Number Systems

This appendix contains a detailed introduction to number systems and their underlying characteristics. The particular focus is on the binary number system, its use with computers, and its similarities to other number systems. This introduction also covers conversions between bases.

In our everyday lives, we use the *decimal number system* to represent values, to count, and to perform arithmetic. The decimal system is also referred to as the *base-10 number system*. We use 10 digits (0 through 9) to represent values in the decimal system.

Computers use the *binary number system* to store and manage information. The binary system, also called the *base-2 number system*, has only two digits (0 and 1). Each 0 and 1 is called a *bit*, short for binary digit. A series of bits is called a *binary string*.

There is nothing particularly special about either the binary or decimal systems. Long ago, humans adopted the decimal number system probably because we have 10 fingers on our hands. If humans had 12 fingers, we would probably be using a base-12 number system regularly and find it as easy to deal with as we do the decimal system now. It all depends on what you get used to. As you explore the binary system, it will become more familiar and natural.

Binary is used for computer processing because the devices used to manage and store information are less expensive and more reliable if they have to represent only two possible values. Computers have been made that use the decimal system, but they are not as convenient.

There are an infinite number of number systems, and they all follow the same basic rules. You already know how the binary number system works, but you just might not be aware that you do. It all goes back to the basic rules of arithmetic.

Place Value

In decimal, we represent the values of 0 through 9 using only one digit. To represent any value higher than 9, we must use more than one digit. The position of each digit has a *place value* that indicates the amount it contributes to the overall value. In decimal, we refer to the one's column, the ten's column, the hundred's column, and so on forever.

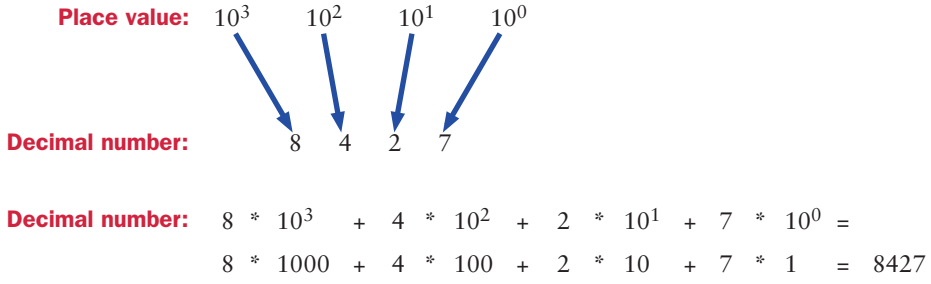

FIGURE B.1 Place values in the decimal system

Each place value is determined by the *base* of the number system, raised to increasing powers as we move from right to left. In the decimal number system, the place value of the digit furthest to the right is 10^0, or 1. The place value of the next digit is 10^1, or 10. The place value of the third digit from the right is 10^2, or 100, and so on. Figure B.1 shows how each digit in a decimal number contributes to the value.

The binary system works the same way except that we exhaust the available digits much sooner. We can represent 0 and 1 with a single bit, but to represent any value higher than 1, we must use multiple bits.

The place values in binary are determined by increasing powers of the base as we move right to left, just as they are in the decimal system. However, in binary, the base value is 2. Therefore the place value of the bit furthest to the right is 2^0, or 1. The place value of the next bit is 2^1, or 2. The place value of the third bit from the right is 2^2, or 4, and so on. Figure B.2 shows a binary number and its place values.

The number 1101 is a valid binary number, but it is also a valid decimal number as well. Sometimes to make it clear which number system is being used, the

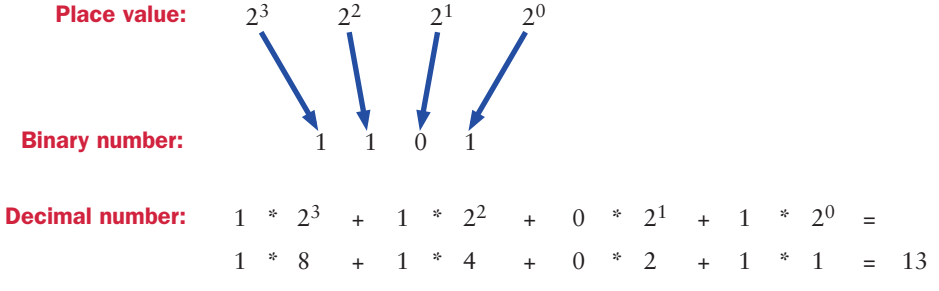

FIGURE B.2 Place values in the binary system

base value is appended as a subscript to the end of a number. Therefore you can distinguish between 1101_2, which is equivalent to 13 in decimal, and 1101_{10} (one thousand, one hundred and one), which in binary is represented as 10001001101_2.

