DIGITAL LOGIC DESIGN

Lecture 4

Gray Code

- The output data of many physical systems are quantities that are continuous.
- These data must be converted into digital form before they are applied to a digital system.
- Continuous or analog information is converted into digital form by means of an analog- to-digital converter.
- It is sometimes convenient to use the Gray code shown in Table 1.6
- The advantage of the Gray code over the straight binary number sequence is that only one bit in the code group changes in going from one number to the next.
- For example, in going from 7 to 8, the Gray code changes from 0100 to 1100.
- Only the first bit changes, from 0 to 1; the other three bits remain the same.
- By contrast, with binary numbers the change from 7 to 8 will be from 0111 to 1000, which causes all four bits to change values.

Gray Code

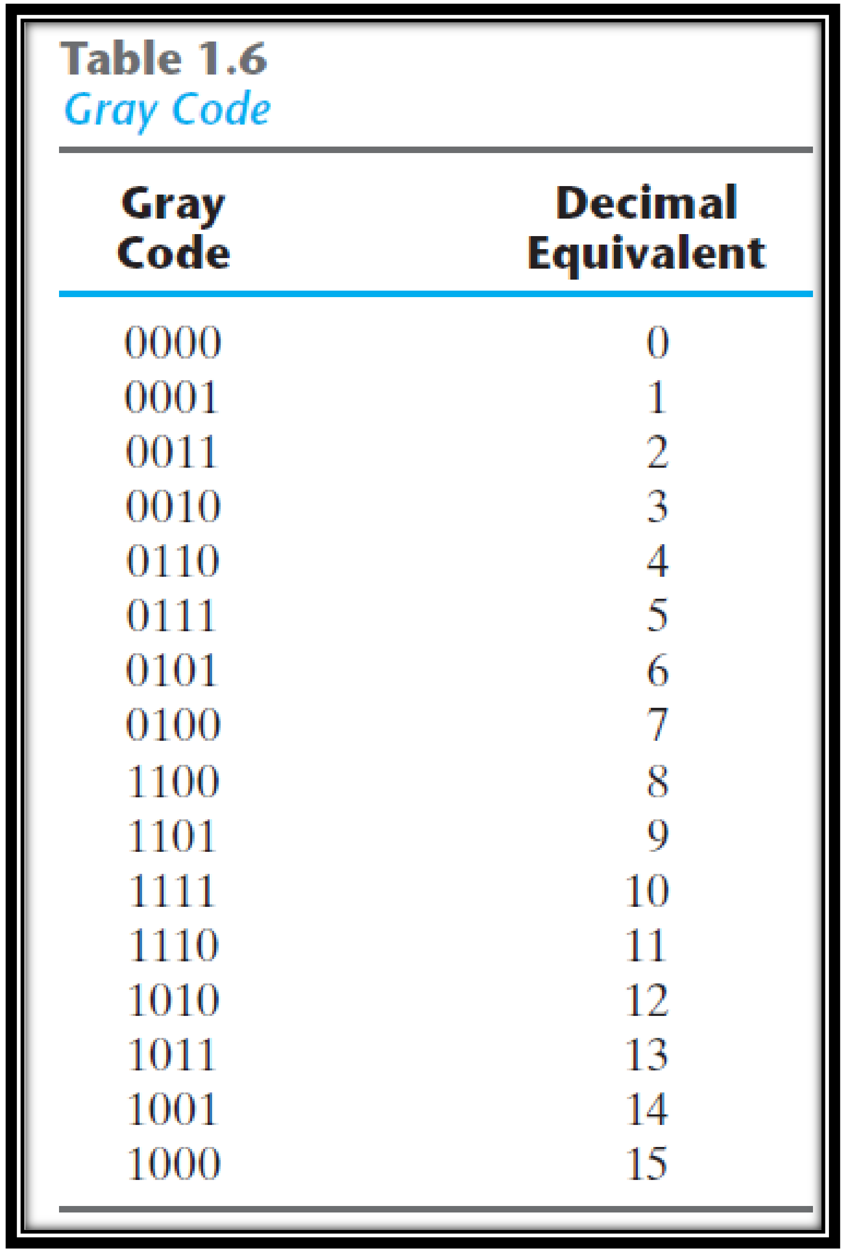


Table 1.6
Gray Code

Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

Gray Code

- The code is called reflected because it can be generated in the following manner.
- Take the Gray code 0, 1. Write it forwards, then backwards: 0, 1, 1, 0. Then prepend 0s to the first half and 1s to the second half: 00, 01, 11, 10.
- Continuing, write 00, 01, 11, 10, 10, 11, 01, 00 to obtain: 000, 001, 011, 010, 110, 111, 101, 100,….
- Each iteration therefore doubles the number of codes

Gray Code

- The Gray code is used in applications in which the normal sequence of binary numbers generated by the hardware may produce an error or ambiguity during the transition from one number to the next.
- If binary numbers are used, a change, for example, from 0111 to 1000 may produce an intermediate erroneous number 1001 if the value of the rightmost bit takes longer to change than do the values of the other three bits.
- This could have serious consequences for the machine using the information.
- The Gray code eliminates this problem, since only one bit changes its value during any transition between two numbers.
- A typical application of the Gray code is the representation of analog data by a continuous change in the angular position of a shaft.

