tween the two approaches is minor in principle, but it has the effect of focusing the attention on one aspect or another in *every* formula in which the roundoff occurs.

Roundoff, once started, propagates through subsequent operations. For example, in the multiplication of two numbers

$$x_1(1 + \varepsilon_1)x_2(1 + \varepsilon_2) = x_1x_2(1 + \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2)$$

Usually $\varepsilon_1\varepsilon_2$ can be neglected, but we need to add the roundoff ε from the present operation to get the total roundoff

$$x_3(1 + \varepsilon_3) = x_1x_2(1 + \varepsilon_1 + \varepsilon_2 + \varepsilon) \qquad |\varepsilon| < \tfrac{1}{2} \times 10^{-2}$$
$$\varepsilon_3 = \varepsilon_1 + \varepsilon_2 + \varepsilon$$

Similarly for the other operations.

PROBLEMS

2.6.1 Calculate the mean and variance for chopping (see Sec. 2.4).

2.6.2 Examine the propagation of roundoff through division.

2.6.3 Show that for roundoff the third moment about the mean is 0 and the fourth is 1/80.

2.7 THE BINARY REPRESENTATION OF NUMBERS

The various features of the floating-point number system have been discussed in terms of the familiar decimal representation. However, most computing is done on machines that use the binary form for representing numbers. The binary representation system is increasingly familiar these days, and so only a few details will be given. As a general rule, if you have trouble with the binary system, then probably it is because you do not really understand the decimal system, and the way out of your trouble is to think about the similar situation in decimals. For example, a decimal number means

$$1{,}414.214 =$$
$$1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 4 \times 10^0 + 2 \times 10^{-1} + 1 \times 10^{-2} + 4 \times 10^{-3}$$

Similarly the binary number

$$1011.101 =$$
$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

It is important to recognize the difference between the *kind* of number system used (counting, fixed, or floating) and the *form* of the representation of the number

(binary, octal, decimal, hexadecimal, etc.). Regardless of the form of the representation, the number is the same; thus *3* in decimal and *11* in binary *are the same number.* It is conventional to speak of the binary number or the decimal number to save words, but in careful thinking it is necessary to differentiate between the number system and the form of the representation.

Perhaps the main stumbling block with the binary system is converting from decimal to binary and back. For example, consider writing the number 417 in the binary representation. That is, we wish to write 417 as a sum of powers of 2 with coefficients of either 0 or 1

$$417 = 1 \cdot 2^n + a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \cdots + a_0 2^0$$

If we divide both sides by 2, the remainder is a_0. If we divide the quotient by 2, the remainder is a_1, and so on.

$$\begin{array}{r|l} 2 & 417 \\ 2 & 208 + 1 = a_0 \\ 2 & 104 + 0 = a_1 \\ 2 & 52 + 0 = a_2 \\ 2 & 26 + 0 = a_3 \\ 2 & 13 + 0 = a_4 \\ 2 & 6 + 1 = a_5 \\ 2 & 3 + 0 = a_6 \\ 2 & 1 + 1 = a_7 \\ 2 & 0 + 1 = a_8 \end{array}$$

$$417 = 110\ \ 100\ \ 001$$

To convert back we reverse the process. Multiply a_8 by 2 and add a_7; multiply the sum by 2 and add a_6; etc.

$$\begin{array}{l} 1 \\ 2 \\ \overline{2 + 1 = 3} \\ \qquad \times 2 \\ \qquad\ \overline{6 + 0 = 6} \\ \qquad\qquad \times 2 \\ \qquad\qquad\ \overline{12 + 1 = 13} \\ \qquad\qquad\qquad \times 2 \\ \qquad\qquad\qquad\ \overline{26 + 0 = 26} \\ \qquad\qquad\qquad\qquad\qquad \text{etc.} \end{array}$$

The above process works for integers. For the fractional part we use a similar trick of doubling and using the overflow on the left. For example,

$$.762 = a_{-1}2^{-1} + a_{-2}2^{-2} + a_{-3}2^{-3} + \ldots$$

doubie

$$1.524 = a_{-1} + a_{-2}2^{-1} + a_{-3}2^{-2} + \ldots$$

and so $a_{-1} = 1$.

In full

$$\begin{array}{r|l} & .762 \\ & \underline{\quad 2} \\ 1 & 524 \\ & \underline{\quad 2} \\ 1 & 048 \\ & \underline{\quad 2} \\ 0 & 096 \\ & \underline{\quad 2} \\ 0 & 192 \\ & \underline{\quad 2} \\ 0 & 384 \\ & \underline{\quad 2} \\ 0 & 768 \\ & \underline{\quad 2} \\ 1 & 536 \end{array}$$

Hence

$$.762 = .110\,000\,1\ldots$$

Reversing the process gets from the binary representation to the decimal.

Previously prepared tables for conversion purposes are another way of converting from one form of representing a given number to another form of representing the same number.

The conversions are customarily done by the computing machine, and since the machine works in binary arithmetic and since the above discussion has used decimal arithmetic, there are differences in exactly what happens in the machine. It is not hard to take the machine's point of view and to deduce how to convert its way.