Text is stored in memory by assigning a specific bit pattern to each character in the alphabet. The *ASCII code* is the most popular code used to represent characters in microcomputer systems. As with floating-point numbers, any processing of the ASCII-coded characters is via software routines since the CPU32 has no instructions that operate specifically on characters.

The basic characteristics of the data types commonly used in MC68332-based systems are discussed in this chapter. Machine instructions to manipulate the data types and various other programming considerations are discussed in later chapters. In particular, arithmetic operations using integers are treated in detail in Chapter 7.

From Harman Motorola M68332

3.1 NUMBER REPRESENTATION

This section describes the representation of positive and negative integers and fractions. A general formulation with the base or radix r for each number representation is presented and then applied to the discussion of binary values with $r = 2$ and other bases as appropriate. The generalized presentation is useful for conversion of numbers from one base to another and techniques of numerical analysis. Representation of binary numbers in the sign-magnitude , one's-complement, and two's-complement systems are presented. Decimal representations for the nine's complement and ten's complement systems are also presented.

3.1.1 Nonnegative Integers

A nonnegative integer in base r is written in positional notation as

$$N_r = (d_{m-1}d_{m-2}\cdots d_0)_r \tag{3.1}$$

where each digit d_i has one of the distinct values $[0, 1, 2, \ldots, r-1]$ and m represents the base 10 or decimal number of digits in the integer. Thus the number 324 would have $d_0 = 4$, $d_1 = 2$, and $d_2 = 3$ in Equation 3.1 and could be written as

$$N_{10} = 324_{10}.$$

For numbers in base 10, the subscript is omitted if no confusion could result from its omission. The form specified by Equation 3.1 is generally referred to as *positional* notation. The position of the digit starting from the rightmost digit represents a power of the base r; that is, 324 represents 4 ones (4×10^0), 2 tens (2×10^1), and 3 hundreds (3×10^2). Mathematically, the value of the number is calculated as

$$\begin{aligned} N_r &= d_{m-1}r^{m-1} + d_{m-2}r^{m-2} + \cdots + d_1 r + d_0 \\ &= \sum_{i=0}^{m-1} d_i r^i \end{aligned} \tag{3.2}$$

in which the digits are restricted in value such that $0 \leq d_i \leq r - 1$. Thus, the number 324 can be calculated as

$$324 = 3 \times 10^2 + 2 \times 10^1 + 4 \times 1.$$

The arithmetic operations in Equation 3.2 could be carried out in any number base and this equation is frequently used to determine the decimal equivalent of a number in another base.

Table 3.1 Digits in Various Number Systems

Number system	Base r	Digits
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Binary	2	0, 1

Table 3.1 lists the range of possible values in the digits in the hexadecimal, decimal, octal, and binary number systems. The decimal system is, of course, used primarily for ordinary arithmetic by human beings, and the binary system is used for computer arithmetic. Octal and hexadecimal representations are convenient for writing long binary numbers. For example,

$$01011010_2 = 5A_{16} = 132_8.$$

The decimal value of the number represented in these bases is obtained from Equation 3.2 by converting the base r digits to decimal equivalents such that

$$\begin{aligned} N &= 5 \times 16^1 + 10 \times 1 \\ &= 1 \times 8^2 + 3 \times 8^1 + 2 \times 1 \\ &= 90_{10}. \end{aligned}$$

The hexadecimal digits [A, B,...., F] represent the decimal numbers [10, 11,...., 15] in the conversion from hexadecimal to decimal.

Example 3.1

Consider the largest m-digit positive integer in positional notation,

$$N_r = ((r-1)(r-1)\cdots(r-1)),$$

as in $(1111\ldots1111)_2$ or $(9999\ldots9999)_{10}$ with m digits each. The sum of Equation 3.2 indicates that the decimal value is

$$N = (r-1)\sum_{i=0}^{m-1} r^i$$

which is an easily summed geometric series. The result is $r^m - 1$. For example, an 8-bit binary number has a maximum value of $2^8 - 1$, or 255.

