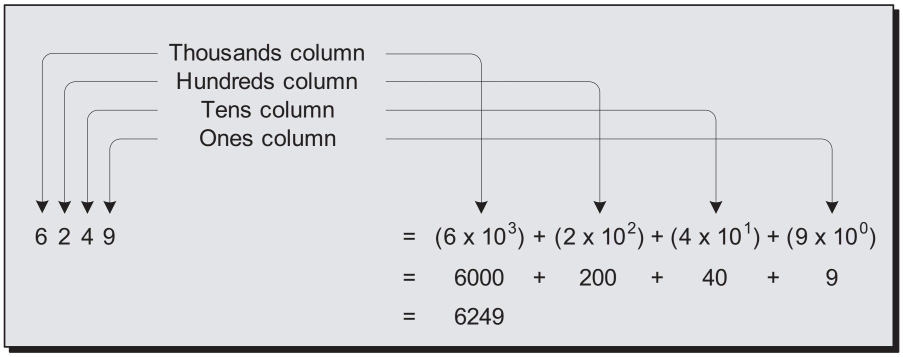

Figure 1-2. Using powers of ten.

There are a number of points associated with powers (or exponents) that are useful to remember:

- Any base raised to the power of 1 is the base itself, so $10^1 = 10$.
- Strictly speaking, a power of 0 is not really part of the series. By convention, however, any base to the power of 0 equals 1, so $10^0 = 1$.
- A value with an exponent of 2 is referred to as the *square* of the number; for example, $3^2 = 3 \times 3 = 9$, where 9 (or 3^2) is the square of 3.
- A value with an exponent of 3 is referred to as the *cube* of the number; for example, $3^3 = 3 \times 3 \times 3 = 27$, where 27 (or 3^3) is the cube of 3.
- The square of a *whole number* (0, 1, 2, 3, 4, etc.) is known as a *perfect square;* for example, $0^2 = 0$, $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $4^2 = 16$, $5^2 = 25$, and so on are perfect squares. (The concepts of whole numbers and their cousins are introduced in more detail in Chapter 4.)
- The cube of a *whole number* is known as a *perfect cube;* for example, $0^3 = 0$, $1^3 = 1$, $2^3 = 8$, $3^3 = 27$, $4^3 = 64$, $5^3 = 125$, and so on are perfect cubes.

But we digress. The key point here is that the column weights are actually powers of the number system's base. This will be of particular interest when we come to consider other systems.

Counting in Decimal

Counting in decimal is easy (mainly because we're so used to doing it). Commencing with 0, we increment the first column until we get to 9, at which point we've run out of available digits. Thus, on the next count we reset the first column to 0, increment the second column to 1, and continue on our way (Figure 1-3).

Similarly, once we've reached 99, the next count will set the first column to 0 and attempt to increment the second column. But the second column already contains a 9, so this will also be set to 0 and we'll increment the *third* column, resulting in 100, and so it goes.

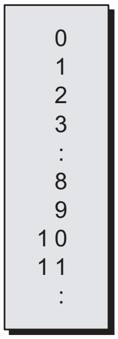

Figure 1-3. Counting in decimal.

The Binary Number System

Unfortunately, the decimal number system is not well suited to the internal workings of computers. In fact, for a variety of reasons that will become apparent as we progress through this book, it is preferable to use the binary (base-2) number system, which employs only two digits: 0 and 1.

Binary is a place value system, so each column in a binary number has a weight, which is a power of the number system's base. In this case we're dealing with a base of two, so the column weights will be $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, and so forth (Figure 1-4).

