Digital Logic and Computer Design

M. MORRIS MANO
Professor of Engineering
California State University, Los Angeles

CHAPTER 1

Binary Systems

1.1 Digital Computers and Digital Systems

Digital computers have made possible many scientific, industrial, and commercial advances that would have been unattainable otherwise. Our space program would have been impossible without real-time, continuous computer monitoring, and many business enterprises function efficiently only with the aid of automatic data processing. Computers are used in scientific calculations, commercial and business data processing, air traffic control, space guidance, the educational field, and many other areas. The most striking property of a digital computer is its generality. It can follow a sequence of instructions, called a *program*, that operates on given data. The user can specify and change programs and/or data according to the specific need. As a result of this flexibility, general-purpose digital computers can perform a wide variety of information-processing tasks.

The general-purpose digital computer is the best-known example of a digital system. Other examples include telephone switching exchanges, digital voltmeters, frequency counters, calculating machines, and teletype machines. Characteristic of a digital system is its manipulation of *discrete elements* of information. Such discrete elements may be electric impulses, the decimal digits, the letters of an alphabet, arithmetic operations, punctuation marks, or any other set of meaningful symbols. The juxtaposition of discrete elements of information represents a quantity of information. For example, the letters *d*, *o*, and *g* form the word *dog*. The digits 237 form a number. Thus, a sequence of discrete elements forms a language, that is, a discipline that conveys information. Early digital computers were used mostly for numerical computations. In this case the discrete elements used are the digits. From this application, the term *digital computer* has emerged. A more appropriate name for a digital computer would be a "discrete information processing system."

Discrete elements of information are represented in a digital system by physical quantities called *signals*. Electrical signals such as voltages and currents are the most common. The signals in all present-day electronic digital systems have only two discrete values and are said to be *binary*. The digital-system designer is restricted to the use of binary signals because of the lower reliability of many-valued electronic circuits. In other words, a circuit with ten states, using one discrete voltage value for each state, can be designed, but it would possess a very low reliability of operation. In contrast, a transistor circuit that is either on or off has two possible signal values and can be constructed to be extremely reliable. Because of this physical restriction of components, and because human logic tends to be binary, digital systems that are constrained to take discrete values are further constrained to take binary values.

Discrete quantities of information emerge either from the nature of the process or may be purposely quantized from a continuous process. For example, a payroll schedule is an inherently

discrete process that contains employee names, social security numbers, weekly salaries, income taxes, etc. An employee's paycheck is processed using discrete data values such as letters of the alphabet (names), digits (salary), and special symbols such as $. On the other hand, a research scientist may observe a continuous process but record only specific quantities in tabular form. The scientist is thus quantizing his continuous data. Each number in his table is a discrete element of information.

Many physical systems can be described mathematically by differential equations whose solutions as a function of time give the complete mathematical behavior of the process. An *analog computer* performs a direct *simulation* of a physical system. Each section of the computer is the analog of some particular portion of the process under study. The variables in the analog computer are represented by continous signals, usually electric voltages that vary with time. The signal variables are considered analogous to those of the process and behave in the same manner. Thus measurements of the analog voltage can be substituted for variables of the process. The term *analog signal* is sometimes substituted for *continuous signal* because "analog computer" has come to mean a computer that manipulates continuous variables.

To simulate a physical process in a digital computer, the quantities must be quantized. When the variables of the process are presented by real-time continuous signals, the latter are quantized by an analog-to-digital conversion device. A physical system whose behavior is described by mathematical equations is simulated in a digital computer by means of numerical methods. When the problem to be processed is inherently discrete, as in commercial applications, the digital computer manipulates the variables in their natural form.

A block diagram of the digital computer is shown in Fig. 1.1. The memory unit stores programs as well as input, output, and intermediate data. The processor unit performs arithmetic and other data-processing tasks as specified by a program. The control unit supervises the flow of information between the various units. The control unit retrieves the instructions, one by one, from the program which is stored in memory. For each instruction, the control unit informs the processor to execute the operation specified by the instruction. Both program and data are stored in memory. The control unit supervises the program instructions, and the processor manipulates the data as specified by the program.

