Chapter 1

Binary Systems

Introduction

We are in “Information age” since digital systems have such a prominent and growing role in modern society. They are involved in our business transactions, communications, transportation, medical treatment and entertainment. In industrial world they are heavily employed in design, manufacturing, distribution and sales.

Digital system

A digital signal is discrete-time as well as discrete-valued signal

A system that works with digital signal is digital system

Digital doesn’t always mean binary

Why digital?

Perfect Reconstruction/Regeneration

Error control

Encryption/Decryption

Advantages of Digital System

- Easy to store and retrieve digital information
- Easy to process digital data
- Digital is less error prone
- Error can be controlled
- Noise does not accumulate from one logic stage to next as it does in analog system (Regeneration)
- The ease of large scale fabrication
- Can be processed by a general purpose processor

Disadvantages of Digital System

- Use more energy than analog circuits to accomplish the same tasks, thus producing more heat as well.
- Digital circuits are often fragile, in that if a single piece of digital data is lost or misinterpreted, the meaning of large blocks of related data can completely change.
- Quantization error during analog signal sampling.

Analog system

It is system that manipulate physical quantities that represent in analog form.

Analogue systems process analogue signals which can take any value within a range, for example the output from an LDR (light sensor) or a microphone.

An audio amplifier is an example of an analogue system. The amplifier produces an output voltage which can be any value within the range of its power supply.

Logic signals

Most digital systems use the simplest possible type of signal which has just two values. This type of signal is called a logic signal because the two values (or states) can be called true and false. Normally the positive supply voltage +Vs represents true and 0V represents false. Other labels for the true and false states are shown in the table.

Logic states

True	False
1	0
High	Low
+Vs	0V
On	Off

Analog vs. Digital design

Digital design need to handle with only few specified (mostly two) logic levels

– It is less important to precisely simulate resistive, capacitive and inductive parameters

– Design simulation can be based on the logic only, tools can therefore follow the truth tables

Analog design requires precise voltage and current characteristics of the devices

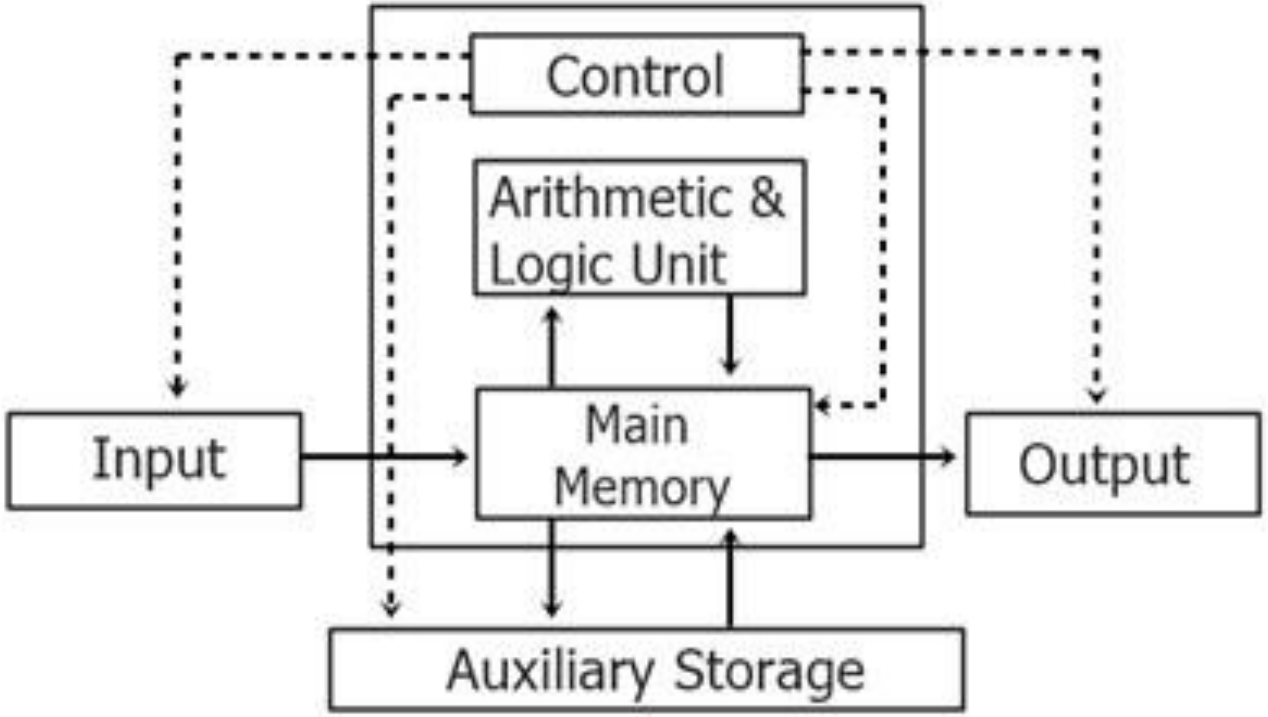
– Parasitic capacitance, resistance, inductance are very important

– Circuit elements are non-linear

– Design verifications require complex simulations

involving solutions of differential equations

Block diagram of digital computer



Block Diagram of Computer

Working principle

Memory stores programs as well as input, output and intermediate data. The Datapath performs arithmetic and other data-processing operations as specified by the program. The control unit supervises the flow of information between the various units. A Datapath, when combined with the control unit, forms a component referred to as a central processing unit, or CPU. The program and data prepared by the user are transferred into memory by means of an input device such as a keyboard. An output device, such as a CRT (cathode-ray tube) monitor, displays the results of the computations and presents them to the user.

BINARY DIGITS, LOGIC LEVELS, AND DIGITAL WAVEFORMS

I. **BINARY DIGITS**

Digital electronics involves circuits and systems in which there are only two possible States.

These states are represented by two different voltage levels: A HIGH and a LOW. The two states can also be represented by current levels, bits and bumps on a CD or DVD. Etc.

Each of the two digits in the binary system, 1 and 0, is called a bit, which is a contraction of the words binary digit. In digital circuits. Two different voltage levels are used to represent the two bits. Generally, (1) is represented by the higher voltage, which we will refer to as a HIGH, and a (0) is represented by the lower voltage level, which we will refer to as a LOW. This is called positive logic and will be used.

HIGH = 1 and LOW = 0

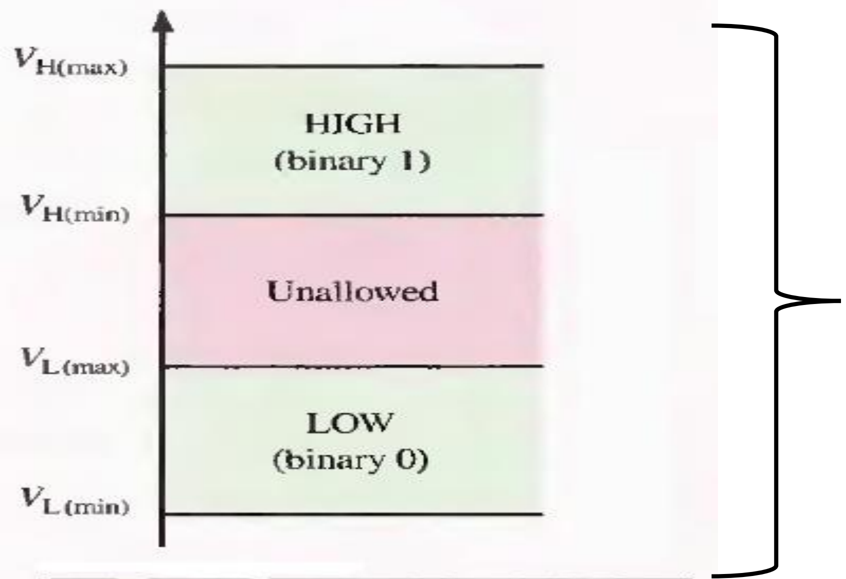
Another system in which a (1) is represented by a LOW and a (0) is represented by a HIGH is called negative logic.

HIGH = 0 and LOW = 1

The combinations of the two states, called codes, are used to represent numbers, symbols, alphabetic characters, and other types of information. The two-state number system is called binary, and its two digits are (0) and (1).

LOGIC LEVELS

Ideally, one voltage level represents a HIGH and another voltage level represents a LOW. In a practical digital circuit, however, a HIGH can be any voltage between a specified minimum value and a specified maximum value. Likewise, a LOW can be any voltage between a specified minimum and a specified maximum. There can be no overlap between the accepted range of HIGH levels and the accepted range of LOW levels. Figure, illustrates the general range of LOWs and HIGHs for a digital circuit.

