Number Systems

We noted in several places that a binary scheme having only the two binary digits 0 and 1 is used to represent information in a computer. In PART OF THE PICTURE: Data Representation in Chapter 2, we described how these binary digits, called *bits,* are organized into groups of 8 called *bytes,* and bytes in turn are grouped together into *words.* Common word sizes are 16 bits (= 2 bytes) and 32 bits (= 4 bytes). Each byte or word has an *address* that can be used to access it, making it possible to store information in and retrieve information from that byte or word. In this appendix we describe the binary number system and how numbers can be converted from one base to another.

The number system that we are accustomed to using is a **decimal** or **base-10** number system, which uses the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. The significance of these digits in a numeral depends on the positions that they occupy in that numeral. For example, in the numeral

485

the digit 4 is interpreted as	4 hundreds
and the digit 8 as	8 tens
and the digit 5 as	5 ones

Thus, the numeral 485 represents the number four-hundred eighty-five and can be written in **expanded form** as

$$(4 \times 100) + (8 \times 10) + (5 \times 1)$$

or

$$(4 \times 10^2) + (8 \times 10^1) + (5 \times 10^0)$$

The digits that appear in the various positions of a decimal (base-10) numeral, thus, are coefficients of powers of 10.

Similar positional number systems can be devised using numbers other than 10 as a base. The **binary** number system uses 2 as the base and has only two digits, 0 and 1. As in a decimal system, the significance of the bits in a binary numeral is determined by their positions in that numeral. For example, the binary numeral

101

can be written in expanded form (using decimal notation) as

$$(1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$

that is, the binary numeral 101 has the decimal value

$$4 + 0 + 1 = 5$$

Similarly, the binary numeral 111010 has the decimal value

$$(1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$$
$$= 32 + 16 + 8 + 0 + 2 + 0$$
$$= 58$$

When necessary, to avoid confusion about which base is being used, it is customary to write the base as a subscript for nondecimal numerals. Using this convention, we could indicate that 5 and 58 have the binary representations just given by writing

$$5 = 101_2$$

and

$$58 = 111010_2$$

Two other nondecimal numeration systems are important in the consideration of computer systems: **octal** and **hexadecimal.** The octal system is a base-8 system and uses the eight digits 0, 1, 2, 3, 4, 5, 6, and 7. In an octal numeral such as

$$1703_8$$

the digits are coefficients of powers of 8; this numeral is therefore an abbreviation for the expanded form

$$(1 \times 8^3) + (7 \times 8^2) + (0 \times 8^1) + (3 \times 8^0)$$

and thus has the decimal value

$$512 + 448 + 0 + 3 = 963$$

A hexadecimal system uses a base of 16 and the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (10), B (11), C (12), D (13), E (14), and F (15). The hexadecimal numeral

$$5E4_{16}$$

has the expanded form

$$(5 \times 16^2) + (14 \times 16^1) + (4 \times 16^0)$$

which has the decimal value

$$1280 + 224 + 4 = 1508$$

Table A3.1 shows the decimal, binary, octal, and hexadecimal representations for the first 31 nonnegative integers.

In the decimal representation of real numbers, digits to the left of the decimal point are coefficients of nonnegative powers of 10, and those to the right are coefficients of negative powers of 10. For example, the decimal numeral 56.317 can be written in expanded form as

$$(5 \times 10^1) + (6 \times 10^0) + (3 \times 10^{-1}) + (1 \times 10^{-2}) + (7 \times 10^{-3})$$

or, equivalently, as

TABLE A3.1 NUMERIC REPRESENTATION

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F

$$(5 \times 10) + (6 \times 1) + \left(3 \times \frac{1}{10}\right) + \left(1 \times \frac{1}{100}\right) + \left(7 \times \frac{1}{1000}\right)$$

Digits in the binary representation of a real number are coefficients of powers of two. Those to the left of the binary point are coefficients of nonnegative powers of two, and those to the right are coefficients of negative powers of two. For example, the expanded form of 110.101_2 is

$$(1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$$

and thus has the decimal value

$$4 + 2 + 0 + \frac{1}{2} + 0 + \frac{1}{8} = 6.625$$

Similarly, in octal representation, digits to the left of the octal point are coefficients of nonnegative powers of eight, and those to the right are coefficients of negative powers of eight. And in hexadecimal representation, digits to the left of the octal point are coefficients of nonnegative powers of sixteen, and those to the right are coefficients of negative powers of sixteen. Thus, the expanded form of 102.34_8 is

$$(1 \times 8^2) + (0 \times 8^1) + (2 \times 8^0) + (3 \times 8^{-1}) + (4 \times 8^{-2})$$

which has the decimal value

$$64 + 0 + 2 + \frac{3}{8} + \frac{4}{64} = 66.4375$$

The expanded form of $1AB.C8_{16}$ is

$$(1 \times 16^2) + (10 \times 16^1) + (11 \times 16^0) + (12 \times 16^{-1}) + (8 \times 16^{-2})$$

whose decimal value is

$$256 + 160 + 11 + \frac{12}{16} + \frac{8}{256} = 417.78125$$

Exercises

Convert each of the binary numerals in exercises 1–6 to base ten.