A number system with base *N* has *N* digits (0 through *N*–1). As we have seen, the decimal system has 10 digits (0 through 9), and the binary system has two digits (0 and 1). They all work the same way. For instance, the base-5 number system has five digits (0 to 4).

Note that, in any number system, the place value of the digit furthest to the right is 1, since any base raised to the zero power is 1. Also notice that the value 10, which we refer to as "ten" in the decimal system, always represents the base value in any number system. In base 10, 10 is one 10 and zero 1's. In base 2, 10 is one 2 and zero 1's. In base 5, 10 is one 5 and zero 1's.

Bases Higher Than 10

Since all number systems with base *N* have *N* digits, then base 16 has 16 digits. But what are they? We are used to the digits 0 through 9, but in bases higher than 10, we need a single digit, a single symbol, that represents the decimal value 10. In fact, in *base 16,* which is also called *hexadecimal,* we need digits that represent the decimal values 10 through 15.

For number systems higher than 10, we use alphabetic characters as single digits for values greater than 9. The hexadecimal digits are 0 through F, where 0 through 9 represent the first 10 digits, and A represents the decimal value 10, B represents 11, C represents 12, D represents 13, E represents 14, and F represents 15.

Therefore the number 2A8E is a valid hexadecimal number. The place values are determined as they are for decimal and binary, using increasing powers of the base. So in hexadecimal, the place values are powers of 16. Figure B.3 shows how the place values of the hexadecimal number 2A8E contribute to the overall value.

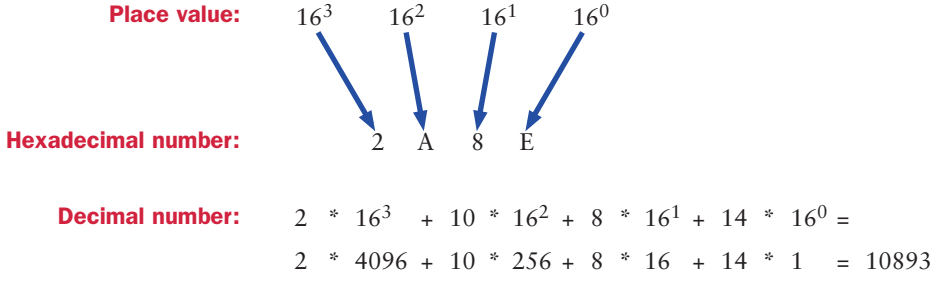

FIGURE B.3 Place values in the hexadecimal system

All number systems with bases greater than 10 use letters as digits. For example, base 12 has the digits 0 through B and base 19 has the digits 0 through I. However, beyond having a different set of digits and a different base, the rules governing each number system are the same.

Keep in mind that when we change number systems, we are simply changing the way we represent values, not the values themselves. If you have 18_{10} pencils, it may be written as 10010 in binary or as 12 in hexadecimal, but it is still the same number of pencils.

Figure B.4 shows the representations of the decimal values 0 through 20 in several bases, including *base 8*, which is also called *octal*. Note that the larger the base, the higher the value that can be represented in a single digit.

Binary (base 2)	Octal (base 8)	Decimal (base 10)	Hexadecimal (base 16)
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
10000	20	16	10
10001	21	17	11
10010	22	18	12
10011	23	19	13
10100	24	20	14

FIGURE B.4 Counting in various number systems

Conversions

We've already seen how a number in another base is converted to decimal by determining the place value of each digit and computing the result. This process can be used to convert any number in any base to its equivalent value in base 10.

Now let's reverse the process, converting a base-10 value to another base. First, find the highest place value in the new number system that is less than or equal to the original value. Then divide the original number by that place value to determine the digit that belongs in that position. The remainder is the value that must be represented in the remaining digit positions. Continue this process, position by position, until the entire value is represented.

For example, Figure B.5 shows the process of converting the decimal value 180 into binary. The highest place value in binary that is less than or equal to 180 is 128 (or 2^7), which is the eighth bit position from the right. Dividing 180 by 128 yields 1 with 52 remaining. Therefore the first bit is 1, and the decimal value 52 must be represented in the remaining seven bits. Dividing 52 by 64, which is the next place value (2^6), yields 0 with 52 remaining. So the second bit is 0. Dividing 52 by 32 yields 1 with 20 remaining. So the third bit is 1, and the remaining five bits must represent the value 20. Dividing 20 by 16 yields 1 with 4 remaining. Dividing 4 by 8 yields 0 with 4 remaining. Dividing 4 by 4 yields 1 with 0 remaining.