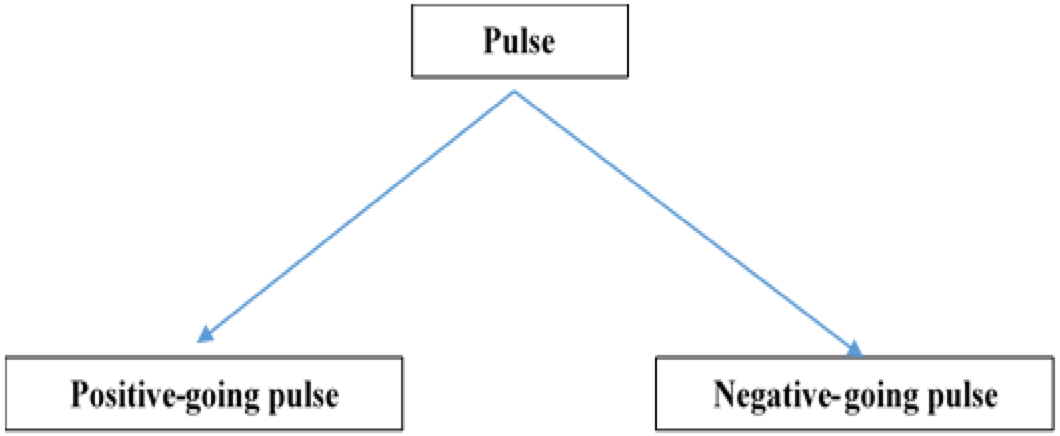


Logic level ranges of voltage for a digital circuit.

Positive Logic

DIGITAL WAVEFORMS

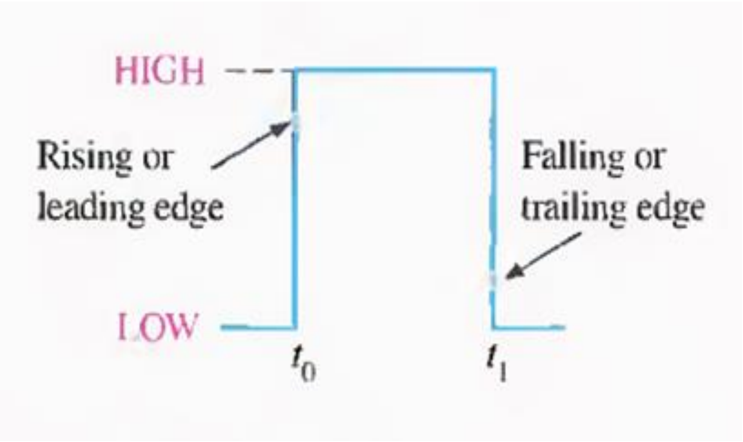
Digital waveforms consist of voltage levels that are changing back and forth between the HIGH and LOW levels. A digital waveform is made up of a series of pulses.

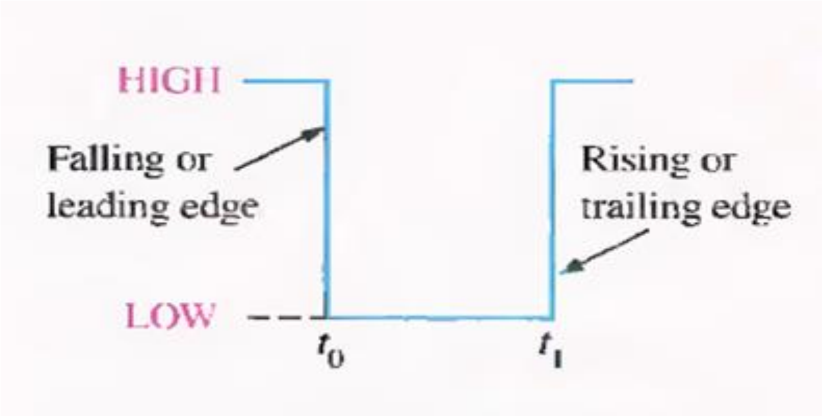


Pulse

Positive-going pulse

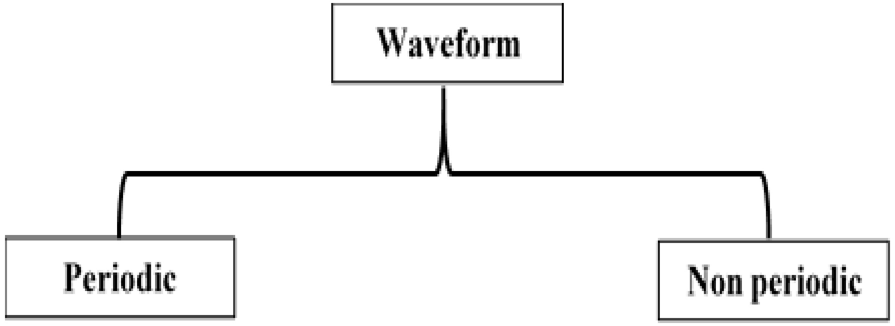
Negative-going pulse





Waveform Characteristics

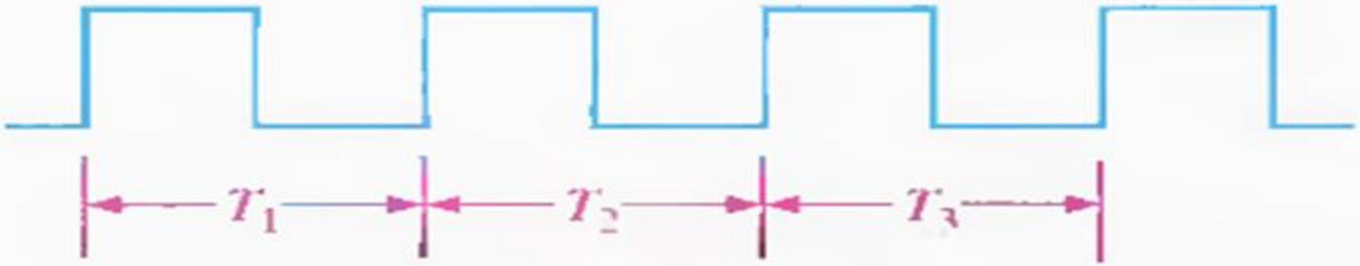
Most Waveform encountered in digital systems are composed of series of pulses, sometimes called pulse trains.



A periodic pulse waveform is one that repeats itself at a fixed interval, called a period (T). The frequency (f) is the rate at which it repeats itself and is measured in hertz (Hz).

A non-periodic pulse waveform, of course, does not repeat itself at fixed intervals and may be composed of pulses of randomly differing pulse widths and/or randomly differing time intervals between the pulses. An example of each type is shown in Figure.

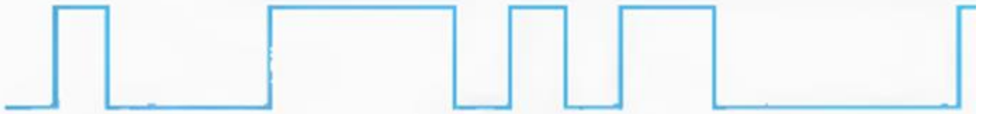
Waveform Types



$$\text{Period} = T_1 = T_2 = T_3 = \ldots = T_n$$

$$\text{Frequency} = \frac{1}{T}$$

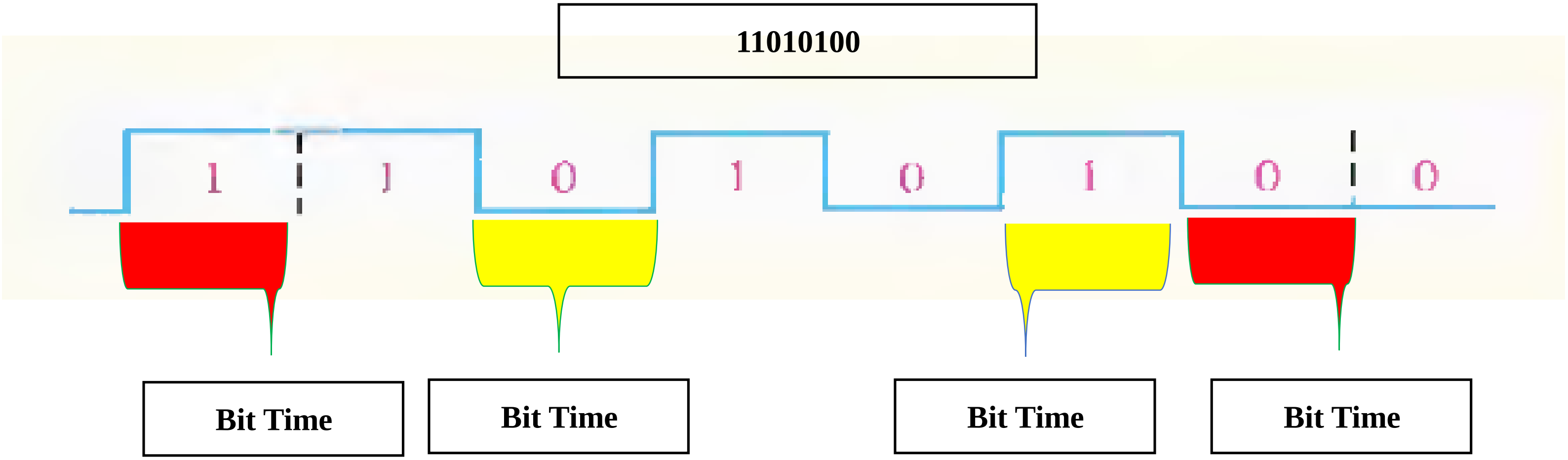
(a) Periodic (square wave)