Gray Code

- Many devices indicate position by closing and opening switches.
- If that device uses natural binary codes, positions 3 and 4 are next to each other but all three bits of the binary representation differ.
- The problem with natural binary codes is that physical switches are not ideal: it is very unlikely that physical switches will change states exactly in synchrony.
- In the transition between the two states shown above, all three switches change state.
- In the brief period while all are changing, the switches will read some false position.
- Even without key-bounce, the transition might look like 011 — 001 — 101 — 100.

Gray Code

- When the switches appear to be in position 001, the observer cannot tell if that is the "real" position 001, or a transitional state between two other positions.
- If the output feeds into a sequential system, possibly via combinational logic, then the sequential system may store a false value.
- The reflected binary code solves this problem by changing only one switch at a time, so there is never any ambiguity of position.

ASCII Character Code

- Many applications of digital computers require the handling not only of numbers, but also of other characters or symbols, such as the letters of the alphabet.
- For instance, consider a high-tech company with thousands of employees.
- To represent the names and other relevant information, it is necessary to formulate a binary code for the letters of the alphabet.
- In addition, the same binary code must represent numerals and special characters (such as $).
- An alphanumeric character set is a set of elements that includes the 10 decimal digits, the 26 letters of the alphabet, and a number of special characters.

ASCII Character Code

- Such a set contains between 36 and 64 elements if only capital letters are included, or between 64 and 128 elements if both uppercase and lowercase letters are included.
- In the first case, we need a binary code of six bits, and in the second, we need a binary code of seven bits.
- The standard binary code for the alphanumeric characters is the American Standard Code for Information Interchange (ASCII), which uses seven bits to code 128 characters, as shown in Table 1.7 .
- The seven bits of the code are designated by *b*1 through b_7, with b_7 the most significant bit.
- The letter *A*, for example, is represented in ASCII as1000001.

