# **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.

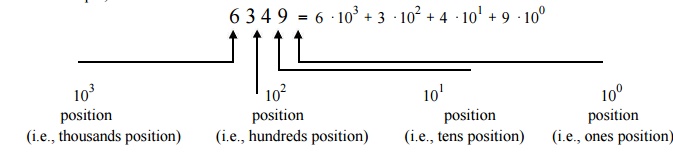
Say we have three numbers – 734, 971 and 207. The value of 7 in all three numbers is different−

* In 734, value of 7 is 7 hundreds or 700 or 7 × 100 or 7 × 102
* In 971, value of 7 is 7 tens or 70 or 7 × 10 or 7 × 101
* In 207, value 0f 7 is 7 units or 7 or 7 × 1 or 7 × 100

The weightage of each position can be represented as follows −



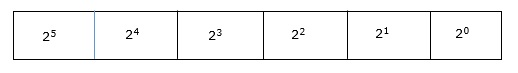
In digital systems, instructions are given through electric signals; variation is done by varying the voltage of the signal. Having 10 different voltages to implement decimal number system in digital equipment is difficult. So, many number systems that are easier to implement digitally have been developed.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



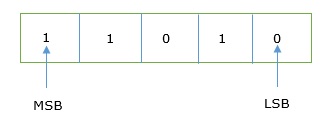
Binary Number System

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.



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



And decimal equivalent of this number is sum of product of each digit with its positional value.

110102 = 1×24+ 1×23 + 0×22 + 1×21 + 0×20

= 16 + 8 + 0 + 2 + 0

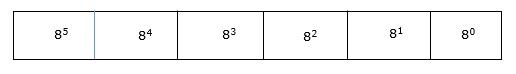
= 2610

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

* 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, as shown here −



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

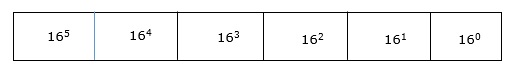
7268 = 7×82+ 2×81 + 6×80

= 448 + 16 + 6

= 47010

Hexadecimal Number System

**Octal 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 powers of 16, as shown here −



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

27FB16 = 2×163 + 7×162 + 15×161 + 10×160

= 8192 + 1792 + 240 +10

= 1023410

Number System Relationship

The following table depicts the relationship between decimal, binary, octal and hexadecimal number systems.

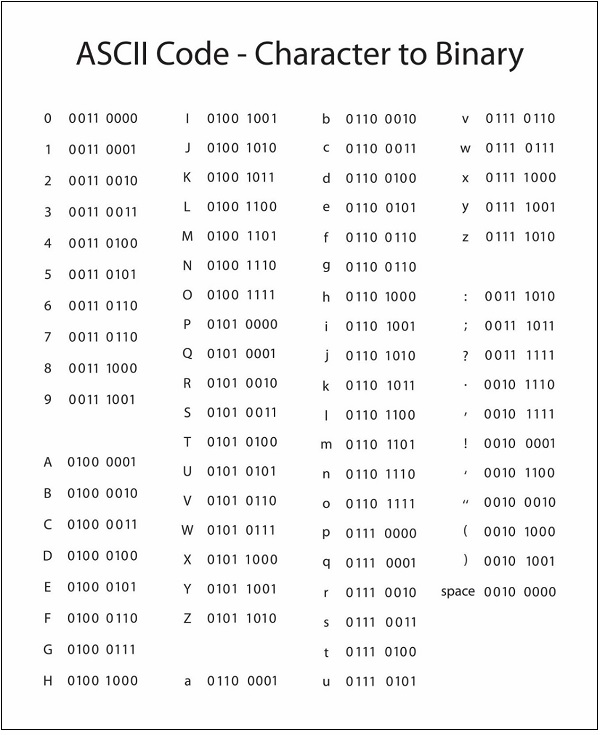
|  |  |  |  |
| --- | --- | --- | --- |
| **HEXADECIMAL** | **DECIMAL** | **OCTAL** | **BINARY** |
| 0 | 0 | 0 | 0000 |
| 1 | 1 | 1 | 0001 |
| 2 | 2 | 2 | 0010 |
| 3 | 3 | 3 | 0011 |
| 4 | 4 | 4 | 0100 |
| 5 | 5 | 5 | 0101 |
| 6 | 6 | 6 | 0110 |
| 7 | 7 | 7 | 0111 |
| 8 | 8 | 10 | 1000 |
| 9 | 9 | 11 | 1001 |
| A | 10 | 12 | 1010 |
| B | 11 | 13 | 1011 |
| C | 12 | 14 | 1100 |
| D | 13 | 15 | 1101 |
| E | 14 | 16 | 1110 |
| F | 15 | 17 | 1111 |