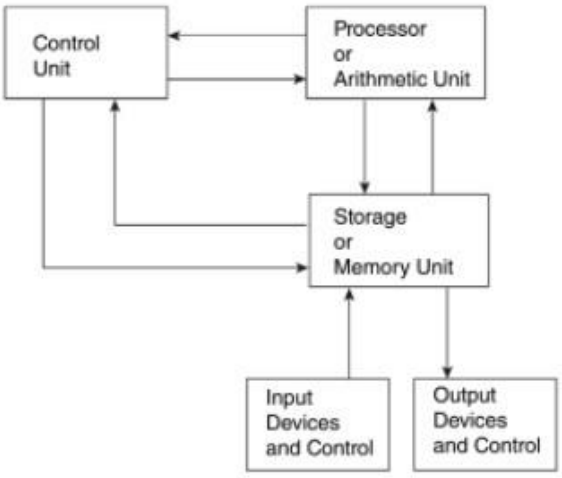


Figure 1.1 Block diagram of a digital computer

The program and data prepared by the user are transferred into the memory unit by means of an input device such as punch-card reader or a teletypewriter. An output device, such as a printer, receives the result of the computations and the printed results are presented to the user. The input and output devices are special digital systems driven by electromechanical parts and controlled by electronic digital circuits.

An electronic calculator is a digital system similar to a digital computer, with the input device being a keyboard and the output device a numerical display. Instructions are entered in the calculator by means of the function keys, such as plus and minus. Data are entered through the numeric keys. Results are displayed directly in numeric form. Some calculators come close to resembling a digital computer by having printing capabilities and programmable facilities. A digital computer, however, is a more powerful device than a calculator. A digital computer can accommodate many other input and output devices; it can perform not only arithmetic computations but logical operations as well and can be programmed to make decisions based on internal and external conditions.

A digital computer is an interconnection of digital modules. To understand the operation of each digital module, it is necessary to have a basic knowledge of digital systems and their general behavior. The first half of this book deals with digital systems in general to provide the background necessary for their design. The second half of the book discusses the various modules of the digital computer, their operation and design. The operational characteristics of the memory unit are explained in Chapter 7. The organization and design of the processor unit is undertaken in Chapter 9. Various methods for designing the control unit are introduced in Chapter 10. The organization and design of a small, complete digital computer is presented in Chapter 11.

A processor, when combined with the control unit, forms a component referred to as a *central processor unit* or CPU. A CPU enclosed in small integrated-circuit package is called a *microprocessor.* The memory unit, as well as the part that controls the interface between the microprocessor and the input and output devices, may be enclosed within the microprocessor package or may be available in other small integrated-circuit packages. A CPU combined with memory and interface control to form a small-size computer is called a *microcomputer.* The availability of microcomputer components has revolutionized the digital system design technology, giving the designer the freedom to create structures that were previously uneconomical. The various components of a microcomputer system are presented in Chapter 12.

It has already been mentioned that a digital computer manipulates discrete elements of information and that these elements are represented in the binary form. Operands used for calculations may be expressed in the binary number system. Other discrete elements, including the decimal digits, are represented in binary codes. Data processing is carried out by means of binary logic elements using binary signals. Quantities are stored in binary storage elements. The purpose of this chapter is to introduce the various binary concepts as a frame of reference for further detailed study in the suceeding chapters.

1.2 Binary Numbers

A decimal number such as 7392 represents a quantity equal to 7 thousands plus 3 hundreds, plus 9 tens, plus 2 units. The thousands, hundreds, etc., are powers of 10 implied by the position of the coefficients. To be more exact, 7392 should be written as:

$$7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$$

However, the convention is to write only the coefficients and from their position deduce the necessary powers of 10. In general, a number with a decimal point is represented by a series of coefficients as follows:

$$a_5\, a_4\, a_3\, a_2\, a_1\, a_0.\, a_{-1}\, a_{-2}\, a_{-3}$$

The a_j coefficients are one of the ten digits (0, 1,2,...., 9), and the subscript value j gives the place value and, hence, the power of 10 by which the coefficient must be multiplied.

