

# Data Representation and Number system

## Introduction

A digital computer stores data in terms of digits (numbers) and proceeds in discrete steps from one state to the next. The states of a digital computer typically involve binary digits which may take the form of the presence or absence of magnetic markers in a storage medium , on-off switches or relays. In digital computers, even letters, words and whole texts are represented digitally.

Digital Logic is the basis of electronic systems, such as computers and cell phones. Digital Logic is rooted in binary code, a series of zeroes and ones each having an opposite value. This system facilitates the design of electronic circuits that convey information, including logic gates. Digital Logic gate functions include and, or and not. The value system translates input signals into specific output. Digital Logic facilitates computing, robotics and other electronic applications.

Digital Logic Design is foundational to the fields of electrical engineering and computer engineering. Digital Logic designers build complex electronic components that use both electrical and computational characteristics. These characteristics may involve power, current, logical function, protocol and user input. Digital Logic Design is used to develop hardware, such as circuit boards and microchip processors. This hardware processes user input, system protocol and other data in computers, navigational systems, cell phones or other high-tech systems.

## Numeric systems

The numeric system we use daily is the decimal system, but this system is not convenient for machines since the information is handled codified in the shape of on or off bits; this way of codifying takes us to the necessity of knowing the positional calculation which will allow us to express a number in any base where we need it.

## Radix number systems

A base of a number system or radix defines the range of values that a digit may have.

**In the binary** system or base 2, there can be only two values for each digit of a number, either a "0" or "1".

**In the octal** system or base 8, there can be eight choices for each digit of a number:

"0", "1", "2", "3", "4", "5", "6", "7".

For counting after 7

10,11,12,13,14,15,16,17,  
20,21,22,23,24,25,26,27,  
30,31,.....37,  
40,.....47,  
50,.....57,  
60,61,62,63,64,65,66,67,  
70,71,72,73,74,75,76,77,  
100,101,102,.....107  
110,111,112,.....117.

**In the decimal** system or base 10, there are ten different values for each digit of a

number: "0", "1", "2", "3", "4", "5", "6", "7", "8", "9".

**In the hexadecimal** system, we allow 16 values for each digit of a number:

"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", and "F".

Where "A" stands for 10, "B" for 11 and so on.

For counting after F

10,11,12,13,14,15,16,17,18,19,1A,1B,1C,1D,1E,1F,  
20,21,22,23,24,25,26,27,28,29,2A,2B,.....2F,  
30,31,.....3A,.....3F  
.  
.  
90,91,92,.....99,9A,9B,.....9F,  
A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,AA,AB,,AC,AD,AE,AF,  
B0,B1,.....B9,BA,.....BF,

# Conversion among radices

## - Convert from Decimal to Any Base

Let's express a **decimal number 1341 in binary notation**. Note that the desired base is 2, so we repeatedly divide the given decimal number by 2.

	Quotient	Remainder	
1341/2 =	670	1	-----+
670/2 =	335	0	-----+
335/2 =	167	1	-----+
167/2 =	83	1	-----+
83/2 =	41	1	-----+
41/2 =	20	1	-----+
20/2 =	10	0	-----+
10/2 =	5	0	-----+
5/2 =	2	1	-----+
2/2 =	1	0	-----+
1/2 =	0	1	---+                     (Stop when the quotient is 0)
			1 0 1 0 0 1 1 1 1 0 1 (BIN; Base 2)

Let's express the same decimal number 1341 in octal notation.

	Quotient	Remainder	
1341/8 =	167	5	-----+
167/8 =	20	7	-----+
20/8 =	2	4	-----+
2/8 =	0	2	---+       (Stop when the quotient is 0)
			2 4 7 5 (OCT; Base 8)

Let's express the same decimal number 1341 in hexadecimal notation.

	Quotient	Remainder	
1341/16 =	83	13	-----+
83/16 =	5	3	-----+
5/16 =	0	5	---+     (Stop when the quotient is 0)
			5 3 D (HEX; Base 16)

In conclusion, the easiest way to convert fixed point numbers to any base is to convert each part separately. We begin by separating the number into its integer and fractional part. The integer part is converted using the **remainder method**, by using a successive division of the number by the base until a zero is obtained. At each division, the remainder is kept and then the new number in the base r is obtained by reading the remainder from the last remainder upwards.