Positive fractional values. The positional notation defined by Equation 3.1 is valid for integers only. If a fraction is to be represented, a radix point in the base r is used to separate the integer from the fractional part of the number. The radix point is called the *binary point* in base 2 and the *decimal point* in base 10. Thus, 324.14 has the value

$$3 \times 10^2 + 2 \times 10^1 + 4 \times 1 + 1 \times 10^{-1} + 4 \times 10^{-2}.$$

In general, a k-digit positive fraction is written with a leading radix point as

$$0.(d_{-1}d_{-2}\cdots d_{-k}) \tag{3.3}$$

with the value

$$0.n_r = d_{-1}r^{-1} + d_{-2}r^{-2} + \cdots + d_{-k}r^{-k} \tag{3.4}$$

where the negative subscript for the digits indicates the appropriate negative power of r.

Internally, the processor performs arithmetic on integers without taking into account the position of the radix point . It is then possible to interpret the internal value of a fraction by writing $0.n_r$ in the form

$$0.n_r = r^{-k} \times (d_{-1}d_{-2}\cdots d_{-k})_r \tag{3.5}$$

with the value in parentheses treated as an integer value. For example, the number 0.1000_2 (0.5_{10}) can be written as

$$2^{-4} \times (1000.)_2 = 2^{-4} \times 8$$

both of which have the value 0.5, as expected. The scaling factor r^{-k} has the effect of shifting the radix point k positions to the left. Thus, $0.n_r r^k$ may be used internally as an integer operand and the final result scaled by r^{-k}. For example, the addition of the binary values 0.1000 (0.5_{10}) and 0.0100 (0.25_{10}) can be accomplished as

$$\begin{array}{rl} 1000. & \times 2^{-4} \\ +0100. & \times 2^{-4} \\ \hline 1100. & \times 2^{-4} \end{array}$$

The machine addition results in 1100_2 or 12 decimal, and the programmer must apply the scale factor to obtain the correct arithmetic result:

$$12 \times 2^{-4} = 0.75.$$

In addition and subtraction, each scaled value must have the same scaling factor. The choice of the scaling factor may cause the radix point to be at the right of the number (integer), at the left (fraction), or anywhere within the number. Thus, the value 0.5 in four-digit binary could be written as a fraction

$$.1000_2$$

as an integer with scaling 2^{-4} as

$$(1000.)_2 \times 2^{-4}$$

or as a mixed quantity scaled arbitrarily: for example,

$$(10.00)_2 \times 2^{-2}.$$

When the radix point is fixed for a particular problem and the programmer must take the scaling into account, the system is called a *fixed-point* representation. All integer operations with the CPU32, such as addition or subtraction, assume that the scaling factor is 2^0. Therefore, the binary point is on the right. The importance of Equation 3.5 is that both for analysis and for machine operations, a fractional value may be treated as an integer during all the intermediate steps of a computation. The appropriate scale factor can be applied as the last step when the actual numerical value is desired.

$\frac{1}{3} = .33\ldots$

$\frac{1}{11} = (\ )(\ )$

Example 3.2

The first example showed that the largest m-digit integer for unsigned integers has the value $r^m - 1$. Thus, the largest 16-bit (binary) integer

$$1111\ 1111\ 1111\ 1111_2$$

has the value

$$2^{16} - 1 = 65,535$$

in decimal representation. The largest 16-bit fraction

$$0.1111\ 1111\ 1111\ 1111_2$$

has the decimal value ← shift left 16 places

$$2^{-16} \times 65,535 = 0.99998474.$$

This is obtained by scaling the 16-bit fraction as

$$2^{-16} \times (2^{16} - 1) = 1 - 2^{-16}$$

and performing the arithmetic on a calculator with a sufficient number of decimal places.