ASCII Character Code

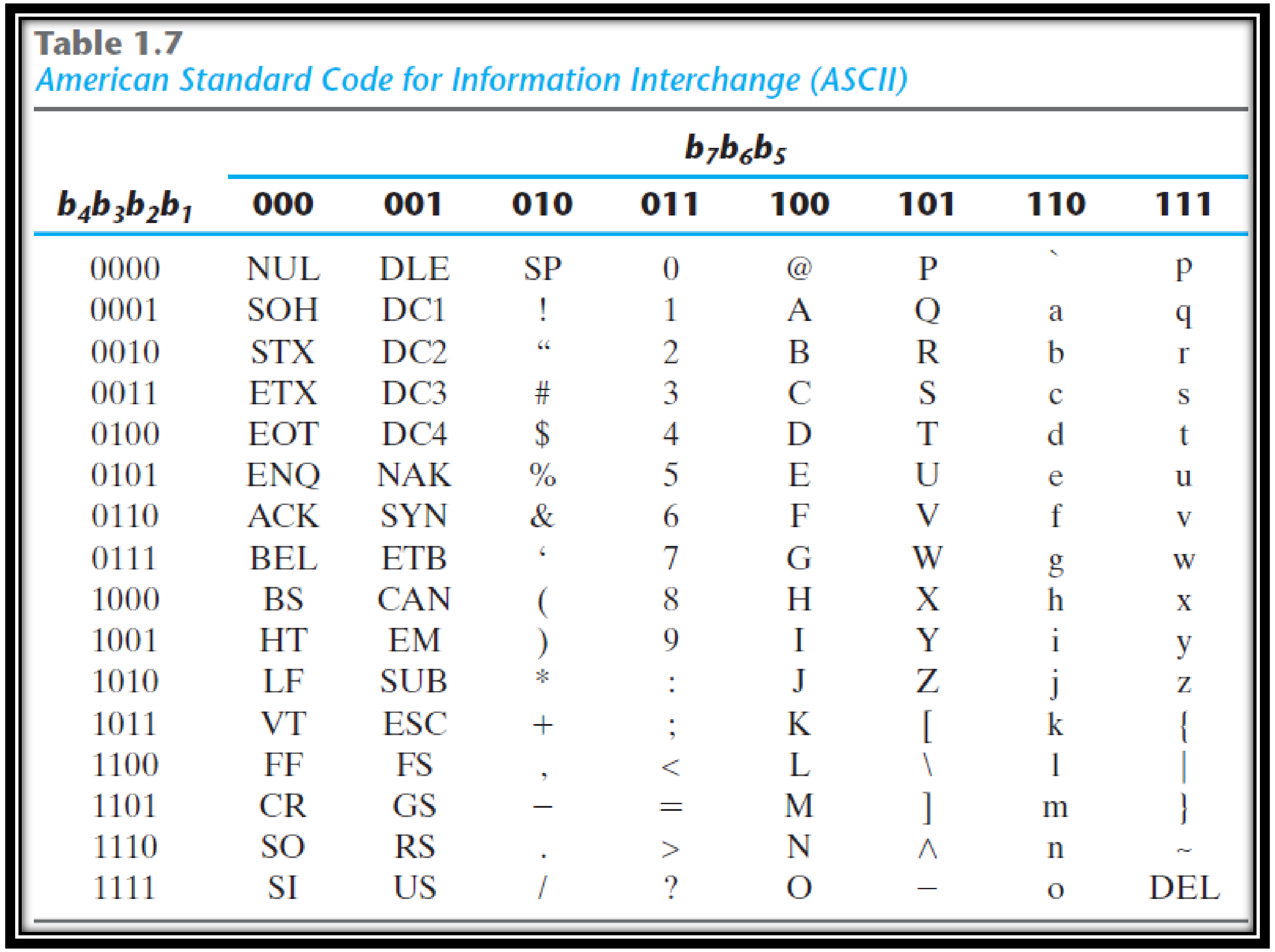


Table 1.7
American Standard Code for Information Interchange (ASCII)

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	∧	n	~
1111	SI	US	/	?	O	–	o	DEL

ASCII Character Code

- The ASCII code also contains 94 graphic characters that can be printed and 34 nonprinting characters used for various control functions.
- The graphic characters consist of the 26 uppercase letters (A through Z), the 26 lowercase letters (a through z), the 10 numerals (0 through 9), and 32 special printable characters, such as %, *, and $.