ASCII

Besides numerical data, computer must be able to handle alphabets, punctuation marks, mathematical operators, special symbols, etc. that form the complete character set of English language. The complete set of characters or symbols are called alphanumeric codes. The complete alphanumeric code typically includes −

* 26 upper case letters
* 26 lower case letters
* 10 digits
* 7 punctuation marks
* 20 to 40 special characters

Now a computer understands only numeric values, whatever the number system used. So all characters must have a numeric equivalent called the alphanumeric code. The most widely used alphanumeric code is American Standard Code for Information Interchange (ASCII). ASCII is a 7-bit code that has 128 (27) possible codes.



ISCII

ISCII stands for **Indian Script Code for Information Interchange**. IISCII was developed to support Indian languages on computer. Language supported by IISCI include Devanagari, Tamil, Bangla, Gujarati, Gurmukhi, Tamil, Telugu, etc. IISCI is mostly used by government departments and before it could catch on, a new universal encoding standard called **Unicode** was introduced.

Unicode

Unicode is an international coding system designed to be used with different language scripts. Each character or symbol is assigned a unique numeric value, largely within the framework of ASCII. Earlier, each script had its own encoding system, which could conflict with each other.

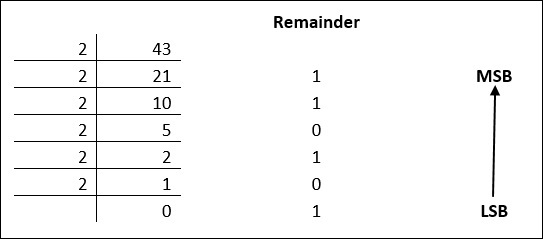
In contrast, this is what Unicode officially aims to do − *Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language*.

# **Number System Conversion**

As you know decimal, binary, octal and hexadecimal number systems are positional value number systems. To convert binary, octal and hexadecimal to decimal number, we just need to add the product of each digit with its positional value. Here we are going to learn other conversion among these number systems.

Decimal to Binary

Decimal numbers can be converted to binary by repeated division of the number by 2 while recording the remainder. Let’s take an example to see how this happens.



The remainders are to be read from bottom to top to obtain the binary equivalent.

4310 = 1010112

**Decimal fractions**

For decimal fractions, the fractional part has to be multiplied by 2 successively and collecting the carries from top to bottom. The multiplication can be repeated till the fractional part becomes zero. If the fractional part is not zero after four or five steps the process can be stopped and we have to be satisfied with the nearest value.

For example the decimal fraction .625 is converted into binary as

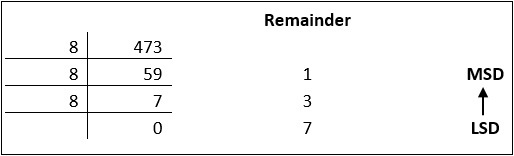
* 0.625 × 2 = 1.250; carry is 1 (MSB)
* 0.250 × 2 = 0.500; carry is 0
* 0.500 × 2 = 1.000; carry is 1 (LSB)
* Therefore (0.625)10 = (0.101)2

Try out:

* Convert (107.6875)
* Convert (52.4)

Decimal to Octal

Decimal numbers can be converted to octal by repeated division of the number by 8 while recording the remainder. Let’s take an example to see how this happens.