EXERCISES

3.1.1.1. Convert the binary number

$$0100.0110_2$$

to decimal.

3.1.1.2. What is the decimal value of

$$1111\ 1111\ .\ 1111\ 1111\ 1111\ 1111_2$$

to five decimal places in the fraction?

3.1.1.3. Compute the decimal value of the following numbers:
(a) 130_9;
(b) 120_5;
(c) 0.7632_8;
(d) $F00A_{16}$.

3.1.1.4. If the base x number

$$111_x = 31_{10}$$

what is the base x?

3.1.1.5. Compute the largest integer representable in a 32-bit computer word. Give the answer as a decimal value.

3.1.2 Representation of Signed Numbers

The positive integers, including zero, can be conveniently represented as shown in Section 3.1.1. However, to represent the complete set of integers, which includes positive integers, zero, and negative integers, a notation for negative values is necessary. In ordinary arithmetic, a negative number is represented by prefixing the magnitude (or absolute value) of the number with a minus sign. Thus -5 is a negative integer with magnitude of 5. For hand calculations, the use of separate symbols to indicate positive (+) and negative (−) numbers is convenient.

Computer arithmetic circuits to manipulate positive and negative integers are also simplified if one of the digits in the positional notation of a number is used to indicate the sign of the integer. Two such possible representations of signed integers are *sign-magnitude* notation and *complement* notation. In both notations, the most significant digit on the left in the positional form of the number indicates the sign. Negative fractions can also be represented in either of these systems. For fractions, the sign digit is written to the left of the radix point.

The binary arithmetic instructions of the CPU32 operate directly on integers in two's-complement notation. Integers in other binary notations or fractions must be manipulated by programs designed for that purpose. The treatment of decimal numbers by the CPU is discussed in Section 3.2 although the mathematical representation of decimal numbers is first introduced in this section.

Sign-magnitude representation. The sign-magnitude representation of a number in positional notation has the form

$$N_r = (d_{m-1}d_{m-2}\cdots d_1 d_0)_r, \tag{3.6}$$

where the sign of the number is indicated by the most significant (leftmost) digit:

$$d_{m-1} = \begin{cases} 0 & \text{if } N_r \geq 0 \\ r-1 & \text{if } N_r < 0 \end{cases} \tag{3.7}$$

Thus, using the sign-magnitude representation, 1011_2 and 9003 are four-digit negative numbers in the binary and decimal systems, respectively. The magnitude of the number, written $|N_r|$, is

$$|N_r| = \sum_{i=0}^{m-2} d_i r^i \tag{3.8}$$

where only the first $m-1$ digits from the right are considered. The representation of a positive number differs from that of the corresponding negative number only in the sign digit. The digits $(d_{m-2}d_{m-3}\cdots d_1 d_0)$ indicate the magnitude. According to the definitions and equation 3.8, the four-digit number 0011_2 represents +3, and 1011_2 represents −3.

The number of digits, including the sign digit, in the representation must be specified or confusion could result. For example, 1011_2 in an eight-digit representation is assumed to be 00001011_2, which has the decimal value 11. For binary values, a negative fraction in sign-magnitude notation has a leading digit of 1 followed by the fractional part. Thus 1.100_2 is the number −0.5.

Example 3.3

The number 16 is written in a 16-bit binary system as

$$0000\ 0000\ 0001\ 0000_2$$

The number −16 has the sign-magnitude representation

$$1000\ 0000\ 0001\ 0000_2$$

Complement representation. Most microcontrollers, including the MC68332, have arithmetic instructions that operate on negative numbers represented in a *complement* number system.[1] In these systems, positive numbers have the same representation as in sign-magnitude notation, but the negative numbers are formed by computing the complement of the number according to the rules of the specific system

[1] Complement representations have an advantage over sign and magnitude notation because the sign digit does not have to be treated in a special way during addition and subtraction. This simplifies the arithmetic circuits of the CPU somewhat, as is discussed in several references in the Further Reading section of this chapter.

being used. The two most common complement systems are the *radix-complement* and the *diminished radix-complement* systems. The general theory of these systems will be presented first. Then, the two's and ten's complements of numbers will be discussed as examples of radix-complement numbers. The one's and nine's complements are examples of diminished radix-complement systems for base 2 and base 10, respectively.