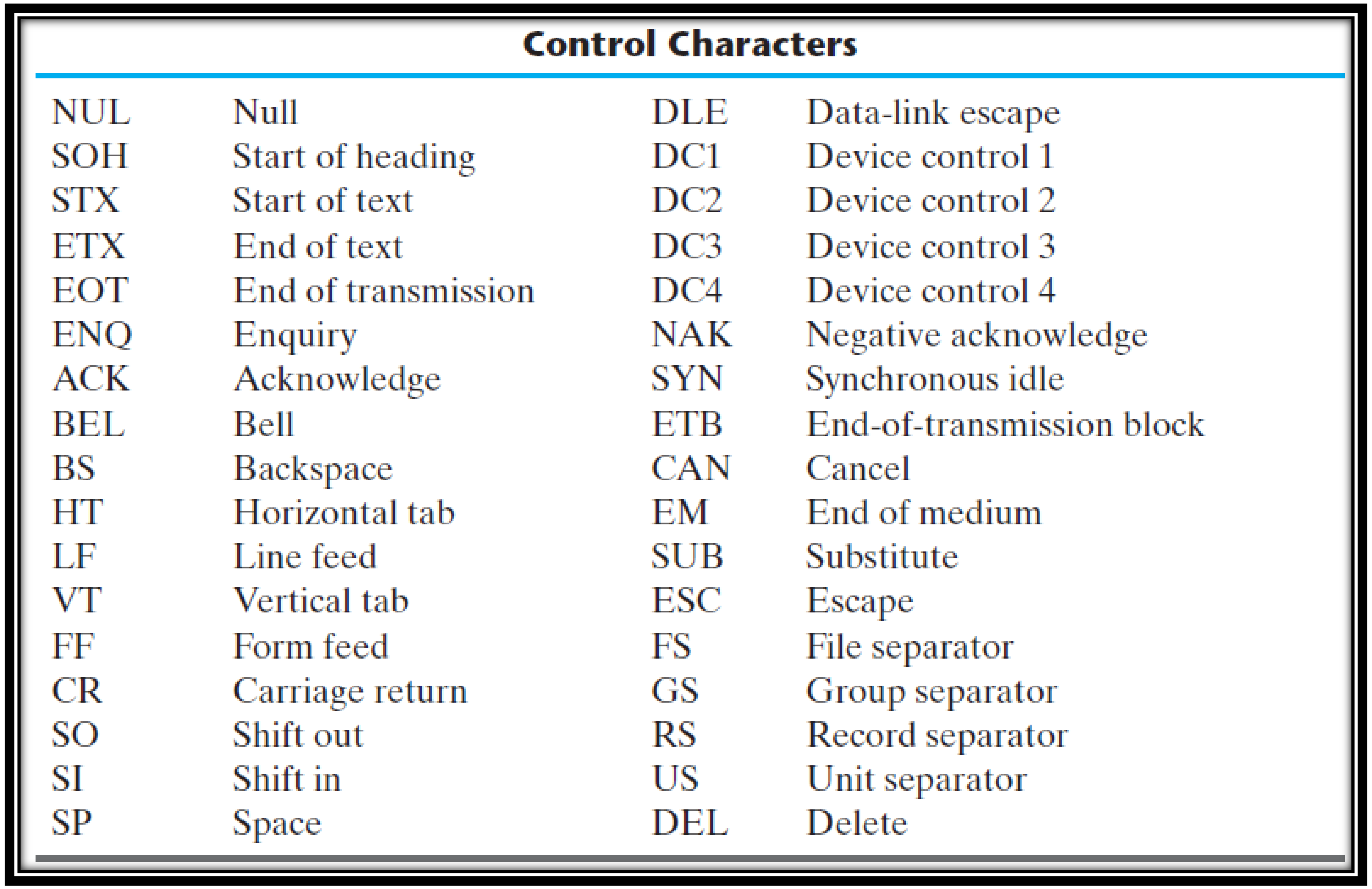
ASCII Character Code

- The 34 control characters are designated in the ASCII table with abbreviated names.
- The control characters are used for routing data and arranging the printed text into a prescribed format.
- There are three types of control characters: **format effectors, information separators, and communication-control characters.**
- Format effectors are characters that control the layout of printing.
- They include the familiar word processor and typewriter controls such as backspace (BS), horizontal tabulation (HT), and carriage return (CR).
- Information separators are used to separate the data into divisions such as paragraphs and pages.
- They include characters such as record separator (RS) and file separator (FS).

ASCII Character Code

- The communication-control characters are useful during the transmission of text between remote devices so that it can be distinguished from other messages using the same communication channel before it and after it.
- Examples of communication-control characters are STX (start of text) and ETX (end of text).
- ASCII is a seven-bit code, but most computers manipulate an eight-bit quantity as a single unit called a *byte*.
- Therefore, ASCII characters most often are stored one per byte. The extra bit is sometimes used for other purposes, depending on the application.
- For example, some printers recognize eight-bit ASCII characters with the most significant bit set to 0.
- An additional 128 eight-bit characters with the most significant bit set to 1 are used for other symbols, such as the Greek alphabet or italic type font.

ASCII Character Code



Control Characters

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

Error-Detecting Code

- To detect errors in data communication and processing, an eighth bit is sometimes added to the ASCII character to indicate its parity.
- A *parity bit* is an extra bit included with a message to make the total number of 1’s either even or odd. Consider the following two characters and their even and odd parity:

	With even parity	**With odd parity**
ASCII A = 1000001	01000001	11000001
ASCII T = 1010100	11010100	01010100

Error-Detecting Code

- In each case, we insert an extra bit in the leftmost position of the code to produce an even number of 1’s in the character for even parity or an odd number of 1’s in the character for odd parity. In general, one or the other parity is adopted, with even parity being more common.
- The parity bit is helpful in detecting errors during the transmission of information from one location to another.
- This function is handled by generating an even parity bit at the sending end for each character.
- The eight-bit characters that include parity bits are transmitted to their destination.
- The parity of each character is then checked at the receiving end.
- If the parity of the received character is not even, then at least one bit has changed value during the transmission.

Error-Detecting Code

- This method detects one, three, or any odd combination of errors in each character that is transmitted.
- An even combination of errors, however, goes undetected, and additional error detection codes may be needed to take care of that possibility.
- What is done after an error is detected depends on the particular application.
- One possibility is to request retransmission of the message on the assumption that the error was random and will not occur again.
- Thus, if the receiver detects a parity error, it sends back the ASCII NAK (negative acknowledge) control character consisting of an evenparity eight bits 10010101.
- If no error is detected, the receiver sends back an ACK (acknowledge) control character, namely, 00000110.

BINARY STORAGE AND REGISTERS

- The binary information in a digital computer must have a physical existence in some medium for storing individual bits.
- A *binary cell* is a device that possesses two stable states and is capable of storing one bit (0 or 1) of information.
- The input to the cell receives excitation signals that set it to one of the two states.
- The output of the cell is a physical quantity that distinguishes between the two states.
- The information stored in a cell is 1 when the cell is in one stable state and 0 when the cell is in the other stable state.

BINARY STORAGE AND REGISTERS

- A *register* is a group of binary cells.
- A register with *n* cells can store any discrete quantity of information that contains *n* bits.
- Consider, for example, a 16-bit register with the following binary content:

  1100001111001001
- A register with 16 cells can be in one of 2^{16} possible states.
- If one assumes that the content of the register represents a binary integer, then the register can store any binary number from 0 to $2^{16} - 1$.

BINARY STORAGE AND REGISTERS

- A digital system is characterized by its registers and the components that perform data processing.
- In digital systems, a ***register transfer*** operation is a basic operation that consists of a transfer of binary information from one set of registers into another set of registers.
- The transfer may be direct, from one register to another, or may pass through data-processing circuits to perform an operation.