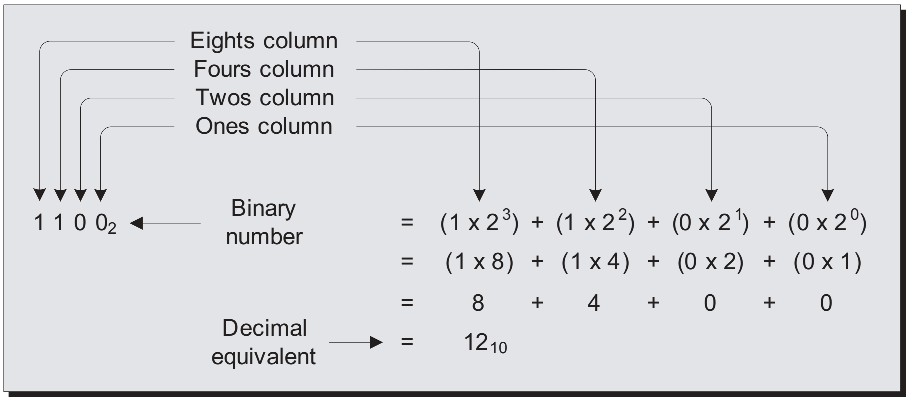
Eights column
Fours column
Twos column
Ones column
$1\,1\,0\,0_2$
Binary number
$= (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0)$
$= (1 \times 8) + (1 \times 4) + (0 \times 2) + (0 \times 1)$
$= 8 + 4 + 0 + 0$
Decimal equivalent
$= 12_{10}$

Figure 1-4. Combining digits and column weights in binary.

When working with number systems other than decimal, or working with a mixture of number systems as shown in Figure 1-4, it is common to use subscripts to indicate whichever base is in use at the time. For example, $1100_2 = 12_{10}$, which means $1100_{\text{Binary}} = 12_{\text{Decimal}}$.

Another common alternative is to prefix numbers with a character to indicate the base; for example, %1100, where the % indicates a binary value. (In fact there are a variety of such conventions, so you need to keep your eyes open and your wits about you as you plunge deeper into the mire.) Unless otherwise indicated, a number without a subscript or a prefix character is generally assumed to represent a decimal value.

Sometime in the late 1940s, the American chemist turned topologist turned statistician John Wilder Tukey realized that computers and the binary number system were destined to become increasingly important. In addition to coining the word "software," Tukey decided that saying "binary digit" was a bit of a mouthful, so he started to look for an alternative. He considered a variety of options, including *binit* and *bigit*, but eventually settled on *bit,* which is elegant in its simplicity and is used to this day.

Binary values of 1100_2 and 11001110_2 are said to be four and eight bits wide, respectively. Groupings of four bits are relatively common, so they are given the special name of *nybble* (or sometimes *nibble*). Similarly, groupings of eight bits are also common, so they are given the special name of *byte.* Thus, *"two nybbles make a byte,"* which goes to show that computer engineers do have a sense of humor (albeit not a tremendously sophisticated one).

Counting in binary

Counting in binary is even easier than counting in decimal; it just seems a little harder if you aren't familiar with it. As usual, we start counting from 0, and then we increment the first column to be 1, at which point we've run out of all the available digits. Thus, on the next count we reset the first column to 0, increment the second column to 1, and proceed on our merry way (Figure 1-5).

Similarly, once we've reached 11_2, the next count will set the first column to 0 and attempt to increment the second column. But the second column already contains a 1, so this will also be set to 0 and we'll increment the *third* column, resulting in 100_2. (Note that Figure 1-5 doesn't require us to use subscripts or special prefix symbols to indicate the binary values, because in this case they are obvious from the context.)

Binary	Decimal
0	0
1	1
1 0	2
1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7
1 0 0 0	8
:	:

Figure 1-5. Counting in binary.

Learning your binary "times tables"

Cast your mind back through the mists of time to those far-off days in elementary (junior) school. You probably remember learning your multiplication "tables" by rote, starting with the "two times table" ("one two is two, two twos are four, three twos are six, . . .") and wending your weary way onward and upward to the "twelve times table" ("ten twelves are one hundred and twenty, eleven twelves are one hundred and thirty-two, and—wait for it, wait for it—twelve twelves are one hundred and forty-four"). Phew!

The bad news is that we need to perform a similar exercise for binary: the good news is that, as illustrated in Figure 1-6, we can do the whole thing in about five seconds flat! That's all there is to it. This is the only binary multiplication table there is. This really is pretty much as complex as it gets!