In general form, the radix complement of an m-digit number is computed mathematically as

$$N_r' = r^m - N_r \tag{3.9}$$

where N_r' is the radix complement of the base r number N_r. In the machine computation, only m-digit values can be represented. If any operation produces a result that requires more than m digits, the higher-order digit is ignored. This is taken to be an out-of-range condition. A machine error of this type is called *overflow*.

The two radix-complement systems used with the CPU32 instructions are the two's-complement and the ten's complement systems. The two's complement of a number N_2 given by Equation 3.9 using 2 as the base r is

$$N_2' = 2^m - N_2. \tag{3.10}$$

Thus, the four-digit two's-complement form of -1 is

$$N_2' = 2^4 - 1 = 1\,0000 - 0001 = 1111_2$$

If the number and its complement are added:

$$\begin{array}{rl} 0001 & N \\ +1111 & N' \\ \hline 10000 & \end{array}$$

the result is 0 to four digits, as expected. In a two's-complement system, negative values always have a leading digit of 1 and positive values have 0 as the leading digit. Thus $+4 = 0100_2$ and $-4 = 1100_2$. If addition is performed in a four-digit representation on two positive numbers, the result must not be greater than 7 or overflow occurs. This limits the range of the m-bit positive numbers to the decimal value $2^{m-1} - 1$. In the case of 4-bit numbers, the addition of 4 + 5 in binary yields

$$\begin{array}{r} 0100 \\ +0101 \\ \hline 1001_2 \end{array}$$

which is a negative number in two's-complement notation. The magnitude as derived from Equation 3.9 would be

$$N_2 = 2^4 - 1001 = 1\,0000 - 1001 = 0111_2$$

which is +7 in decimal, clearly an error. In the CPU32, an indication is given when such an overflow occurs and the programmer must make provisions in the program for such occurrences.

In the ten's complement system, the ten's complement of a number N is formed as

$$N' = 10^L - N \tag{3.11}$$

in an L-digit representation. For four digits, -1 is represented as

$$N' = 10^4 - 1 = (10,000 - 1) = 9999.$$

The CPU32 has arithmetic instructions to operate on numbers represented in the ten's-complement system.

The diminished radix complement is computed as

$$\overline{N}_r = r^m - N_r - 1 \tag{3.12}$$

which is one less than the radix-complement value computed by Equation 3.9. The diminished radix complement, or simply complement as it is usually called, is the one's complement for binary values and the nine's complement for decimal numbers. The four-digit decimal value 0002 has the complement

$$\overline{N}_r = (10^4 - 0002) - 1 = 9999 - 0002 = 9997. \tag{3.13}$$

From the complement, the radix complement is formed by adding 1, as a comparison of Equations 3.12 and 3.9 shows.

Table 3.2 Complement Systems

Value	One's	Two's	Value	Nine's	Ten's
7	0111	0111	4999	4999	4999
6	0110	0110	4998	4998	4998
5	0101	0101	·	·	·
4	0100	0100	·	·	·
3	0011	0011	·	·	·
2	0010	0010	0002	0002	0002
1	0001	0001	0001	0001	0001
0	0000	0000	0000	0000	0000
−0	1111	—	−0000	9999	—
−1	1110	1111	−0001	9998	9999
−2	1101	1110	−0002	9997	9998
−3	1100	1101	·	·	·
−4	1011	1100	·	·	·
−5	1010	1011	·	·	·
−6	1001	1010	·	·	·
−7	1000	1001	−4999	5000	5001
−8	—	1000	−5000	—	5000

Table 3.2 lists the radix complement for four-digit binary and decimal numbers. The one's- and nine's-complement values are also presented for comparison. Notice that in the nine's- and ten's-complement notation, the negative values have leading

digits in the range 5 through 9. The sign digit is not unique as it is in the case of two's complement negative values.