Since the number has been completely represented, the rest of the bits are zero. Therefore 180_{10} is equivalent to 10110100 in binary. This can be confirmed by

Place value	Number	Digit
128	180	1
64	52	0
32	52	1
16	20	1
8	4	0
4	4	1
2	0	0
1	0	0

$$180_{10} = 10110100_2$$

FIGURE B.5 Converting a decimal value into binary

Place value	Number	Digit	
256	1967	7	
16	175	A	$1967_{10} = 7AF_{16}$
1	15	F	

FIGURE B.6 Converting a decimal value into hexadecimal

converting the new binary number back to decimal to make sure we get the original value.

This process works to convert any decimal value to any target base. For each target base, the place values and possible digits change. If you start with the correct place value, each division operation will yield a valid digit in the new base.

In the example in Figure B.5, the only digits that could have resulted from each division operation would have been 1 or 0, since we were converting to binary. However, when we are converting to other bases, any valid digit in the new base could result. For example, Figure B.6 shows the process of converting the decimal value 1967 into hexadecimal.

The place value of 256, which is 16^2, is the highest place value less than or equal to the original number, since the next highest place value is 16^3 or 4096. Dividing 1967 by 256 yields 7 with 175 remaining. Dividing 175 by 16 yields 10 with 15 remaining. Remember that 10 in decimal can be represented as the single digit A in hexadecimal. The 15 remaining can be represented as the digit F. Therefore 1967_{10} is equivalent to 7AF in hexadecimal.

Shortcut Conversions

We have established techniques for converting any value in any base to its equivalent representation in base 10, and from base 10 to any other base. Therefore. you can now convert a number in any base to any other base by going through base 10. However, an interesting relationship exists between the bases that are powers of 2, such as binary, octal, and hexadecimal, which allows very quick conversions between them.

To convert from binary to hexadecimal, for instance, you can simply group the bits of the original value into groups of four, starting from the right, then convert each group of four into a single hexadecimal digit. The example in Figure B.7 demonstrates this process.

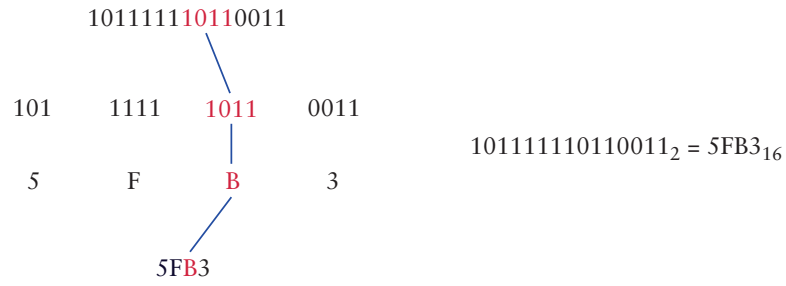

FIGURE B.7 Shortcut conversion from binary to hexadecimal

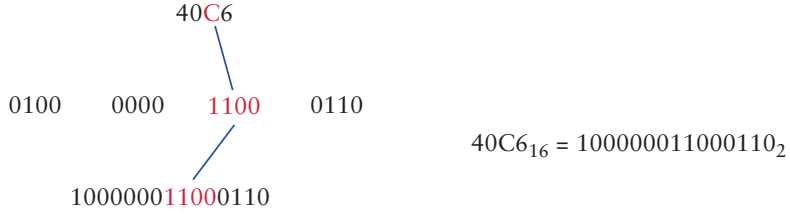

FIGURE B.8 Shortcut conversion from hexadecimal to binary

To go from hexadecimal to binary, we reverse this process, expanding each hexadecimal digit into four binary digits. Note that you may have to add leading zeros to the binary version of each expanded hexadecimal digit if necessary to make four binary digits. Figure B.8 shows the conversion of the hexadecimal value 40C6 to binary.

Why do we section the bits into groups of four when converting from binary to hexadecimal? The shortcut conversions work between binary and any base that is a power of 2. We section the bits into groups of that power. Since $2^4 = 16$, we section the bits in groups of four.

Converting from binary to octal is the same process except that the bits are sectioned into groups of three, since $2^3 = 8$. Likewise, when converting from octal to binary, we expand each octal digit into three bits.

To convert between, say, hexadecimal and octal is now a process of doing two shortcut conversions. First convert from hexadecimal to binary, then take that result and perform a shortcut conversion from binary to octal.

By the way, these types of shortcut conversions can be performed between any base *B* and any base that is a power of *B*. For example, conversions between base 3 and base 9 can be accomplished using the shortcut grouping technique, sectioning or expanding digits into groups of two, since $3^2 = 9$.