$$10^5 a_5 + 10^4 a_4 + 10^3 a_3 + 10^2 a_2 + 10^1 a_1 + 10^0 a_0 + 10^{-1}\, a_{-1} + 10^{-2}\, a_{-2} + 10^{-3}\, a_{-3}$$

The decimal number system is said to be of *base,* or *radix,* 10 because it uses ten digits and the coefficients are multiplied by powers of 10. The *binary* system is a different number system. The coefficients of the binary numbers system have two possible values: 0 and 1. Each coefficient a_j is multiplied by 2^j. For example, the decimal equivalent of the binary number 11010.11 is 26.75, as shown from the multiplication of the coefficients by powers of 2:

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 26.75$$

In general, a number expressed in base-r system has coefficients multiplied by powers of r:

$$a_n . r^n + a_{n-1} . r^{n-1} + \ldots + a_2 . r^2 + a_1 . r + a_0 + a_{-1} . r^{-1} + a_{-2} . r^{-2} + \ldots + a_{-m} . r^{-m}$$

The coefficients a_j range in value from 0 to $r - 1$. To distinguish between numbers of different bases, we enclose the coefficients in parentheses and write a subscript equal to the base used (except sometimes for decimal numbers, where the content makes it obvious that it is decimal). An example of a base-5 number is:

$$(4021.2)_5 = 4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1} = (511.4)_{10}$$

Note that coefficient values for base 5 can be only 0, 1,2, 3, and 4.

It is customary to borrow the needed r digits for the coefficients from the decimal system when the base of the number is less than 10. The letters of the alphabet are used to supplement the ten decimal digits when the base of the number is greater than 10. For example, in the *hexa-decimal* (base 16) number system, the first ten digits are borrowed from the decimal system. The letters A, B, C, D, E, and F are used for digits 10, 11, 12, 13, 14, and 15, respectively. An example of a hexadecimal number is:

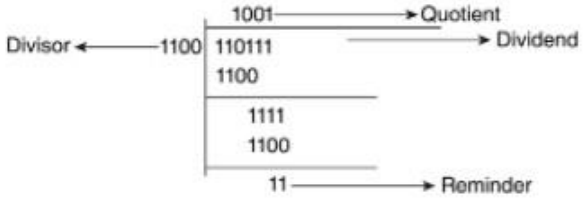
$$(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16 + 15 = (46687)_{10}$$

The first 16 numbers in the decimal, binary, octal, and hexadecimal systems are listed in Table 1-1.

Arithmetic operations with numbers in base r follow the same rules as for decimal numbers. When other than the familiar base 10 is used, one must be careful to use only the r allowable digits. Examples of addition, subtraction, and multiplication of two binary numbers are shown below:

Table 1-1 Numbers with different bases

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	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



augend:	101101	minuend:	101101	multiplicand:	1011
addend:	+100111	subtrahend:	−100111	multiplier:	× 101
sum:	1010100	difference:	000110		1011
					0000
					1011
				product:	110111

The sum of two binary numbers is calculated by the same rules as in decimal, except that the digits of the sum in any significant position can be only 0 or 1. Any "carry" obtained in a given significant position is used by the pair of digits one significant position higher. The subtraction is slightly more complicated. The rules are still the same as in decimal, except that the "borrow" in a given significant position adds 2 to a minuend digit. (A borrow in the decimal

system adds 10 to a minuend digit.) Multiplication is very simple. The multiplier digits are always 1 or 0. Therefore, the partial products are equal either to the multiplicand or to 0.

1.3 Number Base Conversions

A binary number can be converted to decimal by forming the sum of the powers of 2 of those coefficients whose value is 1. For example:

$$(10l0.0ll)_2 = 2^3 + 2^1 + 2^{-2} + 2^{-3} = (10.375)_{10}$$

The binary number has four l's and the decimal equivalent is found from the sum of four powers of 2. Similarly, a number expressed in base r can be converted to its decimal equivalent by multiplying each coefficient with the corresponding power of r and adding. The following is an example of octal-to-decimal conversion:

$$(630.4)_8 = 6 \times 8^2 + 3 \times 8 + 4 \times 8^{-1} = (408.5)_{10}$$

The conversion from decimal to binary or to any other base-r system is more convenient if the number is separated into an *integer part* and a *fraction part* and the conversion of each part done separately. The conversion of an *integer* from decimal to binary is best explained by example.