The conversion of the fractional part can be obtained by successively multiplying the fraction with the base. If we iterate this process on the remaining fraction, then we will obtain successive significant digit. This methods form the basis of the **multiplication methods** of converting fractions between bases

**Example.** Convert the decimal number 3315 to hexadecimal notation. What about the hexadecimal equivalent of the decimal number 3315.3

**Solution:**

	Quotient	Remainder	
3315/16 =	207	3	-----+
207/16 =	12	15	-----+
12/16 =	0	12	--+     (Stop when the quotient is 0)
		C F 3	(HEX; Base 16)

	Product	Integer Part	(HEX; Base 16)
0.3*16 =	<b>4.8</b>	4	0.4 C C C ...
0.8*16 =	<b>12.8</b>	12	-----+
0.8*16 =	<b>12.8</b>	12	-----+
0.8*16 =	<b>12.8</b>	12	-----+
:			-----+
:			-----+

Thus, 3315.3 (DEC) --> **CF3.4CCC...** (HEX)

## - Convert From Any Base to Decimal

Let's think more carefully what a decimal number means. For example, 1234 means that there are four boxes (digits); and there are 4 one's in the right-most box (least significant digit), 3 ten's in the next box, 2 hundred's in the next box, and finally 1 thousand's in the left-most box (most significant digit). The total is 1234:

Original Number:	1	2	3	4	
How Many Tokens:	1	2	3	4	
Digit/Token Value:	1000	100	10	1	
Value:	1000	+ 200	+ 30	+ 4	= 1234

or simply,  $1*1000 + 2*100 + 3*10 + 4*1 = 1234$

Thus, each digit has a value:  $10^0=1$  for the least significant digit, increasing to  $10^1=10$ ,  $10^2=100$ ,  $10^3=1000$ , and so forth.

Likewise, the least significant digit in a hexadecimal number has a value of  $16^0=1$  for the least significant digit, increasing to  $16^1=16$  for the next digit,  $16^2=256$  for the next,  $16^3=4096$  for the next, and so forth. Thus, 1234 means that there are four boxes (digits); and there are 4 one's in the right-most box (least significant digit), 3 sixteen's in the next box, 2 256's in the next, and 1 4096's in the left-most box (most significant digit). The total is:

$$1*4096 + 2*256 + 3*16 + 4*1 = 4660$$

**Example.** Convert 11001.0101 expressed in a Binary notation to decimal.

1	1	0	0	1	.	0	1	0	1
←						→			
4	3	2	1	0		-1	-2	-3	-4
$2^4$	$2^3$	$2^2$	$2^1$	$2^0$		$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$
= $1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 + 0*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4}$									
= $1*16 + 1*8 + 0*4 + 0*2 + 1*1 + 0*1/2 + 1*1/4 + 0*1/8 + 1*1/16$									
= $16 + 8 + 1 + 1/4 + 1/16$									
= $25 + 0.25 + 0.0625$									
= 25.3125									

**Example.** Convert 234.14 expressed in an octal notation to decimal.

2	3	4	.	1	4
←				→	
2	1	0		-1	-2
$8^2$	$8^1$	$8^0$		$8^{-1}$	$8^{-2}$
= $2*8^2 + 3*8^1 + 4*8^0 + 1*8^{-1} + 4*8^{-2}$					
= $2*64 + 3*8 + 4*1 + 1/8 + 4/64$					
= 128 + 24 + 4 + 0.125 + 0.0625 = 156.1875					

**Example.** Convert the hexadecimal number 4B3 to decimal notation. What about the decimal equivalent of the hexadecimal number 4B3.3?

**Solution:**

Original Number:	4	B	3	.	3	
How Many Tokens:	4	11	3		3	
		$16^2$	$16^1$	$16^0$	$16^{-1}$	
Digit/Token Value:	256	16	1		0.0625	
Value:	1024	+176	+ 3		+ 0.1875	<b>= 1203.1875</b>

**Example.** Convert 234.14 expressed in an octal notation to decimal.

**Solution:**

Original Number:	2	3	4	.	1	4	
How Many Tokens:	2	3	4		1	4	
Digit/Token Value:	64	8	1		0.125	0.015625	
Value:	128	+ 24	+ 4		+ 0.125	+ 0.0625	<b>= 156.1875</b>


## - Relationship between Binary - Octal and Binary-hexadecimal

As demonstrated by the table below, there is a direct correspondence between the binary system and the octal system, with three binary digits corresponding to one octal digit. Likewise, four binary digits translate directly into one hexadecimal digit.

Octal	Binary	Hexadecimal	Binary
0	000	0	0000
1	001	1	0001
2	010	2	0010
3	011	3	0011
4	100	4	0100
5	101	5	0101
6	110	6	0110
7	111	7	0111
		8	1000
		9	1001
		A	1010
		B	1011
		C	1100
		D	1101
		E	1110
		F	1111

**EXAMPLE 1:** convert (1010111100) binary to octal


<b>Binary</b>	<b>001</b>	<b>010</b>	<b>111</b>	<b>100</b>
<b>Octal</b>	<b>1</b>	<b>2</b>	<b>7</b>	<b>4</b>



The Octal number is (1274)

**EXAMPLE 2:** convert (101111010110) binary to octal

<b>Binary</b>	<b>101</b>	<b>111</b>	<b>010</b>	<b>110</b>
<b>Octal</b>	<b>5</b>	<b>7</b>	<b>2</b>	<b>6</b>



The Octal number is (5726)

With such relationship, In order to convert a binary number to octal, we partition the base 2 number into groups of three starting from the radix point, and pad the outermost groups with 0's as needed to form triples. Then, we convert each triple to the octal equivalent.