(b) Nonperiodic

Digital Waveform Carries Binary Information

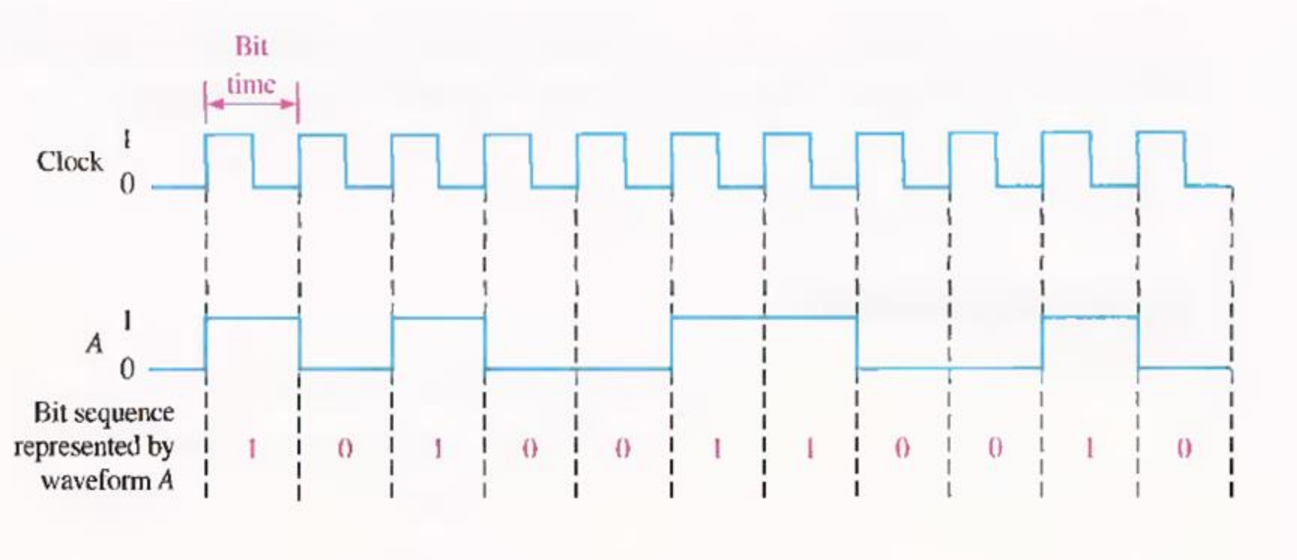
Binary information that is handled by digital systems appears as waveforms that represent sequences of bits. When the waveform is HIGH, a binary 1 is present; when the waveform is LOW, a binary 0 is present. Each bit in a sequence occupies a defined time interval called a bit time.



The Clock

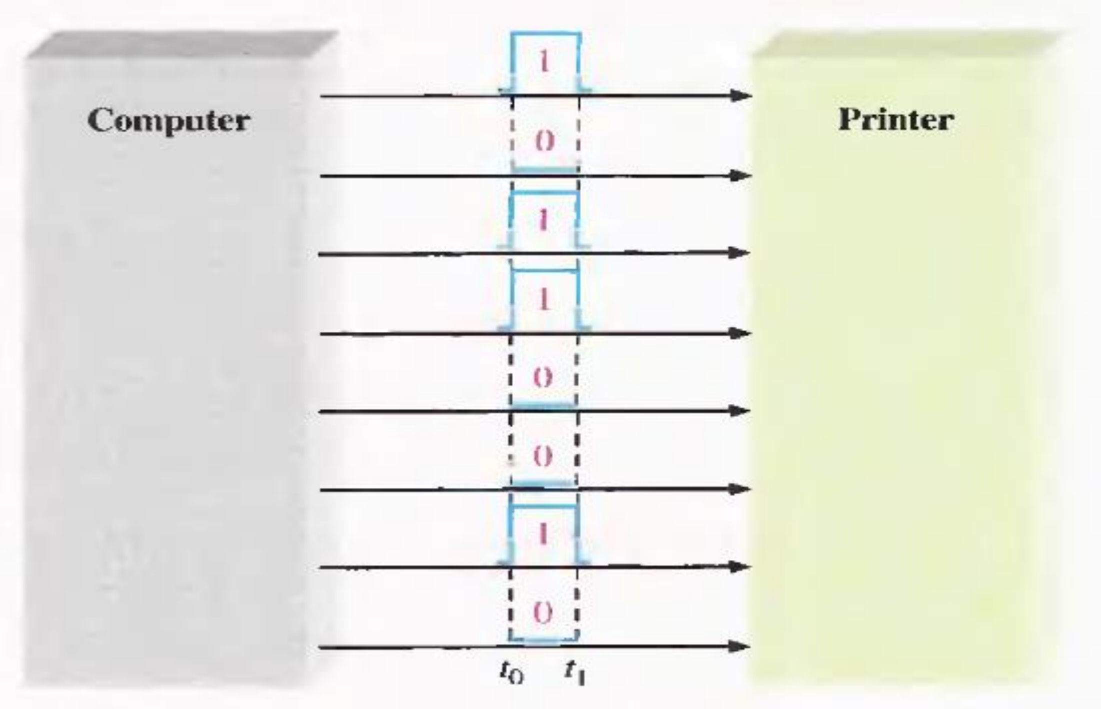
The clock is a periodic waveform in which each interval between pulses (the period) equals the time for one bit. In digital systems, all waveforms are synchronized with a clock.

An example of a clock waveform is shown in Figure. Notice that, in this case, each change in level of waveform A occurs at the leading edge of the clock waveform. In other cases, level changes occur at the trailing edge of the clock. During each bit time of the clock, waveform A is either HIGH or LOW. These HIGHs and LOWs represent a sequence of bits, as indicated. A group of several bits can be used as a piece of binary information, such as a number or a letter. The clock waveform itself does not carry information

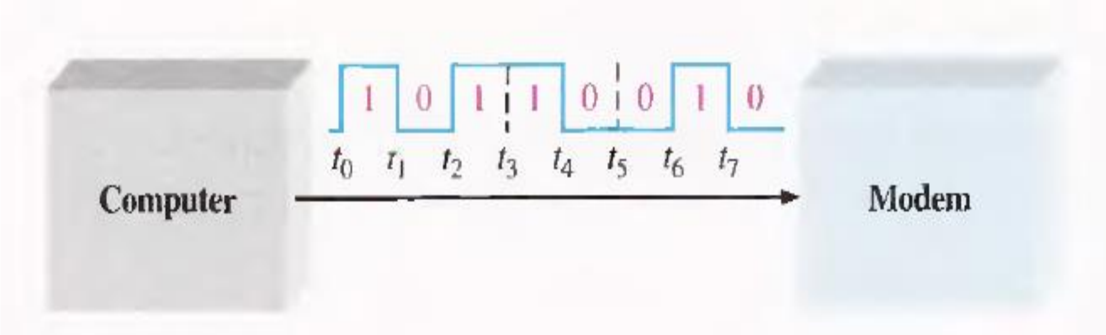


Data Transfer

For example, numbers stored in binary form in the memory of a computer must be transferred to the computer's central processing unit in order to be added. The sum of the addition must then be transferred to a monitor for display and/or transferred back to the memory. In computer systems, as illustrated in Figure, binary data are transferred in two ways: serial and parallel.



(b) Parallel transfer of 8 bits of binary data from computer to printer. The beginning time is t_0.



(a) Serial transfer of 8 bits of binary data from computer to modem. Interval t_0 to t_1 is first.

Data Transfer

When bits are transferred in serial from one point to another. They are sent one bit at a time along a single line. As illustrated in Figure above.

For the case of a computer-to- modem transfer. During the time interval from t0 to t1" the first bit is transferred. During the time interval from t1 to t2, the second bit is transferred, and so on. To transfer eight bits in series, it takes eight time intervals.

When bits are transferred in parallel form, all the bits in a group are sent out on separate lines at the same time. There is one line for each bit, as shown in Figure above for the example of eight bits being transferred from a computer to a printer. To transfer eight bits in parallel, it takes one time interval compared to eight time intervals for the serial transfer.

To summarize, an advantage of serial transfer of binary data is that a minimum of only one line is required. In parallel transfer, a number of lines equal to the number of bits to be transferred at one time is required.

Number system

Number: Arithmetical value representing a particular quantity. The various types of numbers are Natural Numbers, Whole Numbers, Integers, Rational Numbers, Irrational Numbers, Real Numbers etc.

Integers

Integers are the numbers that includes whole numbers along with the negative numbers.

Rational ,irrational, whole number ?

Number system

The technique to represent and work with numbers is called number system. Decimal number system is the most common number system. Other popular number systems include binary number system, octal number system, hexadecimal number system, etc.

Decimal Number System

Decimal number system is a base 10 number system having 10 digits from 0 to 9. This means that any numerical quantity can be represented using these 10 digits. Decimal number system is also a positional value system. This means that the value of digits will depend on its position. Let us take an example to understand this.

Decimal number system

Example – 734, 971 and 207. The value of 7 in all three numbers is different−

In 734, value of 7 is 7 hundreds or 700 or 7 × 100 or 7 × 10^2

In 971, value of 7 is 7 tens or 70 or 7 × 10 or 7 × 10^1

In 207, value 0f 7 is 7 units or 7 or 7 × 1 or 7 × 10^0

In digital systems, instructions are given through electric signals; variation is done by varying the voltage of the signal. Having 10 different voltages to implement decimal number system in digital equipment is difficult. So, many number systems that are easier to implement digitally have been developed.