BINARY STORAGE AND REGISTERS

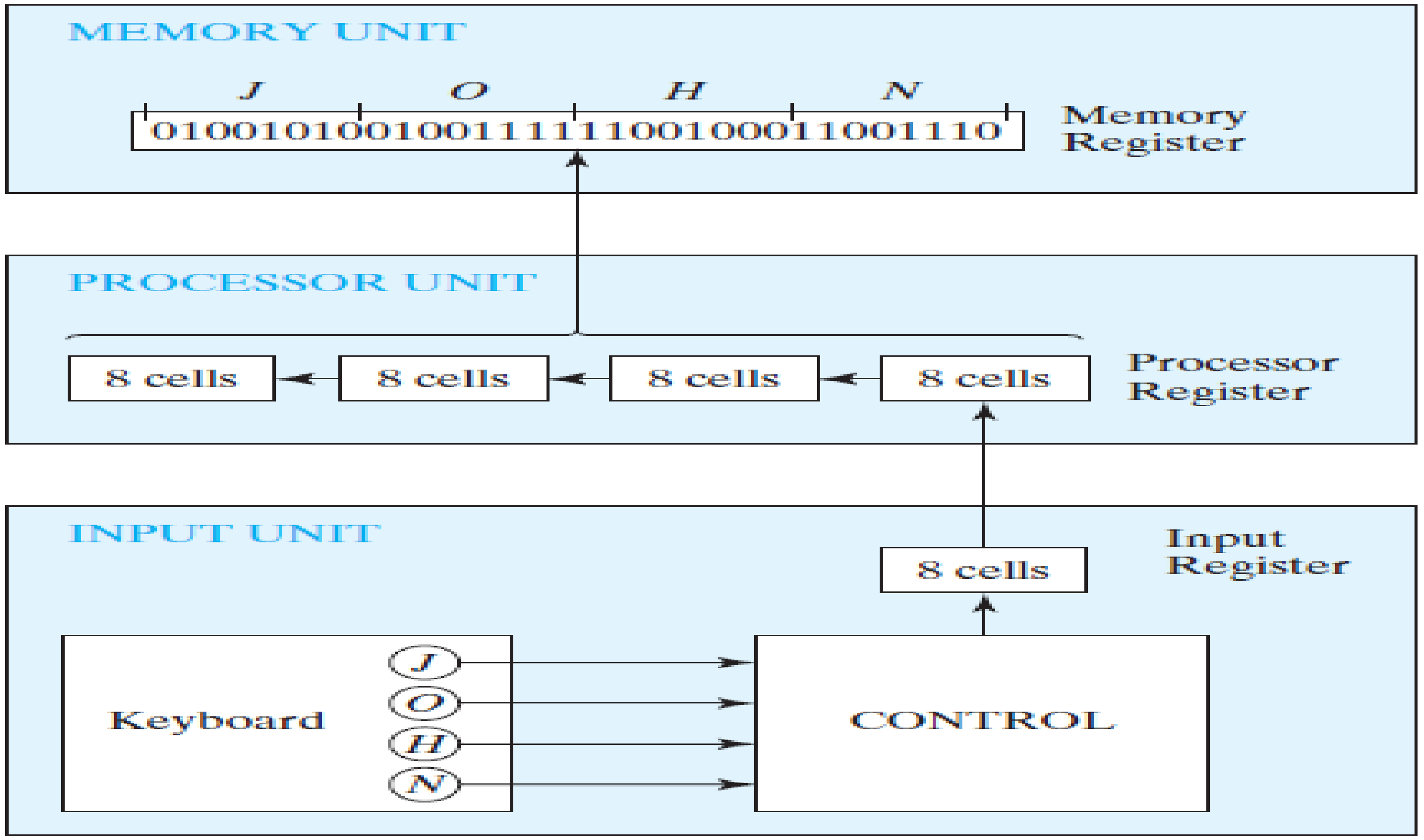


FIGURE 1.1
Transfer of information among registers

BINARY STORAGE AND REGISTERS

- To process discrete quantities of information in binary form, a computer must be provided with devices that hold the data to be processed and with circuit elements that manipulate individual bits of information.
- **The device most commonly used for holding data is a register.**
- Binary variables are manipulated by means of digital logic circuits.