Multiplication
0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Figure 1-6. The binary multiplication table.

Using Wires to Represent Numbers

Today's digital electronic computers are formed from large numbers of microscopic semiconductor switches called transistors, which can be turned on and off incredibly quickly (thousands of millions of times a second).

Note **Computers can be constructed using a variety of engineering disciplines, resulting in such beasts as electronic, hydraulic, pneumatic, and mechanical systems that process data using analog or digital techniques.**

For the purposes of this book however, we will consider only digital electronic implementations, because these account for the overwhelming majority of modern computer systems.

Transistors can be connected together to form a variety of primitive logical elements called *logic gates*. In turn, large numbers of logic gates can be connected together to form a computer.[1]

It's relatively easy for electronics engineers to create logic gates that can detect, process, and generate two distinct voltage levels.[2] For example, logic gates circa 1975 tended to use voltage levels of 0 volts and 5 volts. However, the actual voltage values used inside any particular computer are of interest only to the electronics engineers themselves. All we need know is that these two levels can be used to represent binary 0 and 1, respectively.

In the "counting in binary" example illustrated in Figure 1-5, there was an implication that our table could have continued forever. This is because numbers written with pencil and paper can be of any size, limited only by the length of your paper and your endurance. By comparison, the numbers inside a computer have to be mapped onto a physical system of logic gates and wires. For example, a single wire can be used to represent only $2^1 = 2$ binary values (0 and 1), two wires can be used to represent $2^2 = 2 \times 2 = 4$ different binary values (00, 01, 10, and 11), three wires can be used to represent $2^3 = 2 \times 2 \times 2 = 8$ differ-

[1]Transistors and logic gates are introduced in greater detail in the book *Bebop to the Boolean Boogie (An Unconventional Guide to Electronics),* 2nd edition, by Clive "Max" Maxfield ISBN: 0-7506-7543-8.
[2]Voltage refers to electrical potential, which is measured in *volts.* For example, a 9-volt battery has more electrical potential than a 3-volt battery. Once again, the concept of voltage is introduced in greater detail in *Bebop to the Boolean Boogie (An Unconventional Guide to Electronics).*

ent binary values (000_2, 001_2, 010_2, 011_2, 100_2, 101_2, 110_2, and 111_2), and so on.

The term *bus* is used to refer to a group of signals that carry similar information and perform a common function. In a computer, a bus called the *data bus* is, not surprisingly, used to convey data.

Note **In this context, the term "data" refers to numerical, logical, or other information presented in a form suitable for processing by a computer.For the purposes of this book however, we will consider only digital electronic implementations, because these account for the overwhelming majority of modern computer systems.**

Actually, "data" is the plural of the Latin *datum,* meaning "something given." The plural usage is still common, especially among scientists, so it's not unusual to see expressions like "*These data are . . .*"

However, it is becoming increasingly common to use "data" to refer to a singular group entity such as information. Thus, an expression like "*This data is . . .*" would also be acceptable to a modern audience.

For the purposes of these discussions, let's assume that we have a data bus comprising eight wires that we wish to use to convey binary data. In this case, we can use these wires to represent $2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$ different combinations of 0s and 1s. If we wished to use these different binary patterns to represent numbers, then we might decide to represent decimal values in the range 0 to 255 (Figure 1-7). For reasons that will be discussed in greater detail in Chapter 4, this form of representation is referred to as *unsigned binary numbers.*

In any numbering system, it is usual to write the most-significant digit on the left and the least-significant digit on the right. For example, when you see a decimal number such as 825, you immediately assume that the "8" on the left is the most-significant digit (representing eight hundred), whereas the "5" on the right is the least-significant digit (representing only five). Similarly, the most-significant binary digit, referred to as the *most-significant bit (MSB)*, is on the left-hand side of the binary number, whereas the *least-significant bit (LSB)* is on the right.

The Hexadecimal Number System

The problem with binary values is that, although we humans are generally good at making sense out of symbolic representations like words and numbers, we find it difficult to comprehend the meaning behind long strings of 0s and 1s. For example, the binary value 11001110_2 doesn't im-