EXAMPLE 1-1: Convert decimal 41 to binary. First, 41 is divided by 2 to give an integer quotient of 20 and a remainder of $\frac{1}{2}$. The quotient is again divided by 2 to give a new quotient and remainder. This process is continued until the integer quotient becomes 0. The *coefficients* of the desired binary number are obtained from the *remainders* as follows:

integer quotient		*remainder*	*coefficient*
$\frac{41}{2} = 20$	+	$\frac{1}{2}$	$a_0 = 1$
$\frac{20}{2} = 10$	+	0	$a_1 = 0$
$\frac{10}{2} = 5$	+	0	$a_2 = 0$
$\frac{5}{2} = 2$	+	$\frac{1}{2}$	$a_3 = 1$
$\frac{2}{2} = 1$	+	0	$a_4 = 0$
$\frac{1}{2} = 0$	+	$\frac{1}{2}$	$a_5 = 1$

answer: $(41)_{10} = (a_5 a_4 a_3 a_2 a_1 a_0)_2 = (101001)_2$

The arithmetic process can be manipulated more conveniently as follows:

integer	*remainder*	
41		
20	1	
10	0	
5	0	↑
2	1	|
1	0	|
0	1	└ 101001 = answer

The conversion from decimal integers to any base-r system is similar to the above example, except that division is done by r instead of 2.

EXAMPLE 1-2: Convert decimal 153 to octal. The required base r is 8. First, 153 is divided by 8 to give an integer quotient of 19 and a remainder of 1. Then 19 is divided by 8 to give an integer quotient of 2 and a remainder of 3. Finally, 2 is divided by 8 to give a quotient of 0 and a remainder of 2. This process can be conveniently manipulated as follows:

153	
19	1
2	3 ↑
0	2 └ $= (231)_8$

The conversion of a decimal *fraction* to binary is accomplished by a method similar to that used for integers. However, multiplication is used instead of division, and integers are accumulated instead of remainders. Again, the method is best explained by example.

EXAMPLE 1-3: Convert $(0.6875)_{10}$ to binary. First, 0.6875 is multiplied by 2 to give an integer and a fraction. The new fraction is multiplied by 2 to give a new integer and a new fraction. This process is continued until the fraction becomes 0 or until the number of digits have sufficient accuracy. The coefficients of the binary number are obtained from the integers as follows:

	integer		*fraction*	*coefficient*
$0.6875 \times 2 =$	1	+	0.3750	$a_{-1} = 1$
$0.3750 \times 2 =$	0	+	0.7500	$a_{-2} = 0$
$0.7500 \times 2 =$	1	+	0.5000	$a_{-3} = 1$
$0.5000 \times 2 =$	1	+	0.0000	$a_{-4} = 1$

answer: $(0.6875)_{10} = (0.a_{-1}\, a_{-2} a_{-3}\, a_{-4})_2 = (0.1011)_2$

To convert a decimal fraction to a number expressed in base r, a similar procedure is used. Multiplication is by r instead of 2, and the coefficients found from the integers may range in value from 0 to $r - 1$ instead of 0 and 1.