BINARY STORAGE AND REGISTERS

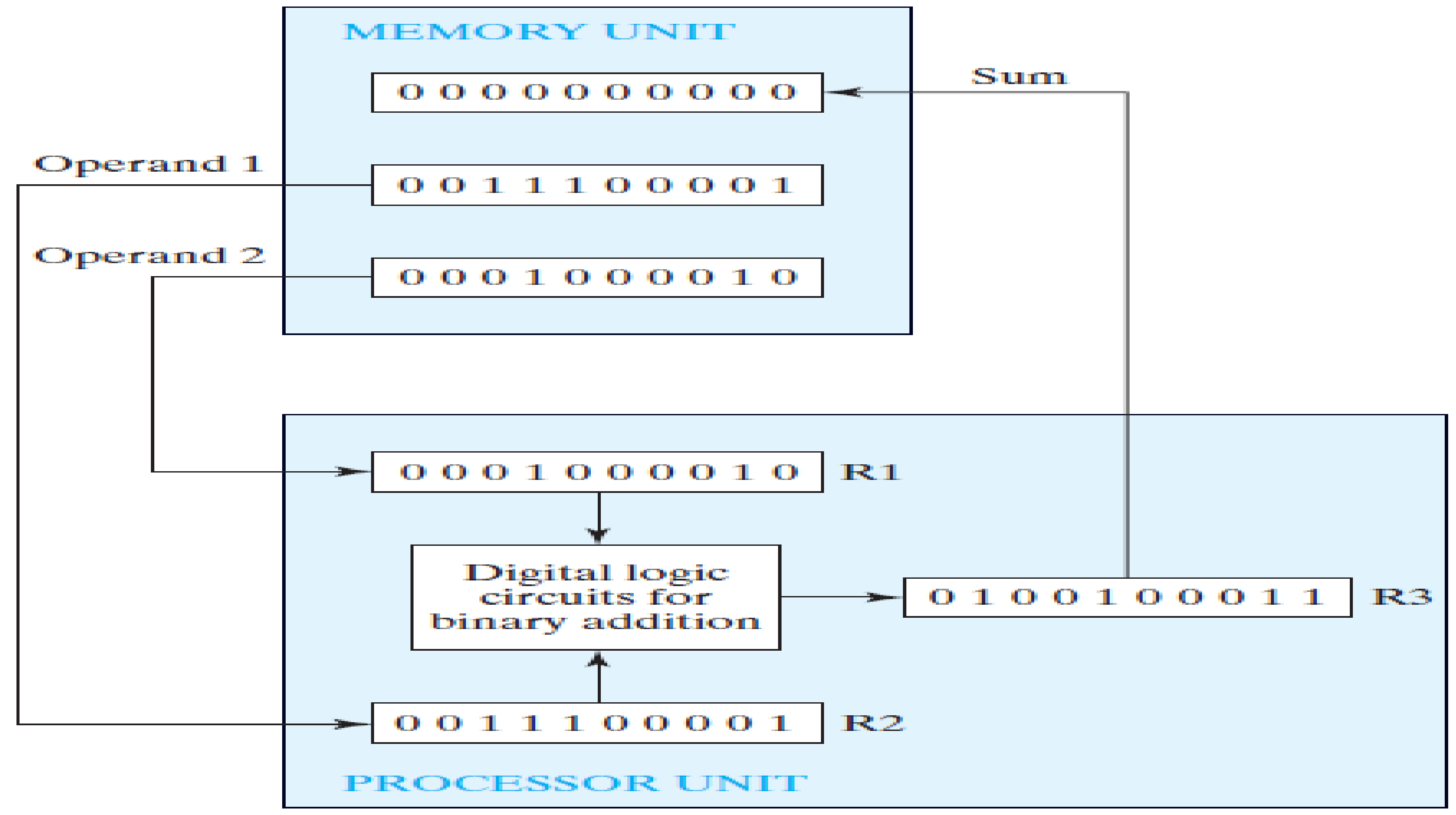


FIGURE 1.2
Example of binary information processing

BINARY LOGIC

- Binary logic deals with variables that take on two discrete values and with operations that assume logical meaning.
- The two values the variables assume may be called by different names (*true* and *false*, *yes* and *no*, etc.), but for our purpose, it is convenient to think in terms of bits and assign the values 1 and 0.
- The binary logic introduced in this section is equivalent to an algebra called Boolean algebra.
- The purpose of this section is to introduce Boolean algebra in a heuristic manner and relate it to digital logic circuits and binary signals.

BINARY LOGIC

- Binary logic consists of binary variables and a set of logical operations.
- The variables are designated by letters of the alphabet, such as A, B, C, x, y, z, etc., with each variable having two and only two distinct possible values: 1 and 0.
- There are three basic logical operations: AND, OR, and NOT. Each operation produces a binary result, denoted by z.
- **1. AND:** This operation is represented by a dot or by the absence of an operator. For example, $x \cdot y = z$ or $xy = z$ is read "x AND y is equal to z." The logical operation AND is interpreted to mean that $z = 1$ if and only if $x = 1$ and $y = 1$; otherwise $z = 0$. (Remember that x, y, and z are binary variables and can be equal either to 1 or 0, and nothing else.) The result of the operation $x \cdot y$ is z.

BINARY LOGIC

- **2. OR:** This operation is represented by a plus sign. For example, $x + y = z$ is read “x OR y is equal to z,” meaning that $z = 1$ if $x = 1$ or if $y = 1$ or if both $x = 1$ and $y = 1$. If both $x = 0$ and $y = 0$, then $z = 0$.
- **3. NOT:** This operation is represented by a prime (sometimes by an overbar).
- For example, $x' = z$ (or $\overline{X} = z$) is read “not x is equal to z,” meaning that z is what x is not. In other words, if $x = 1$, then $z = 0$, but if $x = 0$, then $z = 1$. The NOT operation is also referred to as the complement operation, since it changes a 1 to 0 and a 0 to 1, i.e., the result of complementing 1 is 0, and vice versa.
- Binary logic resembles binary arithmetic, and the operations AND and OR have similarities to multiplication and addition, respectively.