Example 3.4

The radix complement of a number is easily computed by complementing each digit [subtracting it from $(r-1)$] and adding 1 to the result formed from the complemented digits. Thus, the value -2 is represented as follows in various four-digit systems:

$$\begin{aligned}
\text{2's complement:} &-2 = (1111 - 0010) + 1 = 1110_2 \\
\text{10's complement:} &-2 = (9999 - 0002) + 1 = 9998 \\
\text{16's complement:} &-2 = (\text{FFFF} - 0002) + 1 = \text{FFFE}_{16}
\end{aligned}$$

Example 3.5

For a fraction of length k digits, the radix complement is computed by complementing each digit and adding r^{-k} (not 1) to the result. Thus, the number

$$0101.01_2 = 5.25$$

has as its complement the value

$$1010.10_2$$

Its radix or two's-complement representation would be

$$\begin{array}{r} 1010.10 \\ +\quad .01 \\ \hline 1010.11_2 \end{array}$$

where the value 2^{-2} is added to the one's complement of the number since $k = 2$. Similarly, the fraction 0.01_2 has complement 1.10_2 and two's complement 1.11_2.

Number range. The range of integers (or fractions) for a given number system is specified by the smallest and largest values that can be represented. For positive integers represented in m digits, for example, there are r^m possible values with a numerical range of 0 to $r^m - 1$. For the 8-bit representation of a positive binary integer, the range is 0 to $2^8 - 1$, or 0 to 255. The range of signed integers in an m-digit representation still allows r^m values, but one-half of these values are negative numbers.

The maximum positive integer in sign-magnitude , one's-complement, or two's-complement representation is

$$(0111\ldots111)_2$$

where $(m-1)$ 1's are shown. The maximum decimal value is thus $2^{m-1}-1$, considering the discussion in Example 3.1. In an 8-bit representation, the largest positive number is $2^7 - 1$, or 127. In sign-magnitude notation, the most negative value is

$$(1111\ldots111)_2 = (2^{m-1} - 1).$$

The most negative one's-complement number $(100\ldots00)_2$ has this same numerical value. Both of these systems allow a positive and a negative value of zero since the "positive" zero

$$(000\ldots000)_2$$

has negative values of $(1000\ldots000)_2$ and $(1111\ldots111)_2$ in the sign-magnitude and one's-complement notations, respectively. In the two's-complement notation, however, only one value of zero is allowed since the two's complement of the number 0 is the same value to m bits. Since there are a total of 2^m values for each m-bit representation, the two's-complement notation allows one more negative value than the others.

The two's complement of the positive value

$$(011\ldots111)_2$$

is the negative number

$$(100\ldots001)_2 = -(2^{m-1} - 1)$$

for m bits. The integer

$$(100\ldots000)_2$$

must then represent -2^{m-1} with no positive counterpart.

The m-digit ten's complement allows 10^m values in the range

$$-10^m/2 \text{ to } 10^m/2 - 1$$

as from -5000 to $+4999$ in the four-digit representation shown in Table 3.2. The positive values are

$$0, 1, 2, \ldots, 499\ldots99$$

for m digits and the negative values are represented as

$$999\ldots999, 999\ldots998, \ldots, 500\ldots001, 500\ldots000$$

with values -1, -2.... The nine's-complement representation of the negative numbers has a negative 0 and consequently one less nonzero negative value than the ten's-complement notation allows, that is, a range from

$$-10^m/2 + 1 \text{ to } 10^m/2 - 1.$$

In this case, the magnitude of the number is restricted so that

$$|N| \leq 10^m/2 - 1$$

which causes the most significant digits of 0, 1, 2, 3, or 4 to indicate a positive number and the digits 5, 6, 7, 8, or 9 to indicate a negative value.