Binary Number System

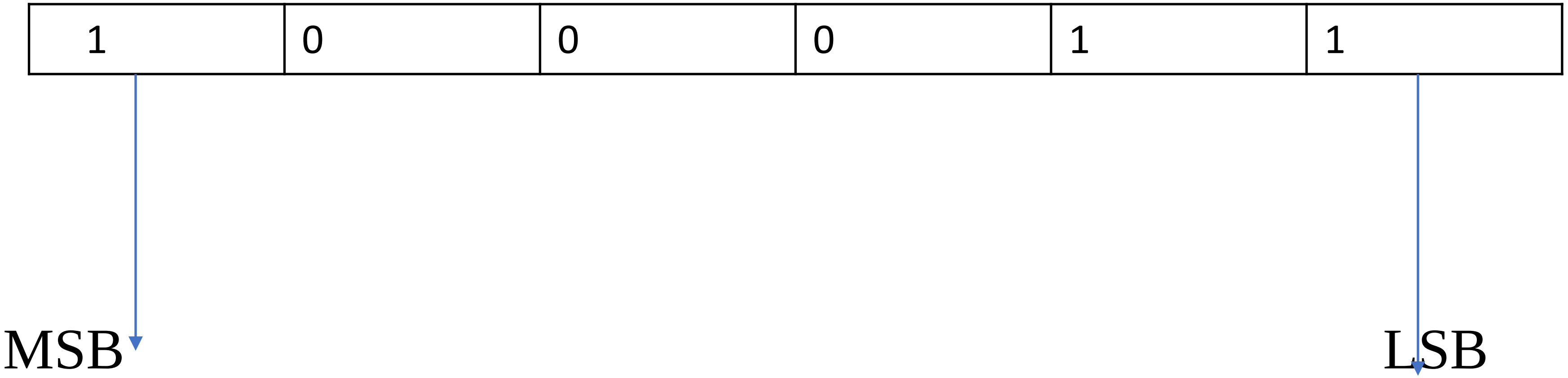
The easiest way to vary instructions through electric signals is two-state system – on and off. On is represented as 1 and off as 0, though 0 is not actually no signal but signal at a lower voltage. The number system having just these two digits – 0 and 1 – is called binary number system.

Each binary digit is also called a bit. Binary number system is also positional value system, where each digit has a value expressed in powers of 2, as displayed here.

2^5	2^4	2^3	2^2	2^1	2^0

In any binary number, the rightmost digit is called least significant bit (LSB) and leftmost digit is called most significant bit (MSB).

Binary Number System



1	0	0	0	1	1

MSB LSB

Computer memory is measured in terms of how many bits it can store.
Here is a chart for memory capacity conversion.

Binary Arithmetic

1. Addition
2. Subtraction
3. Division
4. Multiplication

Perform the following binary operation:

10101*101

101011/111

100001-111

10111+11101

Unit of data storage

Bit=1 0r 0 (on or off)

1 byte (B) = 8 bits

1 Kilobytes (KB) = 1024 bytes

1 Megabyte (MB) = 1024 KB

1 Gigabyte (GB) = 1024 MB

1 Terabyte (TB) = 1024 GB

1 Exabyte (EB) = 1024 PB

1 Zettabyte = 1024 EB

1 Yottabyte (YB) = 1024 ZB

Octal Number System

Octal number system has eight digits – 0, 1, 2, 3, 4, 5, 6 and 7. Octal number system is also a positional value system with where each digit has its value expressed in powers of 8.

Decimal equivalent of any octal number is sum of product of each digit with its positional value.

$726 = 7\times8^2 + 2\times8^1 + 6\times8^0$

$= 448 + 16 + 6$

$= 470$

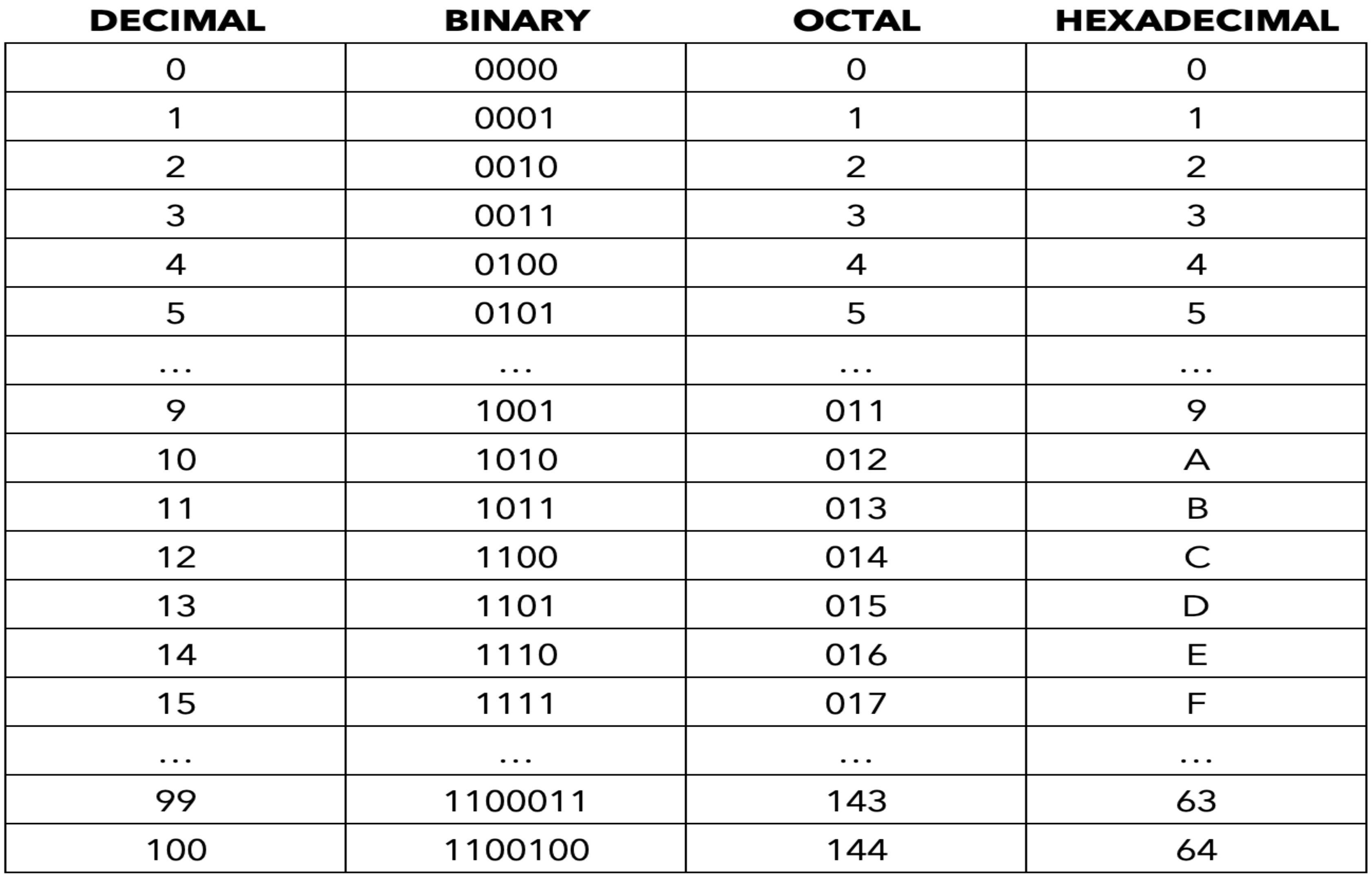
Hexadecimal Number System

Hexadecimal number system has 16 symbols – 0 to 9 and A to F where A is equal to 10, B is equal to 11 and so on till F. Hexadecimal number system is also a positional value system with where each digit has its value expressed In power of 16.

Decimal equivalent of any hexadecimal number is sum of product of each digit with its positional value.

What is the decimal equivalent of hex number CAFÉ ?

Number System Relationship



DECIMAL	BINARY	OCTAL	HEXADECIMAL
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
...	...	...	...
9	1001	011	9
10	1010	012	A
11	1011	013	B
12	1100	014	C
13	1101	015	D
14	1110	016	E
15	1111	017	F
...	...	...	...
99	1100011	143	63
100	1100100	144	64

Number Base Conversions

Case I: **Base-r system to Decimal**: Base-r system can be binary (r=2), octal (r=8), hexadecimal (r=16), base-60 system or any other. For decimal system as destination of conversion, we just use power series explained above with varying r and sum the result according to the arithmetic rules of base-10 system.

Case II: **Decimal to Base-r system:** Conversion follows following algorithm.

1. Separate the number into integer and fraction parts if radix point is given.

2. Divide "Decimal Integer part" by base r repeatedly until quotient becomes zero and storing remainders at each step.

3. Multiply "Decimal Fraction part" successively by r and accumulate the integer digits so obtained.

4. Combine both accumulated results and parenthesize the whole result with subscript r.

Number Base Conversions

Q. $(41.6875)_{10}=(?)_2$

Q. $(41.675)_8=(?)_2$

Q. $(A5.6875)_{16}=(?)_2$

Case III: **Binary to octal & hexadecimal and vice-versa**: Conversion from and to binary, octal and hexadecimal representation plays an important part in digital computers. Since, $2^3 = 8$, octal digit can be represented by at least 3 binary digits. So to convert given binary number into its equivalent octal, we divide

it into groups of 3 bits, give each group an octal symbol and combine the result.

Integer part: Group bits from right to left of an octal point. 0's can be added to make

it multiple of 3.

Fractional part: Group bits from left to right of an octal point. 0's must be added to

if bits are not multiple of 3 (Note it).