BINARY LOGIC

- For each combination of the values of *x* and *y*, there is a value of *z* specified by the definition of the logical operation.
- Definitions of logical operations may be listed in a compact form called *truth tables*.
- A truth table is a table of all possible combinations of the variables, showing the relation between the values that the variables may take and the result of the operation.
- The truth tables for the operations AND and OR with variables *x* and *y* are obtained by listing all possible values that the variables may have when combined in pairs.
- For each combination, the result of the operation is then listed in a separate row.
- The truth tables for AND, OR, and NOT are given in Table 1.8 .

BINARY LOGIC

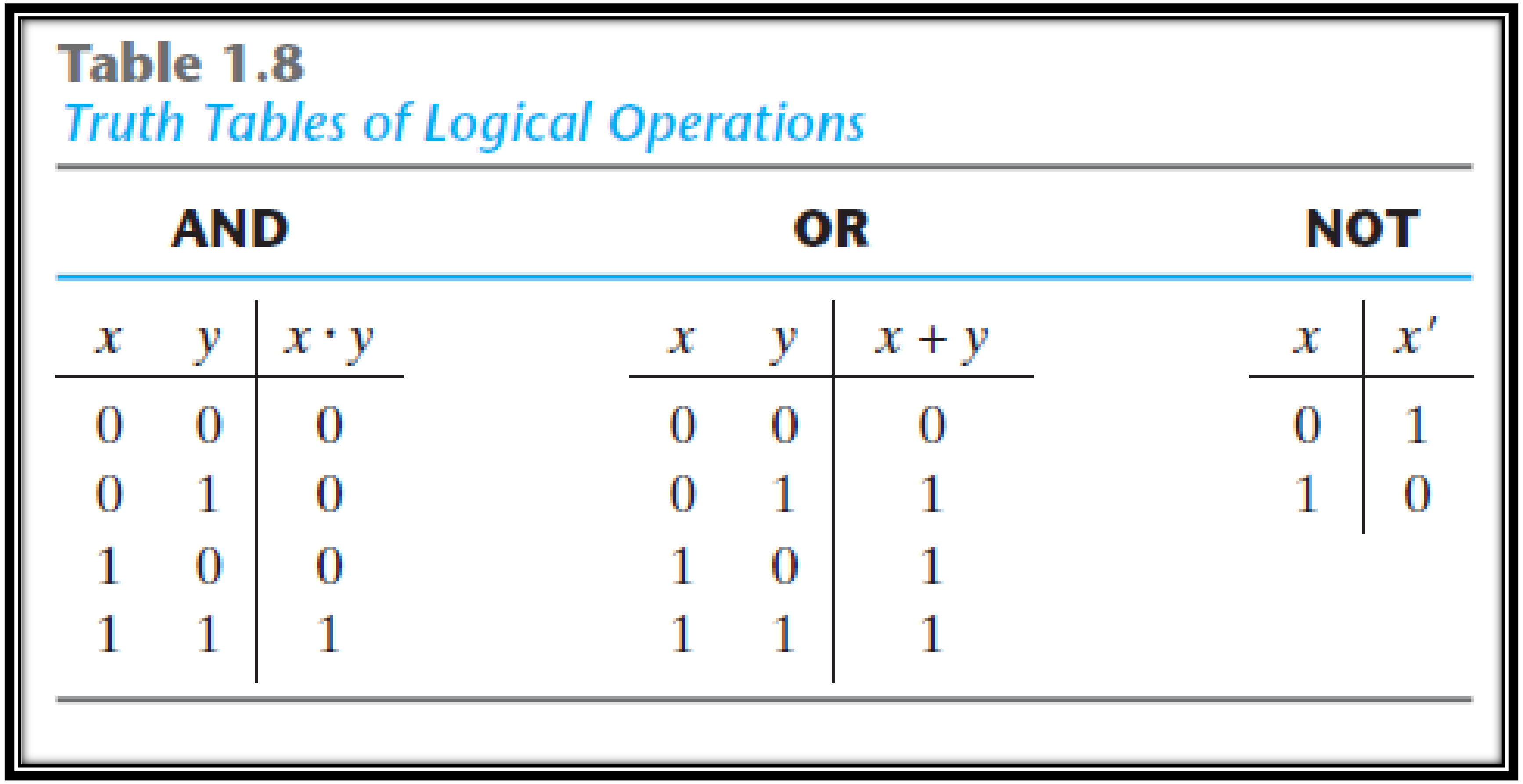


Table 1.8
Truth Tables of Logical Operations

AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

x	x'
0	1
1	0

Logic Gates

- Logic gates are electronic circuits that operate on one or more input signals to produce an output signal.
- Electrical signals such as voltages or currents exist as analog signals having values over a given continuous range, say, 0 to 3 V, but in a digital system these voltages are interpreted to be either of two recognizable values, 0 or 1.
- Voltage-operated logic circuits respond to two separate voltage levels that represent a binary variable equal to logic 1 or logic 0.
- For example, a particular digital system may define logic 0 as a signal equal to 0 V and logic 1 as a signal equal to 3 V.
