TABLE 7.5 IEEE/Industry Floating-Point Formats

Format	Total Bits	Exponent Bits	Exponent Bias	Smallest Exponent	Largest Exponent	Significant Bits	Mantissa MSB
Single precision	32	8	127	–126	+127	23	Hidden
Double precision	64	11	1,023	–1022	+1023	52	Hidden
Extended precision	80	15	16,383	–16382	+16383	64	Explicit
Quadruple precision	128	15	16,383	–16382	+16383	112	Hidden

ing the single-precision format, the exponent field is calculated by adding the true exponent value to the bias, 127, to get a final value of 131. Expressing these fields in a 32-bit word yields the floating point value 0x41CA0000 as shown in Fig. 7.10.

Note that the sign bit is 0 and that the mantissa's MSB has been omitted. This example is convenient, because the binary representation of 25.25 is finite. However, certain numbers that have finite representations in decimal cannot be represented as cleanly in binary, and vice versa. The number 0.23 clearly has a finite decimal representation but, when converted to binary, it must be truncated at whatever precision limitation is imposed by the floating-point format in use. The number 0.23 can be converted to a binary fraction by factoring out successive negative powers of 2 and expressing the result with 24 significant figures (leading 0s do not count), because the single precision format supports a 24-bit mantissa,

0.0011_1010_1110_0001_0100_0111_11

This fraction is then converted to a mantissa and power-of-two representation,

$$1.1101\_0111\_0000\_1010\_0011\_111 \times 2^{-3}$$

A single-precision floating-point exponent value is obtained by adding the bias, 127+(–3) = 124, for a final representation of

0011_1110_0110_1011_1000_0101_0001_1111 (0x3E6B851F)

These conversions are shown only to explain the IEEE formats and almost never need to be done by hand. Floating-point processing is performed either by dedicated hardware or software algorithms. Most modern high-performance microprocessors contain on-chip floating-point units (FPUs), and their performance is measured in floating-point operations per second (FLOPS). High-end microprocessors can deliver several gigaFLOPS (GFLOPS) of throughput on benchmark tests. Computers without hardware FPUs must emulate floating-point processing in software, which can be a relatively slow process. However, if a computer needs to perform only a few hundred floating-point operations per second, it may be worth saving the cost and space of a dedicated hardware FPU.

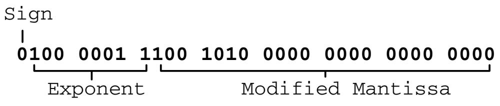

FIGURE 7.10 Single-precision floating-point expression of 25.25.

As can be readily observed from Table 7.5, very large and very small numbers can be represented because of the wide ranges of exponents provided in the various formats. However, the representation of 0 seems rather elusive with the requirement that the mantissa always have a leading 1. Values including 0 and infinity are represented by using the two-exponent values that are not supported for normal numbers: 0 and $2^n - 1$. In the case of the single-precision format, these values are 0x00 and 0xFF.

An exponent field of 0x00 is used to represent numbers of very small magnitude, where the interpreted exponent value is fixed at the minimum for that format: –126 for single precision. With a 0 exponent field, the mantissa's definition changes to a number greater than or equal to 0 and less than 1. Smaller numbers can now be represented, though with decreasing significant figures, because magnitude is now partially represented by the significand field. For example, 101×2^{-130} is expressed as 0.0101×2^{-126}. Such special-case numbers are *denormalized,* because their mantissas defy the *normalized* form of being greater than or equal to 1 and less than 2. Zero can now be expressed by setting the significand to 0 with the result that $0 \times 2^{-126} = 0$. The presence of the sign bit produces two representations of zero, positive and negative, that are mathematically identical.

Setting the exponent field to 0xFF (in single precision) is used to represent either infinity or an undefined value. Positive and negative infinity are represented by setting the significand field to 0 and using the appropriate sign bit. When the exponent field is 0xFF and the significand field is nonzero, the representation is "not a number," or *NaN*. Examples of computations that may return NaN are $0 \div 0$ and $\infty \div \infty$.

7.7 DIGITAL SIGNAL PROCESSORS

Microprocessor architectures can be optimized for increased efficiency in certain applications through the inclusion of special instructions and execution units. One major class of application-specific microprocessors is the *digital signal processor*, or DSP. DSP entails a microprocessor mathematically manipulating a sampled analog signal in a way that emulates transformation of that signal by discrete analog components such as filters or amplifiers. To operate on an analog signal digitally, the analog signal must be sampled by an analog-to-digital converter, manipulated, and then reconstructed with a digital-to-analog converter. A rough equivalency of digital signal processing versus conventional analog transformation is shown in Fig. 7.11 in the context of a simple filter.

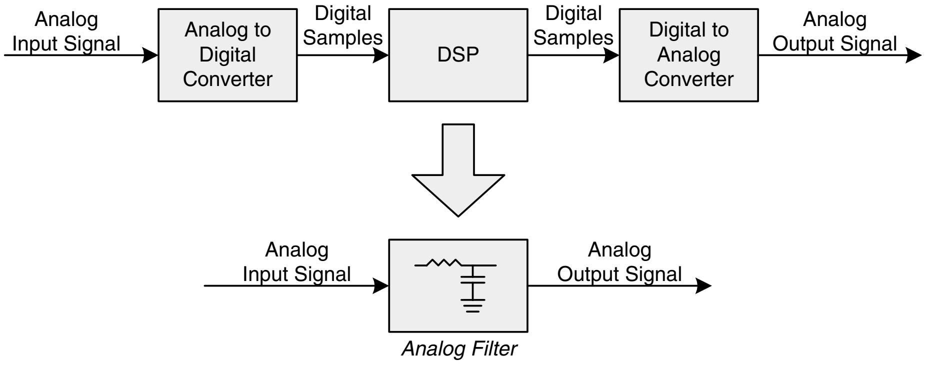

FIGURE 7.11 Digital signal processing.