$2^4 = 16$, each hex digit corresponds to 4 bits. So to convert given binary number into its equivalent hex, we divide it into groups of 4 bits, give each group a hex digit and combine the result. If hex point is given, then process is similar as of octal.

Number Base Conversions

Q. $(41.6875)_{10}{}^{=}(?)_{8}$

Q. $(41.675)_{8}{}^{=}(?)_{16}$

Q. $(A5.6875)_{16}{}^{=}(?)_{8}$

Q. $(1010101.1101)_{2}{}^{=}(?)_{8}$

Q. $(11101.1010)_{2}{}^{=}(?)_{16}$

Complements

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation. There are two types of complements for each base-r system: r's complement and the second as the (r - 1)'s complement. When the value of the base r is substituted, the two types are referred to as the 2's complement and 1's complement for binary numbers, the 10's complement and 9's complement for decimal numbers etc.

(r-1)'s Complement (diminished radix compl.)

(r-1)'s complement of a number N is defined as

(r^n-1) –N

Where N is the given number

r is the base of number system

n is the number of digits in the given number

To get the (r-1)'s complement fast, subtract each

digit of a number from (r-1).

Example:

- 9's complement of 835 is 164 (Rule: (10^n-1) –N)
- 1's complement of 1010 is 0101 (bit by bit complement operation)

r's Complement (radix complement)

r's Complement (radix complement)

r's complement of a number N is defined as r^n–N

Where N is the given number

r is the base of number system

n is the number of digits in the given number

To get the r's complement fast, add 1 to the low-order digit of its (r-1)'s complement.

Example:

- 10's complement of 835 is 164 + 1 =

165

- 2's complement of 1010 is 0101 + 1 = 0110

Subtraction with complements

The direct method of subtraction taught in elementary schools uses the borrow concept. When subtraction is implemented with digital hardware, this method is found to be less efficient than the method that uses complements.

1. Subtraction using 1’s complement
2. Subtraction using 2’s complement
3. Subtraction using 10’s complement
4. Subtraction using 9’s complement

Perform the following:

1. Subtract 101011 from 11101 using 1’s complement
2. Subtract 10111 from 111011 using 1’s complement
3. Subtract 101011 from 11101 using 2’s complement
4. Subtract 10111 from 111011 using 2’s complement
5. Subtract 5657 from 6745 using 9’s complement
6. Subtract 456434 from 124567 using 10’s complement

Binary Codes

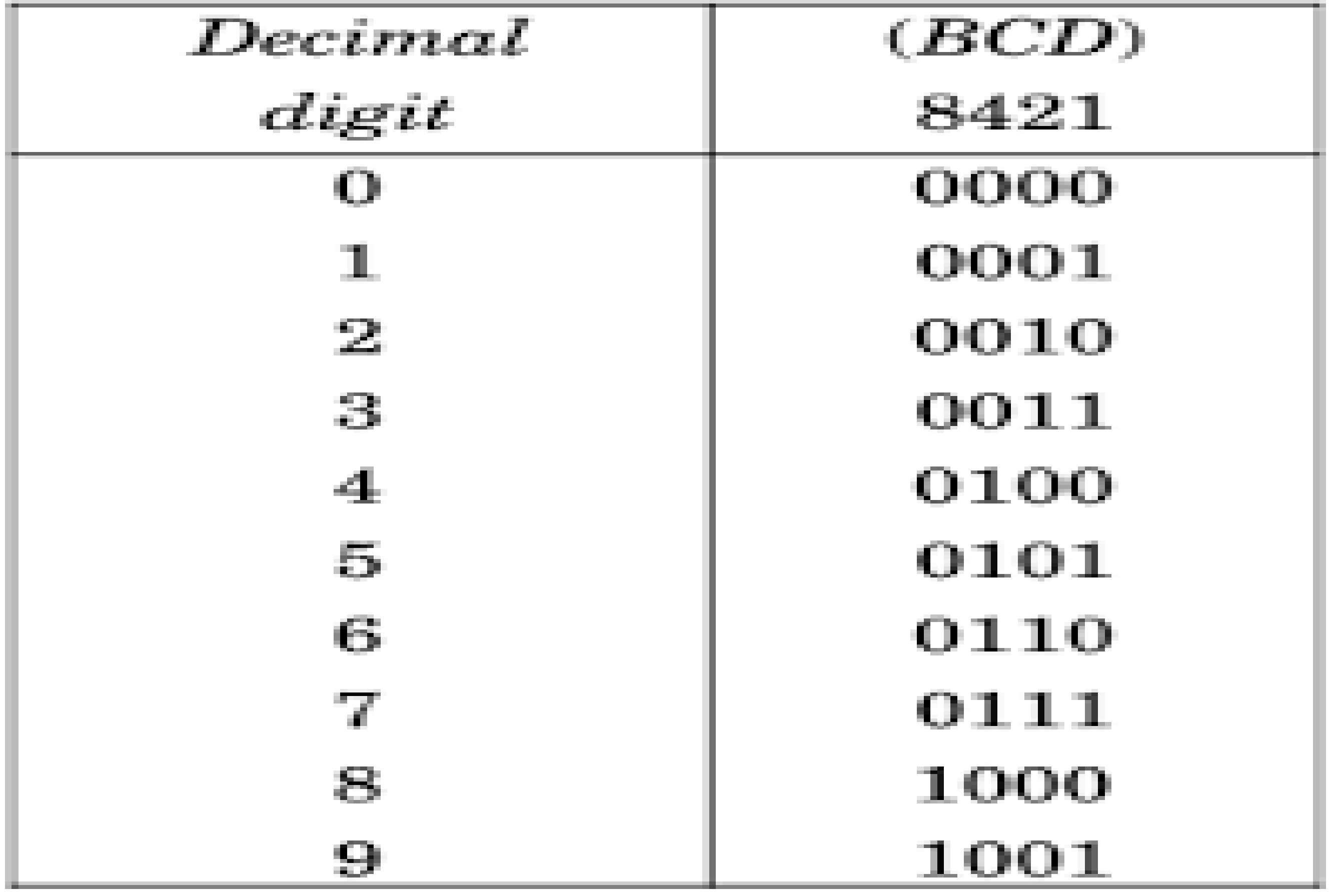
Electronic digital systems use signals that have two distinct values and circuit elements that have two stable states. There is a direct analogy among binary signals, binary circuit elements, and binary digits. A binary number of n digits, for example, may be represented by n binary circuit elements, each having an output signal equivalent to a 0 or a 1.

Digital systems represent and manipulate not only binary numbers, but also many other discrete elements of information. Any discrete element of information distinct among a group of quantities can be represented by a binary code. Binary codes play an important role in digital computers. The codes must be in binary because computers can only hold 1's and 0's.

Binary Coded Decimal (BCD)

The binary number system is the most natural system for a computer, but people are accustomed to the decimal system. So, to resolve this difference, computer uses decimals in coded form which the hardware understands. A binary code that distinguishes among 10 elements of decimal digits must contain at least four bits. Numerous different binary codes can be obtained by arranging four bits into 10 distinct combinations. The code most commonly used for the decimal digits is the straightforward binary assignment listed in the table below. This is called binary-coded decimal and is commonly referred to as BCD

Binary Coded Decimal (BCD)



Decimal	(BCD)
digit	8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Binary Coded Decimal (BCD)

A number with n decimal digits will require 4n bits in BCD. E.g. decimal 396 is represented in BCD with 12 bits as 0011 1001 0110.

Numbers greater than 9 has a representation different from its equivalent binary number, even though both contain 1's and 0's.

Binary combinations 1010 through 1111 are not used and have no meaning in the BCD code.

Example:

$(185)_{10}$ = (0001 1000 0101)BCD = $(10111001)_2$

Error-Detection codes

Binary information can be transmitted from one location to another by electric wires or other communication medium. Any external noise introduced into the physical communication medium may change some of the bits from 0 to 1 or vice versa.

The purpose of an error-detection code is to detect such bit-reversal errors. One of the most common ways to achieve error detection is by means of a parity bit. A parity bit is the extra bit included to make the total number of 1's in the resulting code word either even or odd.