- In practice, each voltage level has an acceptable range, as shown in Fig. 1.3 .
- The input terminals of digital circuits accept binary signals within the allowable range and respond at the output terminals with binary signals that fall within the specified range.

Logic Gates

- The intermediate region between the allowed regions is crossed only during a state transition.
- Any desired information for computing or control can be operated on by passing binary signals through various combinations of logic gates, with each signal representing a particular binary variable.
- When the physical signal is in a particular range it is interpreted to be either a 0 or a 1.
- The graphic symbols used to designate the three types of gates are shown in Fig. 1.4 .
- The gates are blocks of hardware that produce the equivalent of logic-1 or logic-0 output signals if input logic requirements are satisfied.
- The input signals *x* and *y* in the AND and OR gates may exist in one of four possible states: 00, 10, 11, or 01

Logic Gates

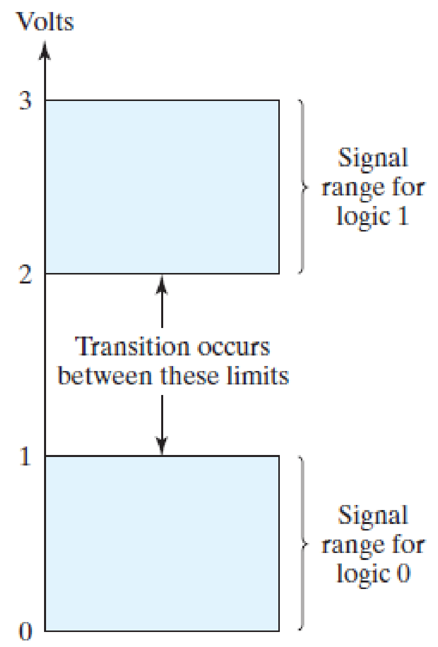


FIGURE 1.3
Signal levels for binary logic values

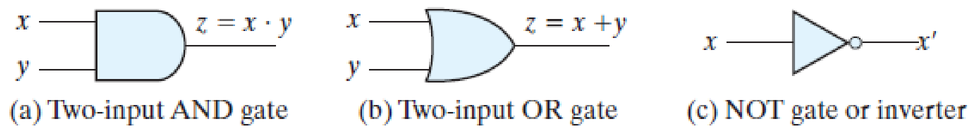


FIGURE 1.4
Symbols for digital logic circuits

Logic Gates

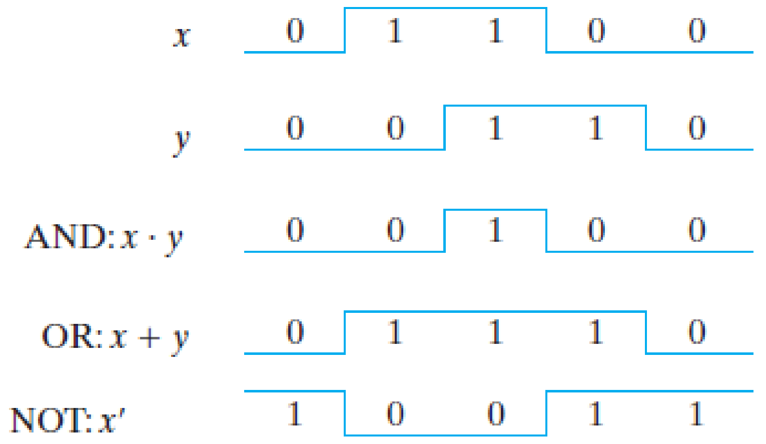
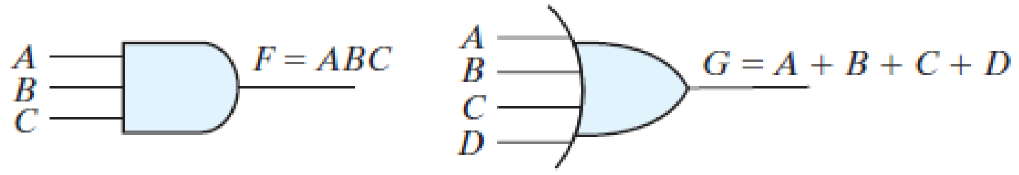


FIGURE 1.5
Input–output signals for gates



(a) Three-input AND gate (b) Four-input OR gate

FIGURE 1.6
Gates with multiple inputs