

### 3-4 Floating-Point Representation

*mantissa*

*exponent*

The floating-point representation of a number has two parts. The first part represents a signed, fixed-point number called the mantissa. The second part designates the position of the decimal (or binary) point and is called the exponent. The fixed-point mantissa may be a fraction or an integer. For example, the decimal number +6132.789 is represented in floating-point with a fraction and an exponent as follows:

<i>Fraction</i>	<i>Exponent</i>
+0.6132789	+04

The value of the exponent indicates that the actual position of the decimal point is four positions to the right of the indicated decimal point in the fraction. This representation is equivalent to the scientific notation  $+0.6132789 \times 10^{+4}$ .

Floating-point is always interpreted to represent a number in the following form:

$$m \times r^e$$

Only the mantissa  $m$  and the exponent  $e$  are physically represented in the register (including their signs). The radix  $r$  and the radix-point position of the mantissa are always assumed. The circuits that manipulate the floating-point numbers in registers conform with these two assumptions in order to provide the correct computational results.

A floating-point binary number is represented in a similar manner except that it uses base 2 for the exponent. For example, the binary number +1001.11 is represented with an 8-bit fraction and 6-bit exponent as follows:

<i>Fraction</i>	<i>Exponent</i>
01001110	000100

*fraction*

The fraction has a 0 in the leftmost position to denote positive. The binary point of the fraction follows the sign bit but is not shown in the register. The exponent has the equivalent binary number +4. The floating-point number is equivalent to

$$m \times 2^e = +(.1001110)_2 \times 2^{+4}$$

*normalization*

A floating-point number is said to be *normalized* if the most significant digit of the mantissa is nonzero. For example, the decimal number 350 is normalized but 00035 is not. Regardless of where the position of the radix point is assumed to be in the mantissa, the number is normalized only if its leftmost digit is nonzero. For example, the 8-bit binary number 00011010 is not normal-

ized because of the three leading 0's. The number can be normalized by shifting it three positions to the left and discarding the leading 0's to obtain 11010000. The three shifts multiply the number by  $2^3 = 8$ . To keep the same value for the floating-point number, the exponent must be subtracted by 3. Normalized numbers provide the maximum possible precision for the floating-point number. A zero cannot be normalized because it does not have a nonzero digit. It is usually represented in floating-point by all 0's in the mantissa and exponent.

Arithmetic operations with floating-point numbers are more complicated than arithmetic operations with fixed-point numbers and their execution takes longer and requires more complex hardware. However, floating-point representation is a must for scientific computations because of the scaling problems involved with fixed-point computations. Many computers and all electronic calculators have the built-in capability of performing floating-point arithmetic operations. Computers that do not have hardware for floating-point computations have a set of subroutines to help the user program scientific problems with floating-point numbers. Arithmetic operations with floating-point numbers are discussed in Sec. 10-5.