Error Checking Mechanism

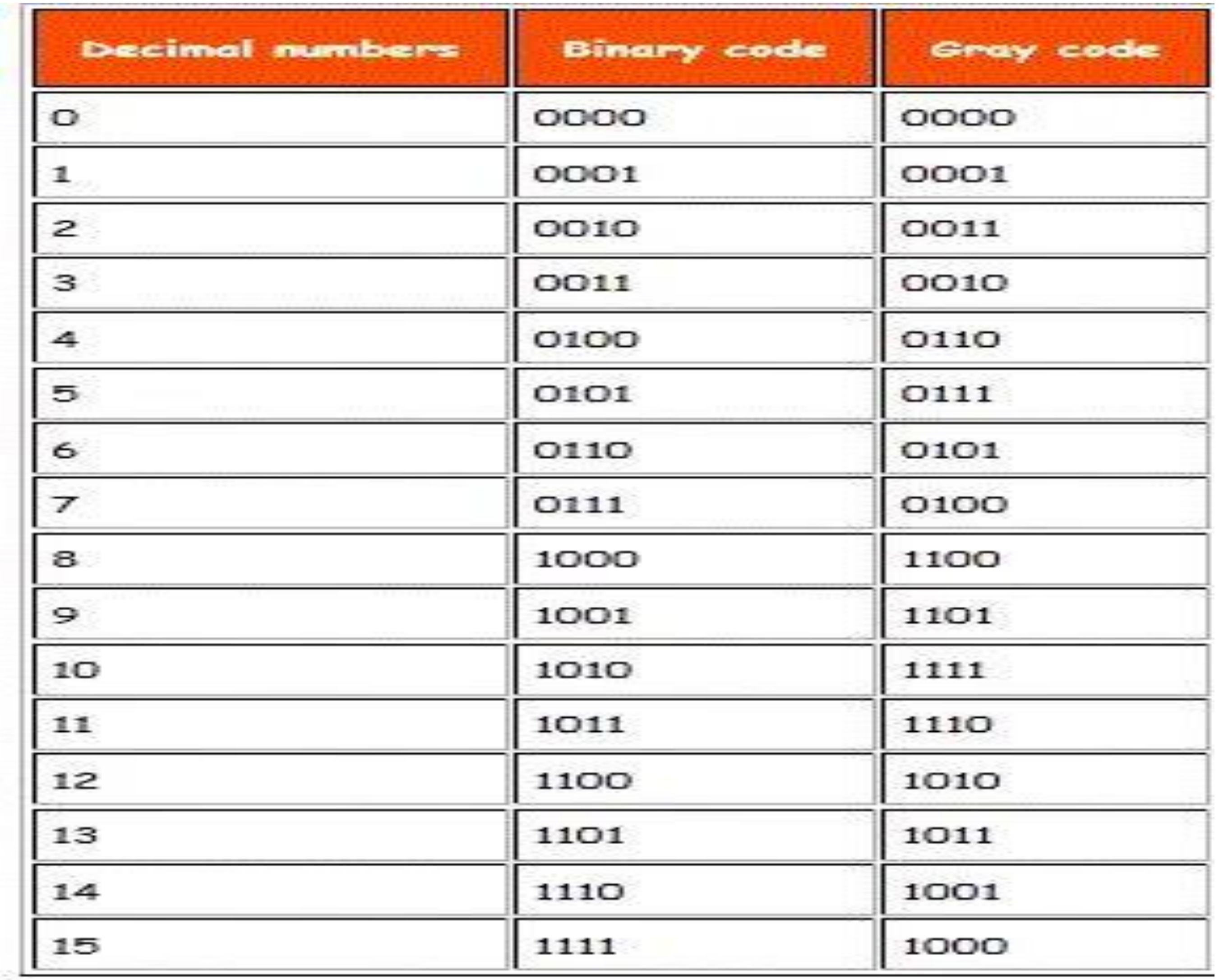
During the transmission of information from one location to another, an even parity bit is generated in the sending end for each message transmission. The message, together with the parity bit, is transmitted to its destination. The parity of the received data is checked in the receiving end. If the parity of the received information is not even, it means that at least one bit has changed value during the transmission.

This method detects one, three, or any odd combination of errors in each message that is transmitted. An even combination of errors is undetected. Additional error detection schemes may be needed to take care of an even combination of errors.

Gray code (Reflected code)

It is a binary coding scheme used to represent digits generated from a mechanical sensor that may be prone to error. Used in telegraphy in the late 1800s, and also known as "reflected binary code". Gray code was patented by Bell Labs researcher Frank Gray in 1947. In Gray code, there is only one bit location different between two successive values, which makes mechanical transitions from one digit to the next less error prone. The following chart shows normal binary representations from 0 to 15 and the corresponding Gray code.

Gray code (Reflected code)



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

Gray code (Reflected code)

The Gray code is used in applications where the normal sequence of binary numbers may produce an error or ambiguity during the transition from one number to the next. If binary numbers are used, a change from 0111 to 1000 may produce an intermediate erroneous number 1001 if the rightmost bit takes more time to change than the other three bits. The Gray code eliminates this problem since only one bit changes in value during any transition between two numbers.

Excess-3 code

The excess-3 code (or XS3) is a non-weighted code used to express decimal numbers. It is a self-complementary binary coded decimal (BCD) code and numerical system which has biased representation. It is particularly significant for arithmetic operations as it overcomes shortcoming encountered while using 8421 BCD code to add two decimal digits whose sum exceeds 9. Excess-3 arithmetic uses different algorithm than normal non-biased BCD or binary positional number system.

Representation of Excess-3 Code

Excess-3 codes are unweighted and can be obtained by adding 3 to each decimal digit then it can be represented by using 4 bit binary number for each digit. An Excess-3 equivalent of a given binary number is obtained using the following steps:

Find the decimal equivalent of the given binary number.

Add +3 to each digit of decimal number. Convert the newly obtained decimal number back to binary number to get required excess-3 equivalent.

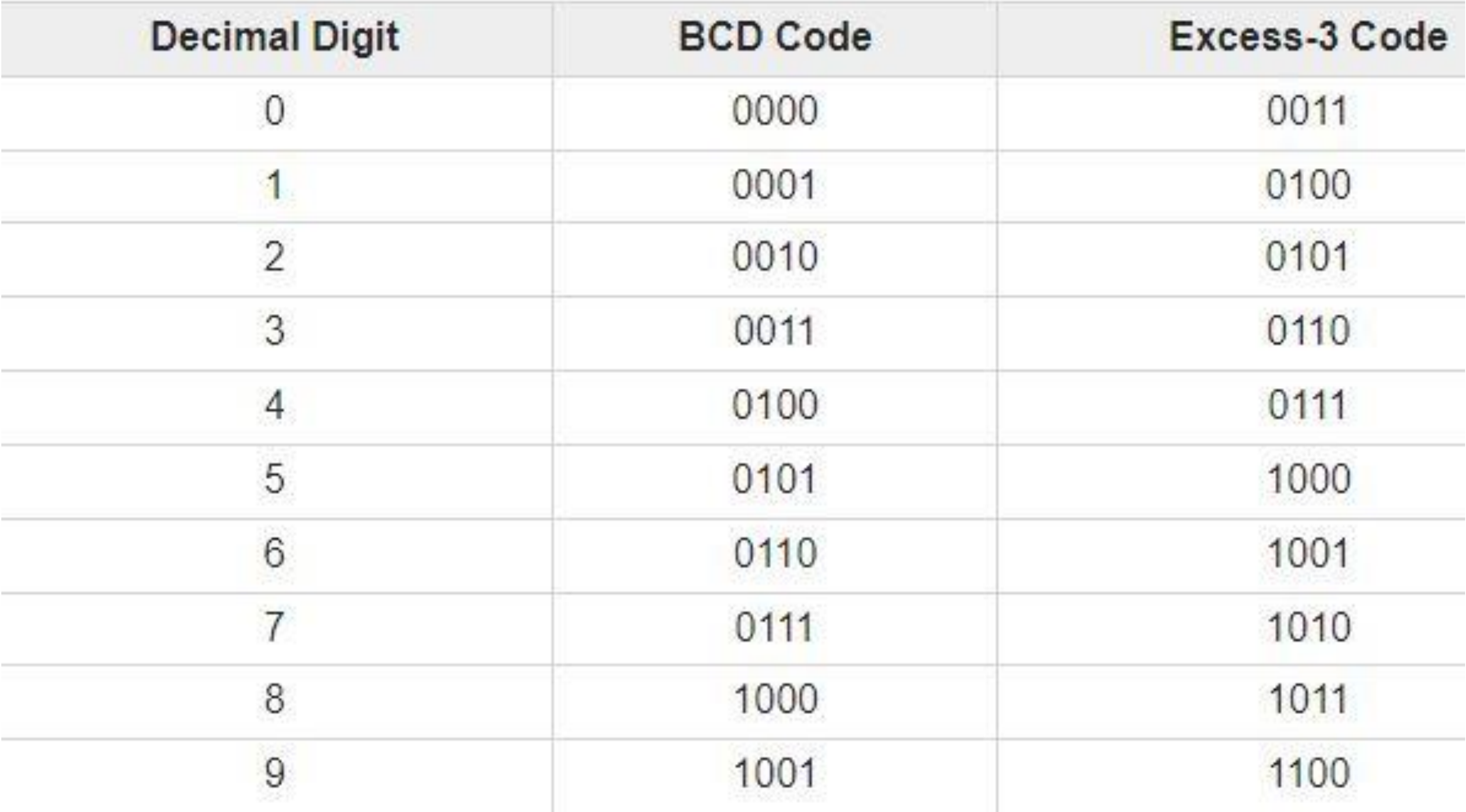
You can add 0011 to each four-bit group in binary coded decimal number (BCD) to get desired excess-3 equivalent.

Example-1 Convert decimal number 23 to Excess-3 code.

So, according to excess-3 code we need to add 3 to both digit in the decimal number then convert into 4-bit binary number for result of each digit. Therefore,

= 23+33=56 =0101 0110 which is required excess-3 code for given decimal number 23.

Representation of Excess-3 Code



Decimal Digit	BCD Code	Excess-3 Code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Alphanumeric codes

Alphanumeric character set is a set of elements that includes the 10 decimal digits, 26 letters of the alphabet and special characters such as $, %, + etc. It is necessary to formulate a binary code for this set to handle different data types. If only capital letters are included, we need a binary code of at least six bits, and if both uppercase letters and lowercase letters are included, we need a binary code of at least seven bits.