For conversion from base 2 to base 16, we use groups of four.

Consider converting  $10110_2$  to base 8:


$$10110_2 = 010_2 110_2 = 2_8 6_8 = 26_8$$

Notice that the leftmost two bits are padded with a 0 on the left in order to create a full triplet.

### Converting between Hexadecimal and Binary

**EXAMPLE 1:** convert (01011110101101010010) binary to hexadecimal

Binary	0101	1110	1011	0101	0010
hexa	5	E	B	5	2



The Hexadecimal number is (5EB52)

Now consider converting  $10110110_2$  to base 16:

$$10110110_2 = 1011_2 0110_2 = B_{16} 6_{16} = B6_{16}$$

(Note that 'B' is a base 16 digit corresponding to  $11_{10}$ . B is not a variable.)

The conversion methods can be used to convert a number from any base to any other base, but it may not be very intuitive to convert something like 513.03 to base 7. As an aid in performing an unnatural conversion, we can convert to the more familiar base 10 form as an intermediate step, and then continue the conversion from base 10 to the target base. As a general rule, we use the polynomial method when converting *into* base 10, and we use the remainder and multiplication methods when converting *out* of base 10.

Example : convert the Octal number (752) to Hexadecimal number .

Step1 :

Octal to Binary Conversion

$$\begin{array}{ccc} 7 & 5 & 2 \\ (111 & 101 & 010) \end{array}$$

So the binary equivalent 111101010

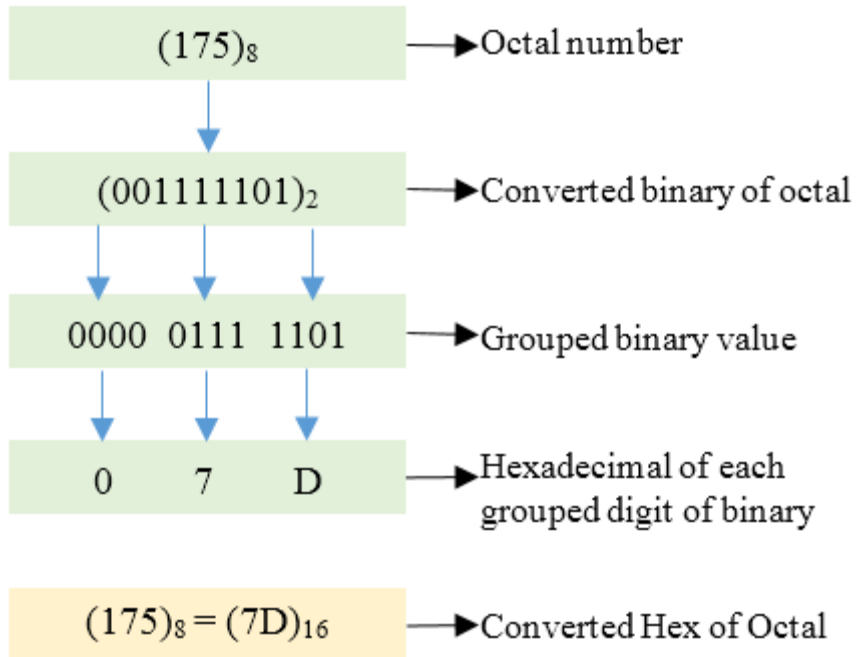
Step2 :

Binary to Hexadecimal Conversion

$$\begin{array}{ccc} 0001 & 1110 & 1010 \\ 1 & F & A \end{array}$$



Example



# Binary code

Internally, digital computers operate on binary numbers. When interfacing to humans, digital processors, e.g. pocket calculators, communication is decimal-based. Input is done in decimal then converted to binary for internal processing. For output, the result has to be converted from its internal binary representation to a decimal form. Digital system represents and manipulates not only binary number but also many other discrete elements of information.

## -Binary coded Decimal

In computing and electronic systems, binary-coded decimal (BCD) is an encoding for decimal numbers in which each digit is represented by its own binary sequence. Its main virtue is that it allows easy conversion to decimal digits for printing or display and faster decimal calculations. Its drawbacks are the increased complexity of circuits needed to implement mathematical operations and a relatively inefficient encoding. It occupies more space than a pure binary representation. In BCD, a digit is usually represented by four bits which, in general, represent the values/digits/characters 0-9

To BCD-encode a decimal number using the common encoding, each decimal digit is stored in a four-bit nibble.