EXAMPLE 1-4: Convert $(0.513)_{10}$ to octal.

$$0.513 \times 8 = 4.104$$
$$0.104 \times 8 = 0.832$$
$$0.832 \times 8 = 6.656$$
$$0.656 \times 8 = 5,248$$
$$0.248 \times 8 = 1.984$$
$$0.984 \times 8 = 7.872$$

The answer, to seven significant figures, is obtained from the integer part of the products:

$$(0.513)_{10} = (0.406517 \ldots)_8$$

The conversion of decimal numbers with both integer and fraction parts is done by converting the integer and fraction separately and then combining the two answers together. Using the results of Examples 1-1 and 1-3, we obtain:

$$(41.6875)_{10} - (101001.1011)_2$$

From Examples 1-2 and 1-4, we have:

$$(153.513)_{10} = (231.406517)_8$$

1.4 Octal and Hexadecimal Numbers

The conversion from and to binary, octal, and hexadecimal plays an important part in digital computers. Since $2^3 = 8$ and $2^4 = 16$, each octal digit corresponds to three binary digits and each hexadecimal digit corresponds to four binary digits. The conversion from binary to octal is easily accomplished by partitioning the binary number into groups of three digits each, starting from the binary point and proceeding to the left and to the right. The corresponding octal digit is then assigned to each group. The following example illustrates the procedure:

$$\left(\underbrace{10}_{2} \underbrace{110}_{6} \underbrace{001}_{1} \underbrace{101}_{5} \underbrace{011}_{3} \cdot \underbrace{111}_{7} \underbrace{100}_{4} \underbrace{000}_{0} \underbrace{110}_{6} \right)_2 = (26153.7406)_8$$

Conversion from binary to hexadecimal is similar, except that the binary number is divided into groups of four digits:

$$\left(\underbrace{10}_{2} \underbrace{1100}_{C} \underbrace{0110}_{6} \underbrace{1011}_{B} \cdot \underbrace{1111}_{F} \underbrace{0010}_{2} \right)_2 = (2C6B.F2)_{16}$$

The corresponding hexadecimal (or octal) digit for each group of binary digits is easily remembered after studying the values listed in Table 1-1.

Conversion from octal or hexadecimal to binary is done by a procedure reverse to the above. Each octal digit is converted to its three-digit binary equivalent. Similarly, each hexadecimal digit is converted to its four-digit binary equivalent. This is illustrated in the following examples:

$$(673.124)_8 = \left(\underbrace{110}_{6}\ \underbrace{111}_{7}\ \underbrace{011}_{3} \cdot \underbrace{001}_{1}\ \underbrace{010}_{2}\ \underbrace{100}_{4}\right)_2$$

$$(306.\ D)_{16} = \left(\underbrace{0011}_{3}\ \underbrace{0000}_{0}\ \underbrace{0110}_{6} \cdot \underbrace{1101}_{D}\right)_2$$

Binary numbers are difficult to work with because they require three or four times as many digits as their decimal equivalent. For example, the binary number 111111111111 is equivalent to decimal 4095. However, digital computers use binary numbers and it is sometimes necessary for the human operator or user to communicate directly with the machine by means of binary numbers. One scheme that retains the binary system in the computer but reduces the number of digits the human must consider utilizes the relationship between the binary number system and the octal or hexadecimal system. By this method, the human thinks in terms of octal or hexadecimal numbers and performs the required conversion by inspection when direct communication with the machine is necessary. Thus the binary number 111111111111 has 12 digits and is expressed in octal as 7777 (four digits) or in hexadecimal as FFF (three digits). During communication between people (about binary numbers in the computer), the octal or hexadecimal representation is more desirable because it can be expressed more compactly with a third or a quarter of the number of digits required for the equivalent binary number. When the human communicates with the machine (through console switches or indicator lights or by means of programs written in *machine language*), the conversion from octal or hexadecimal to binary and vice versa is done by inspection by the human user.

1.5 Complements

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulations. There are two types of complements for each base-r system: (1) the r's complement and (2) the $(r-1)$'s complement. When the value of the base is substituted, the two types receive the names 2's and 1's complement for binary numbers, or 10's and 9's complement for decimal numbers.

1.5.1 The r's Complement

Given a positive number N in base r with an integer part of n digits, the r's complement of N is defined as $r^n - N$ for $N \neq 0$ and 0 for $N = 0$. The following numerical example will help clarify the definition.

The 10's complement of $(52520)_{10}$ is $10^5 - 52520 = 47480$.

The number of digits in the number is $n = 5$.

The 10's complement or $(0.3267)_{10}$ is $1 - 0.3267 = 0.6733$.