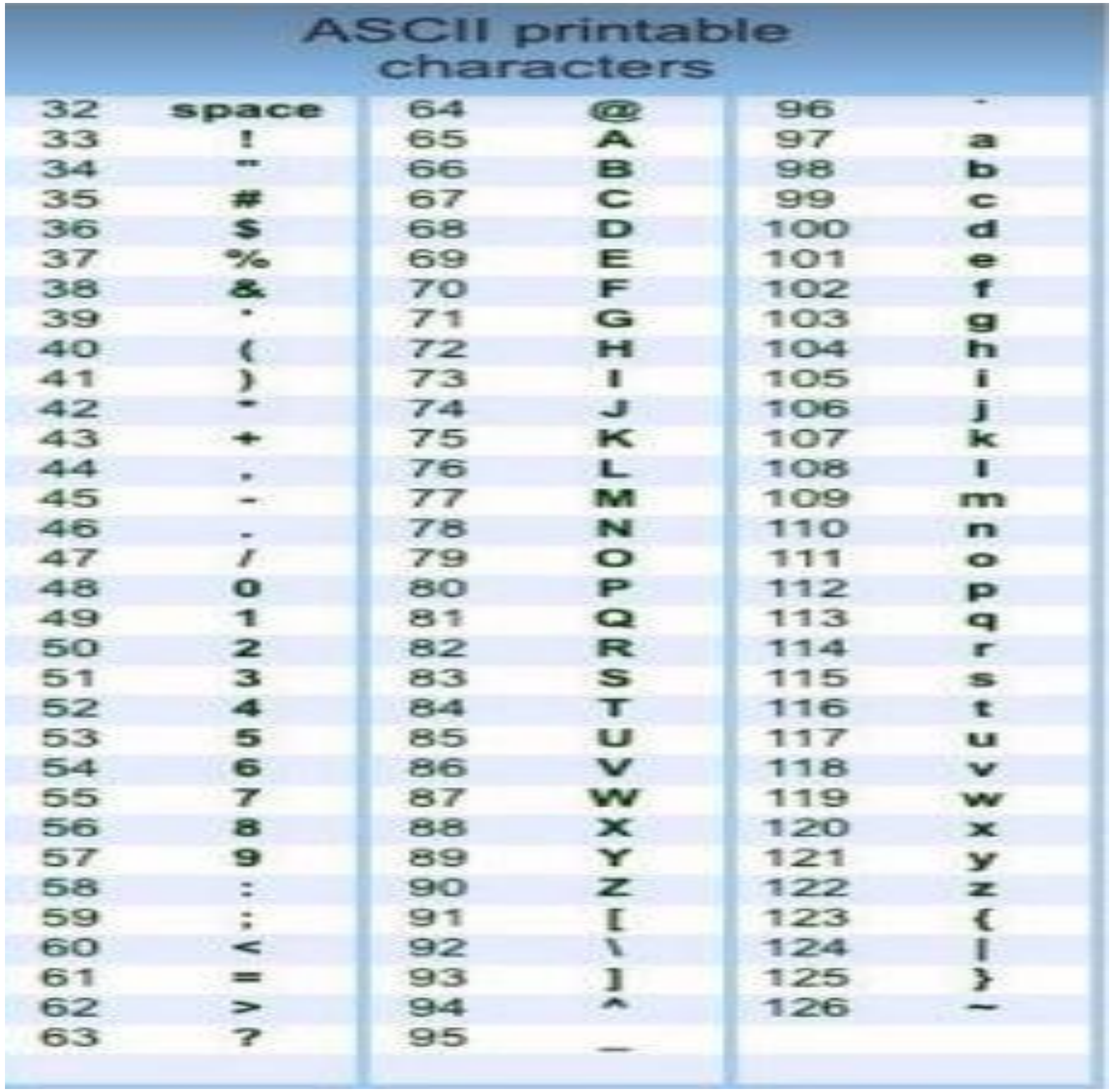
The most common alphanumeric codes is ASCII code

ASCII code

ASCII is the abbreviation for American Standard Code for Information Interchange. ASCII is a universally accepted alphanumeric code used in most computers and other electronic equipment. Most computer keyboards are standardized with the ASCII. When we enter a letter, a number, or control command, the corresponding ASCII code goes into the computer.

- ✓ ASCII has 128 characters and symbols represented by a 7-bit binary code. Actually, ASCII can be considered an 8-bit code with the MSB always 0. This 8-bit code is 00 through 7F in hexadecimal.
- ✓ The first thirty-two ASCII characters are non-graphic commands that are never printed or displayed and are used only for control purposes. Examples of the control characters are "null," "line feed," "start of text," and "escape."
- ✓ The other characters are graphic symbols that can be printed or displayed and include the letters of the alphabet (lowercase and uppercase), the ten decimal digits, punctuation signs and other commonly used symbols.