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

Thus, the BCD encoding for the number 127 would be:

0001 0010 0111

The position weights of the BCD code are 8, 4, 2, 1. Other codes (shown in the table) use position weights of 8, 4, -2, -1 and 2, 4, 2, 1.

An example of a non-weighted code is the excess-3 code where digit codes is obtained from

their binary equivalent after adding 3. Thus the code of a decimal 0 is 0011, that of 6 is 1001, etc

**Decimal  
Digit**

	<b>8 4 2 1 Code</b>
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

it is very important to understand the difference between the conversion of a decimal number to binary and the binary coding of a decimal number. In each case, the final result is a series of bits. The bits obtained from conversion are binary digit. Bits obtained from coding are combinations of 1's and 0's arranged according to the rule of the code used. e.g. the binary conversion of 13 is 1101; the BCD coding of 13 is 00010011

The main advantage of binary coded decimal is that it allows easy conversion between decimal (base-10) and binary (base-2) form. However, the disadvantage is that BCD code is wasteful as the states between 1010 (decimal 10), and 1111 (decimal 15) are not used. Nevertheless, binary coded decimal has many important applications especially using digital displays.

# Character Representation

## ASCII Code (Decimal)

0 nul	16 dle	32 Sp	48 0	64 @	80 P	96 `	112 p
1 soh	17 dc1	33 !	49 1	65 A	81 Q	97 a	113 q
2 stx	18 dc2	34 "	50 2	66 B	82 R	98 b	114 r
3 etx	19 dc3	35 #	51 3	67 C	83 S	99 c	115 s
4 eot	20 dc4	36 \$	52 4	68 D	84 T	100 d	116 t
5 enq	21 nak	37 %	53 5	69 E	85 U	101 e	117 u
6 ack	22 syn	38 &	54 6	70 F	86 V	102 f	118 v
7 bel	23 etb	39 '	55 7	71 G	87 W	103 g	119 w
8 bs	24 can	40 (	56 8	72 H	88 X	104 h	120 x
9 ht	25 em	41 )	57 9	73 I	89 Y	105 i	121 y
10 nl	26 sub	42 *	58 :	74 J	90 Z	106 j	122 z
11 vt	27 esc	43 +	59 ;	75 K	91 [	107 k	123 {
12 np	28 fs	44 ,	60 <	76 L	92 \	108 l	124
13 cr	29 gs	45 -	61 =	77 M	93 ]	109 m	125 }
14 so	30 rs	46 .	62 >	78 N	94 ^	110 n	126 ~
15 si	31 us	47 /	63 ?	79 O	95 _	111 o	127 del

The characters between 0 and 31 are generally not printable (control characters, etc). 32 is the space character. Also note that there are only 128 ASCII characters. This means only 7 bits are required to represent an ASCII character. However, since the smallest size representation on most computers is a byte, a byte is used to store an ASCII character. The Most Significant bit(MSB) of an ASCII character is 0.

## ASCII Code (Hex)

00 nul	10 dle	20 sp	30 0	40 @	50 P	60 `	70 p
01 soh	11 dc1	21 !	31 1	41 A	51 Q	61 A	71 q
02 stx	12 dc2	22 "	32 2	42 B	52 R	62 B	72 r
03 etx	13 dc3	23 #	33 3	43 C	53 S	63 C	73 s
04 eot	14 dc4	24 \$	34 4	44 D	54 T	64 D	74 t
05 enq	15 nak	25 %	35 5	45 E	55 U	65 E	75 u
06 ack	16 syn	26 &	36 6	46 F	56 V	66 F	76 v
07 bel	17 etb	27 '	37 7	47 G	57 W	67 G	77 w
08 bs	18 can	28 (	38 8	48 H	58 X	68 H	78 x
09 ht	19 em	29 )	39 9	49 I	59 Y	69 I	79 y
0a nl	1a sub	2a *	3a :	4a J	5a Z	6a J	7a z
0b vt	1b esc	2b +	3b ;	4b K	5b [	6b K	7b {
0c np	1c fs	2c ,	3c <	4c L	5c \	6c L	7c
0d cr	1d gs	2d -	3d =	4d M	5d ]	6d M	7d }
0e so	1e rs	2e .	3e >	4e N	5e ^	6e N	7e ~
0f si	1f us	2f /	3f ?	4f O	5f _	6f O	7f del

The difference in the ASCII code between an uppercase letter and its corresponding lowercase letter is  $20_{16}$ . This makes it easy to convert lower to uppercase (and back) in