Example 3.6

The largest positive number for an m-digit radix-complement number in base r is

$$(1/2) \times r^m - 1$$

since there are $(1/2) \times r^m$, positive integers, including zero. Thus, the two's-complement maximum value is

$$(1/2) \times 2^m - 1 = 2^{m-1} - 1$$

while the ten's complement number has the maximum value of

$$(1/2) \times 10^m - 1$$

as indicated in the previous discussion. The four-digit binary number allows values up to +7 in the two's-complement system, while the four-digit decimal value has a largest positive value of 4999.

Example 3.7

Applying the formulas for the most negative and most positive numbers in m-bit representations gives the following results:

Representation	Most negative	Most positive
Sign-magnitude	$-2^{m-1} + 1$	$2^{m-1} - 1$
One's complement	$-2^{m-1} + 1$	$2^{m-1} - 1$
Two's complement	-2^{m-1}	$2^{m-1} - 1$

For a 16-bit representation, the sign-magnitude and one's-complement numbers range from $-32,767$ to $32,767$, and the two's complement numbers range from $-32,768$ to $32,767$.

If a signed, binary fraction is represented as

$$(b_0 b_{-1} \cdots b_{-(k-1)})_2$$

the range is determined by scaling by $2^{-(k-1)}$. For example, the 8-bit fraction in two's-complement representation has the range -1 to $1 - 2^{-7}$. The binary values in this range are

1.000 0000 (−1)
1.000 0001
.
.
.
0.000 0000 (0)
.
.
.
0.111 1110
0.111 1111 $(1 - 2^{-7})$

EXERCISES

3.1.2.1. Find the two's-complement representation of the following numbers:
(a) -0647_{16} to 16 bits;
(b) -11_{10} to 16 bits;
(c) -00101.110_2 to 8 bits.

3.1.2.2. The most negative two's complement number is $100\ldots0_2$ for m bits. What value results when the two's complement of the number is taken?

3.1.2.3. In the two's-complement notation, the sign bit has the weight -2^{m-1} for an m-bit integer. Determine the procedure to extend the m-bit number to $2m$ bits for the following:
(a) a positive number;
(b) a negative number.
This is called *sign extension*.

3.1.2.4. Represent the given numbers in the notation specified:
(a) nine's complement of 653.72 with five digits;
(b) -223_{16} in sign-magnitude form with four hexadecimal digits;
(c) $-3/8$ in one's-complement form with 8 bits, including the sign bit.

3.1.2.5. Determine the range of numbers for the sign-magnitude , one's-complement, and two's-complement forms for an m-bit representation if:
(a) $m = 8$;
(b) $m = 16$;
(c) $m = 32$.

3.1.2.6. Suppose the largest positive number is limited to 999 (i.e., three digits) in a four-digit, ten's-complement representation. Determine the corresponding range and representation of the negative numbers. Note that negative numbers always begin with 9 as the sign digit when the magnitude of the numbers is limited in this way.

3.1.3 Conversions between Representations

The number systems of most interest to computer users are the binary, octal, decimal, and hexadecimal systems. Although the internal machine representation in microcomputers is binary, the other representations are important for the convenience of the user. In this regard, the conversions of numbers in other bases to decimal, and vice versa, are frequently required.

Conversion between arbitrary number bases is sometimes necessary, although in computer work, the conversions between binary, octal, and hexadecimal systems are of greatest importance. Fortunately, conversions between these bases are straightforward.