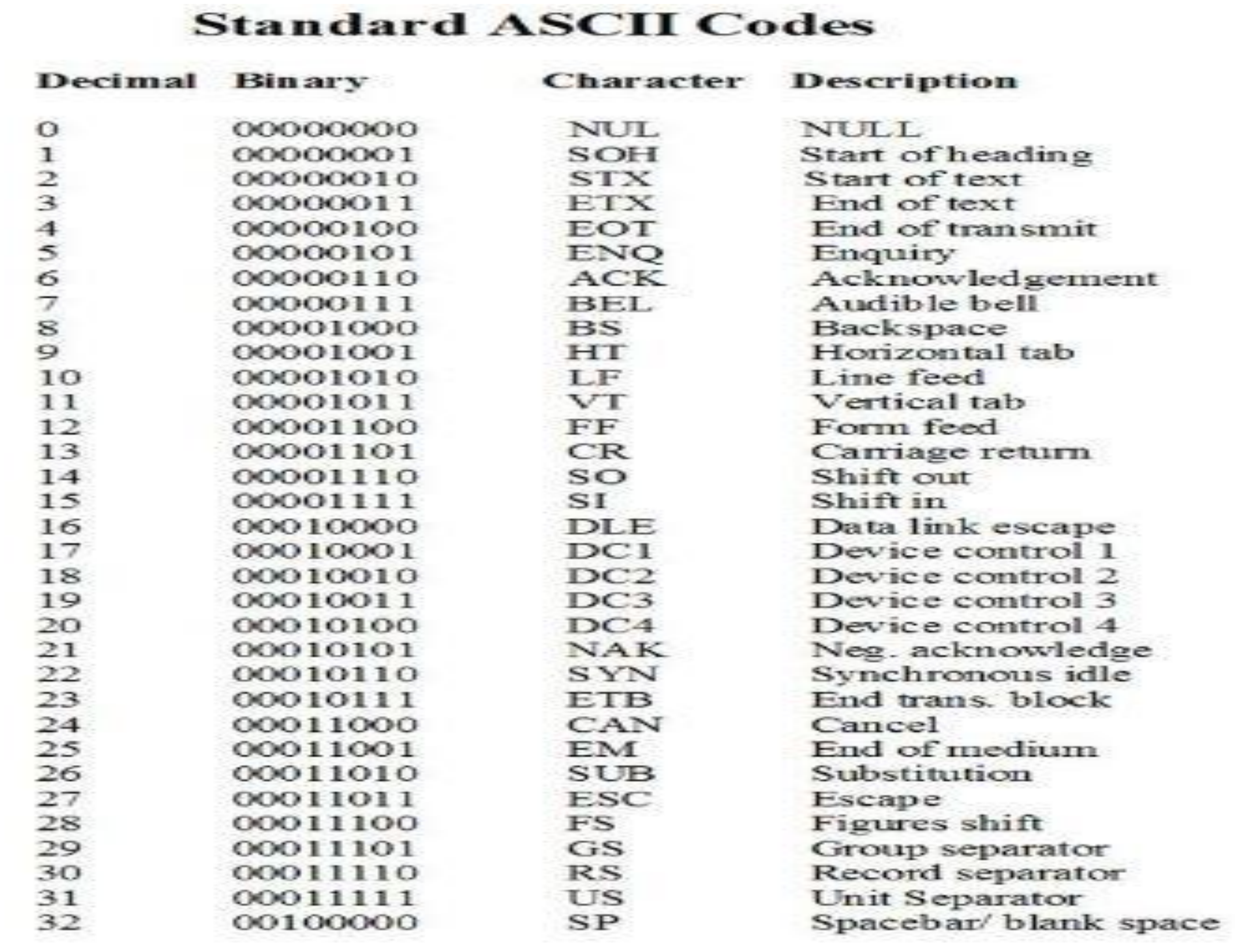
ASCII code



ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

ASCII nonprintable code



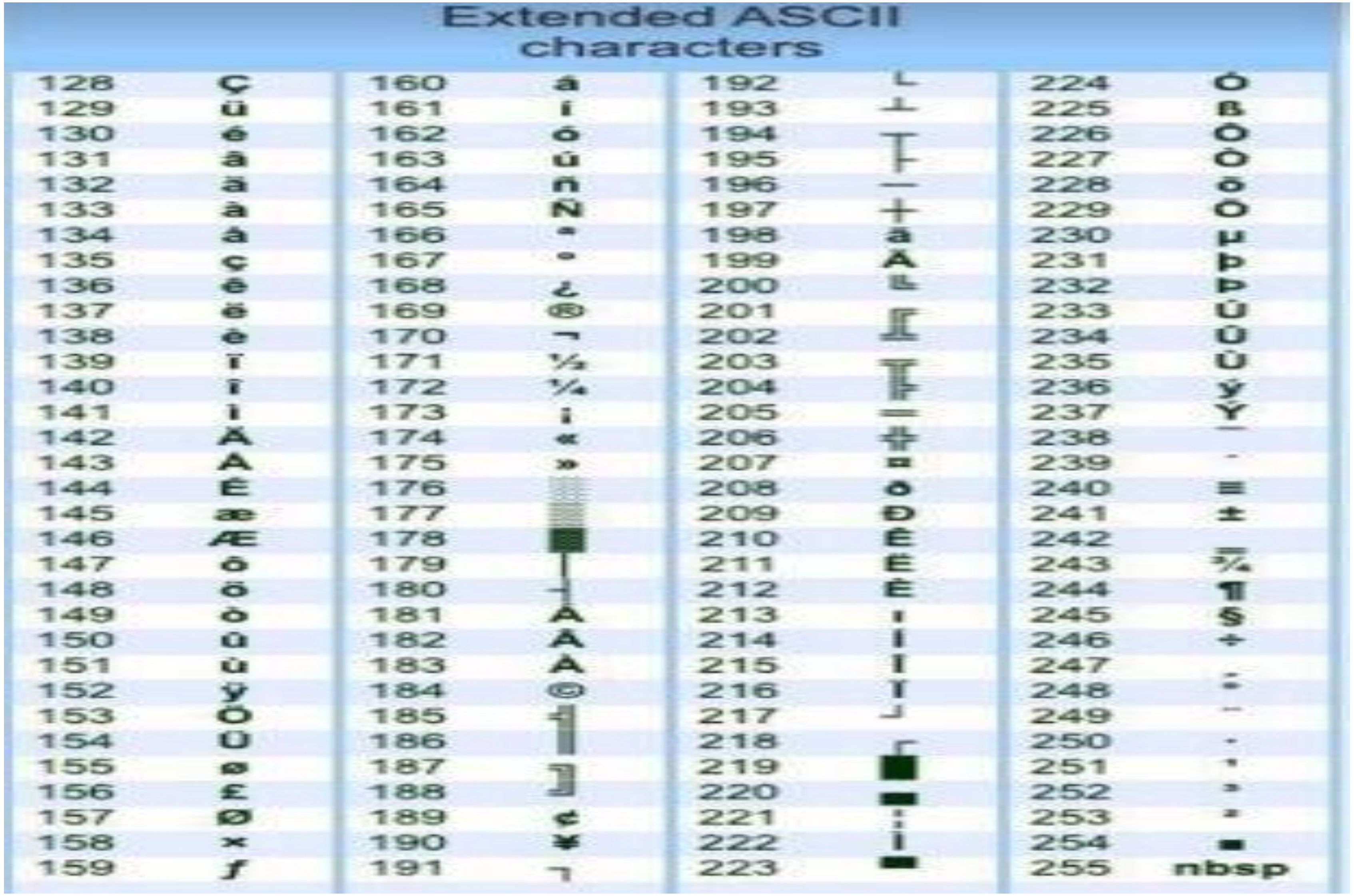
Standard ASCII Codes

Decimal	Binary	Character	Description
0	00000000	NUL	NULL
1	00000001	SOH	Start of heading
2	00000010	STX	Start of text
3	00000011	ETX	End of text
4	00000100	EOT	End of transmit
5	00000101	ENQ	Enquiry
6	00000110	ACK	Acknowledgement
7	00000111	BEL	Audible bell
8	00001000	BS	Backspace
9	00001001	HT	Horizontal tab
10	00001010	LF	Line feed
11	00001011	VT	Vertical tab
12	00001100	FF	Form feed
13	00001101	CR	Carriage return
14	00001110	SO	Shift out
15	00001111	SI	Shift in
16	00010000	DLE	Data link escape
17	00010001	DC1	Device control 1
18	00010010	DC2	Device control 2
19	00010011	DC3	Device control 3
20	00010100	DC4	Device control 4
21	00010101	NAK	Neg. acknowledge
22	00010110	SYN	Synchronous idle
23	00010111	ETB	End trans. block
24	00011000	CAN	Cancel
25	00011001	EM	End of medium
26	00011010	SUB	Substitution
27	00011011	ESC	Escape
28	00011100	FS	Figures shift
29	00011101	GS	Group separator
30	00011110	RS	Record separator
31	00011111	US	Unit Separator
32	00100000	SP	Spacebar/ blank space

Extended ASCII characters

In addition to the 128 standard ASCII characters, there are an additional 128 characters that were adopted by IBM for use in their PCs (personal computers). Because of the popularity of the PC, these particular extended ASCII characters are also used in applications other than PCs and have become essentially an unofficial standard. The extended ASCII characters are represented by an 8-bit code series from hexadecimal 80 to hexadecimal FF.

Extended ASCII characters



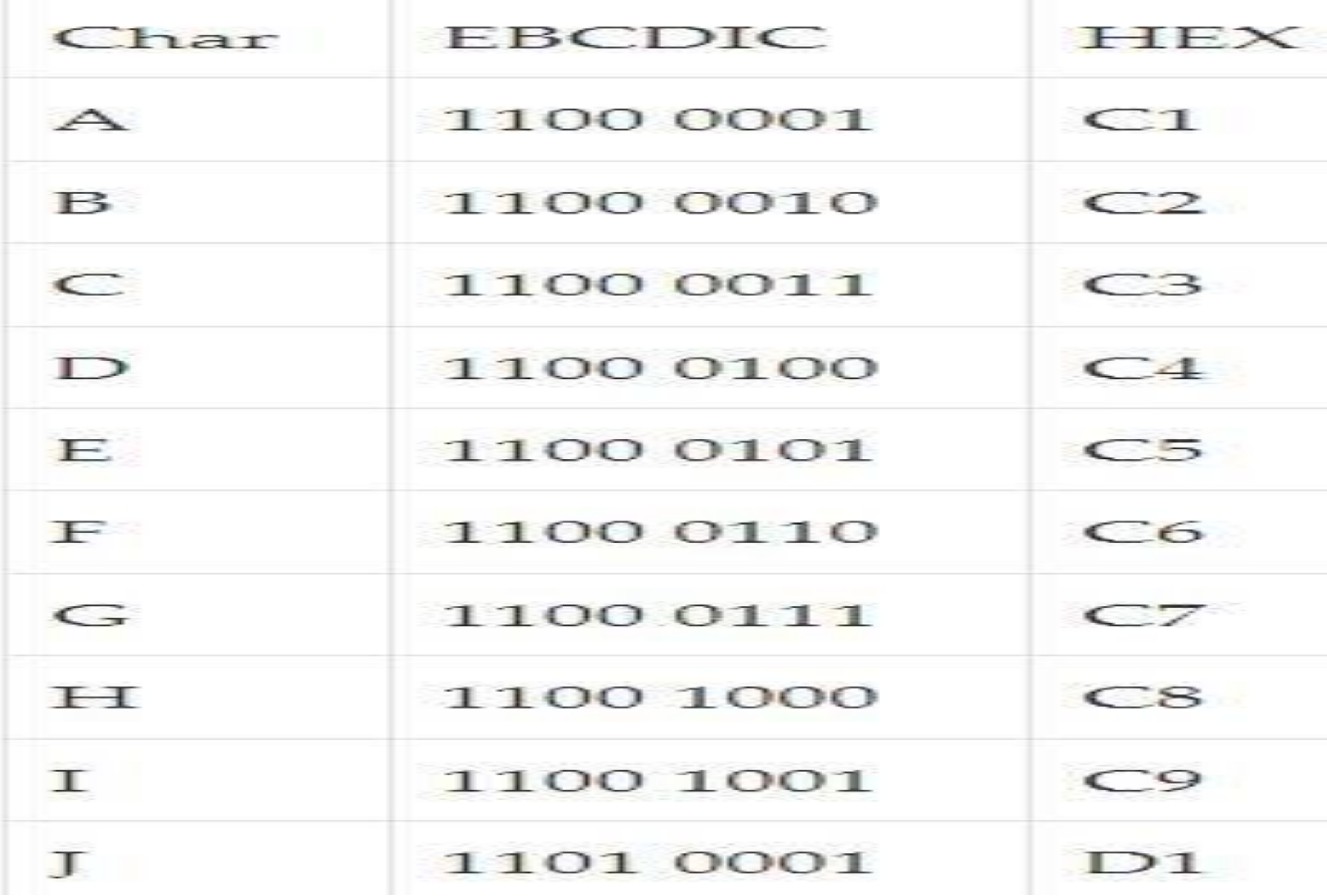
Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

EBCDIC character code

EBCDIC, in full extended binary-coded decimal interchange code, data-encoding system, developed by IBM and used mostly on its computers, that uses a unique eight-bit binary code for each number and alphabetic character as well as punctuation marks and accented letters and nonalphabetic characters. EBCDIC differs in several respects from Unicode and ASCII, the most widely used systems of encoding text, dividing the eight bits for each character into two four-bit zones, with one zone indicating the type of character, digit, punctuation mark, lowercase letter, capital letter, and so on, and the other zone indicating the value that is, the specific character within this type.

EBCDIC character code



Char	EBCDIC	HEX
A	1100 0001	C1
B	1100 0010	C2
C	1100 0011	C3
D	1100 0100	C4
E	1100 0101	C5
F	1100 0110	C6
G	1100 0111	C7
H	1100 1000	C8
I	1100 1001	C9
J	1101 0001	D1