1. 1001
2. 110010
3. 1000000
4. 111111111111111 (fifteen 1s)
5. 1.1
6. 1010.10101

Convert each of the octal numerals in exercises 7–12 to base ten.

7. 23
8. 77777
9. 2705
10. 7.2
11. 10000
12. 123.45

Convert each of the hexadecimal numerals in exercises 13–18 to base ten.

13. 12
14. FFF
15. 1AB
16. 8.C
17. ABC
18. AB.CD

19–24. Converting from octal representation to binary representation is easy, as we need only replace each octal digit with its three-bit binary equivalent. For example, to convert 617_8 to binary, replace 6 with 110, 1 with 001, and 7 with 111, to obtain 110001111_2. Convert each of the octal numerals in exercises 7–12 to binary numerals.

25–30. Imitating the conversion scheme in exercises 19–24, convert each of the hexadecimal numerals in exercises 13–18 to binary numerals.

31–36. To convert a binary numeral to octal, place the digits in groups of three, starting from the binary point, or from the right end if there is no binary point, and replace each group with the corresponding octal digit. For example, $10101111_2 = 010\ 101\ 111_2 = 257_8$. Convert each of the binary numerals in exercises 1–6 to octal numerals.

37–42. Imitating the conversion scheme in exercises 31–36, convert each of the binary numerals in exercises 1–6 to hexadecimal numerals.

One method for finding the base-b representation of a whole number given in base-ten notation is to divide the number repeatedly by b until a quotient of zero results. The successive remainders are the digits from right to left of the base-b representation. For example, the binary representation of 26 is 11010_2, as the following computation shows:

$$\begin{array}{r} 0 \text{ R } 1 \\ 2\overline{)1} \text{ R } 1 \\ 2\overline{)3} \text{ R } 0 \\ 2\overline{)6} \text{ R } 1 \\ 2\overline{)13} \text{ R } 0 \\ 2\overline{)26} \phantom{\text{ R } 0} \end{array}$$

Convert each of the base-ten numerals in exercises 43–46 to (a) binary, (b) octal, and (c) hexadecimal:

43. 27

44. 314

45. 99

46. 5280

To convert a decimal fraction to its base-b equivalent, repeatedly multiply the fractional part of the number by b. The integer parts are the digits from left to right of the base-b representation. For example, the decimal numeral 0.6875 corresponds to the binary numeral 0.1011_2, as the following computation shows:

$$\begin{array}{r|r} & .6875 \\ & \times\ \ 2 \\ \hline 1 & .375 \\ & \times\ 2 \\ \hline 0 & .75 \\ & \times 2 \\ \hline 1 & .5 \\ & \times 2 \\ \hline 1 & .0 \end{array}$$

Convert each of the base-ten numerals in exercises 47–51 to (a) binary, (b) octal, and (c) hexadecimal:

47. 0.5

48. 0.25

49. 0.625

50. 16.0625

51. 8.828125

Even though the base-ten representation of a fraction may terminate, its representation in some other base need not terminate. For example, the following computation shows that the binary representation of 0.7 is $(0.10110011001100110011001100110\ldots)_2$, where the block of bits 0110 is repeated indefinitely. This representation is commonly written as $0.1\overline{0110}_2$.

$$
\begin{array}{c|r}
 & .7 \\
 & \times 2 \\ \hline
1 & .4 \\
 & \times 2 \\ \hline
0 & .8 \\
 & \times 2 \\ \hline
1 & .6 \\
 & \times 2 \\ \hline
1 & .2 \\
 & \times 2 \\ \hline
0 & .4
\end{array}
$$

Convert each of the base-ten numerals in exercises 52–55 to (a) binary, (b) octal, and (c) hexadecimal:

52. 0.3

53. 0.6

54. 0.05

55. $0.\overline{3} = 0.33333\cdots = 1/3$

Exercises 56–67 assume the two's-complement representation of integers described in the section PART OF THE PICTURE: Data Representation in Chapter 2.

Find the decimal value of each of the 32-bit integers in exercises 56–61, assuming two's complement representation.

56. 00000000000000000000000001000000

57. 11111111111111111111111111111110

58. 1111111110111111

59. 0000000011111111

60. 1111111100000000

61. 1000000000000001

Find the 32-bit two's complement representation of each of the integers in exercises 62–67.

62. 255

63. 1K

64. −255

65. −256

66. -34567_8

67. $-3ABC_{16}$

Assuming the IEEE floating point representation of real numbers described in the section PART OF THE PICTURE: Data Representation in Chapter 2, indicate how each of the real numbers in exercises 68–73 would be stored.

68. 0.375

69. 37.375

70. 0.03125

71. 63.84375

72. 0.1

73. 0.01

Assuming the representation of character strings described in the section PART OF THE PICTURE: Data Representation in Chapter 2 and using the table of ASCII characters in appendix A, indicate how each of the character strings in exercises 74–79 would be stored in 4-byte words.

74. to

75. FOUR

76. Amount

77. etc.

78. J. Doe

79. A#*4–C