Conversion to decimal for positive numbers. The number N_r in base can be represented in decimal as

$$N.n = D_{m-1}r^{m-1} + \ldots + D_0 + D_{-1}r^{-1} + \ldots + D_{-k}r^{-k} \qquad (3.14)$$

where the D_i are the equivalent values in the base 10 of the digits in base r. The number is converted by multiplying each digit by the appropriate power of r and adding each result to the sum. The number is designated here as $N.n$ to emphasize the fact that it has an integral as well as a fractional part.

Example 3.8

To convert $11\ 1110_2$ to decimal, the value is computed as a series from Equation 3.2 or 3.14 with the result

$$\begin{aligned} N &= 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \\ &= 32 + 16 + 8 + 4 + 2 \\ &= 62. \end{aligned}$$

Example 3.9

The value of 0.502_8 in decimal is

$$\begin{aligned} 0.n &= 5 \times 8^{-1} + 0 + 2 \times 8^{-3} \\ &= 0.6250 + 0.003906250 \\ &= 0.62890625_{10} \end{aligned}$$

as determined by Equation 3.4 or Equation 3.14.

Conversion from decimal to any number base. To convert a decimal number by hand to a number in another base, it is convenient to work in the decimal system with the series representation of the number. Conversion of a positive integer is accomplished by repeated division by the new radix using successive remainders as digits in the new system. A fraction is converted by repeated multiplication by the radix, with the resulting integer parts of the products taken as digits of the result.

Example 3.10

To convert 3964_{10} to octal, the number is repeatedly divided by 8, as follows:

$$\begin{aligned} 3964/8 &= 495 + (4/8) \\ 495/8 &= 61 + (7/8) \\ 61/8 &= 7 + (5/8) \\ 7/8 &= 0 + (7/8). \end{aligned}$$

The remainders of each division, represented by the numerators of each fraction, are digits in the resulting answer. The order of these digits is the reverse of the order in which they were obtained. Thus, in the example above,

$$3964_{10} = 7574_8.$$

Example 3.11

The number 0.78125_{10} is converted to a hexadecimal fraction as follows:

$$0.78125 \times 16 = 12.0 + 0.5$$
$$0.5 \times 16 = 8.0 + 0.$$

The result is therefore $0.78125_{10} = 0.C8_{16}$.

To understand the theory of these conversions, write Equation 3.14 in the form

$$N = ((\cdots((d_{m-1}r + d_{m-2})r + d_{m-3})r + \cdots + d_1)r + d_0)$$

for the integer part and set it equal to the decimal value it represents. Then divide N by the base desired. Using 8 as the base in Example 3.10, the first remainder is the octal value d_0. Continuing to divide the whole numbers (quotients) by the base successively yields $d_0, d_1, d_2, \ldots, d_{m-1}$ in that order. Try the fractional part of Equation 3.14 in a similar manner with negative powers of r to see how the value in Example 3.11 was computed.

Conversion of a number from any base to another. A number written in base r_1 can be converted to a number in base r_2 by performing the arithmetic operations in a base other than decimal. For human computation a more acceptable method is to convert the selected number to decimal from base r_1 and then convert the result from decimal to base r_2.

Example 3.12

Converting 112_3 to base 5 is accomplished by converting 112_3 to decimal:

$$1 \times 3^2 + 1 \times 3^1 + 2 = 9 + 3 + 2 = 14_{10}.$$

Then, the decimal number 14 is converted to base 5 in the form

$$14/5 = 2 + (4/5)$$
$$2/5 = 0 + (2/5)$$

or $14_{10} = 24_5$. Thus $112_3 = 24_5$.

Conversion of positive numbers with bases that are powers of 2. If the relationship between a number base r_1 and another base r_2 is of the form

$$r_2 = r_1^L$$

where L is a positive or negative integer, conversion between the bases is particularly simple using positional notation. In particular, since the binary, octal, and hexadecimal bases are related as

$$16 = 2^4$$
$$8 = 2^3$$

the conversion from binary to octal or binary to hexadecimal requires only the grouping of the binary digits by threes or fours, respectively. The conversion from octal