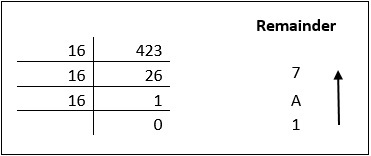


Reading the remainders from bottom to top,

47310 = 7318

Decimal to Hexadecimal

Decimal numbers can be converted to octal by repeated division of the number by 16 while recording the remainder. Let’s take an example to see how this happens.



Reading the remainders from bottom to top we get,

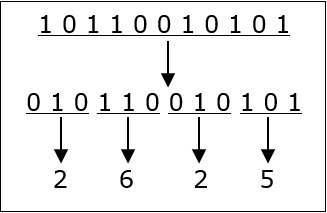
42310 = 1A716

Binary to Octal and Vice Versa

To convert a binary number to octal number, these steps are followed −

* Starting from the least significant bit, make groups of three bits.
* If there are one or two bits less in making the groups, 0s can be added after the most significant bit
* Convert each group into its equivalent octal number

Let’s take an example to understand this.



101100101012 = 26258

* (10 101)2 = (010 101)2 = (25)8
* (10111.1) = (010 111 . 100) = (2 7 . 4)8

To convert an octal number to binary, each octal digit is converted to its 3-bit binary equivalent according to this table.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Octal Digit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Binary Equivalent | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

546738 = 1011001101110112

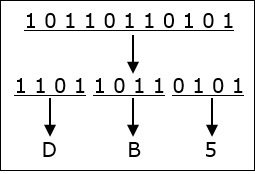


Binary to Hexadecimal

To convert a binary number to hexadecimal number, these steps are followed −

* Starting from the least significant bit, make groups of four bits.
* If there are one or two bits less in making the groups, 0s can be added after the most significant bit.
* Convert each group into its equivalent octal number.

Let’s take an example to understand this.



101101101012 = DB516

**Binary Fraction to hexadecimal**

* (1010.1101)2 = (A.D)H
* (110.101) = (0110.1010) = (6.A)H
* (1110.11)2 = (1110.1100) = (E.C)H

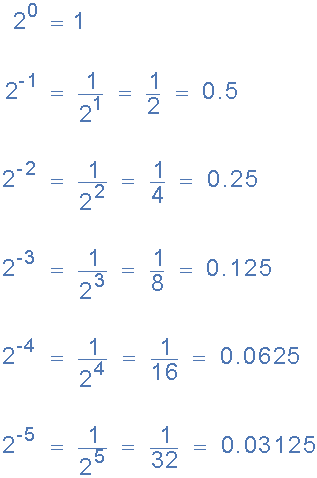
To convert an octal number to binary, each octal digit is converted to its 3-bit binary equivalent.

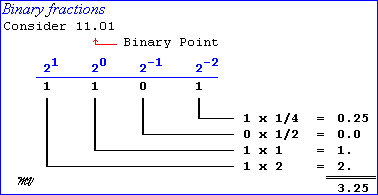
Binary to Decimal

The binary numbering system is a base-2 numbering system which contains only two digits, a “0” or a “1”. Thus each digit of a binary number can take the “0” or the “1” value with the position of the 0 or 1 indicating its value or weighting.

Similar to decimal fractions, binary numbers can also be represented as unsigned fractional numbers by placing the binary digits to the right of the decimal point or in this case, binary point. Thus all the fractional digits to the right of the binary point have respective weightings which are negative powers of two, creating a binary fraction. In other words, the powers of 2 are negative.

So for the fractional binary numbers to the right of the binary point, the weight of each digit becomes more negative giving: 2-1, 2-2, 2-3, 2-4, and so on as shown.





Now lets suppose we have the following binary number of: 1101.01112, what will be its decimal number equivalent.

1101.0111 = (1×23) + (1×22) + (0×21) + (1×20) + (0×2-1) + (1×2-2) + (1×2-3) + (1×2-4)

= 8 + 4 + 0 + 1 + 0 + 1/4 + 1/8 + 1/16

= 8 + 4 + 0 + 1 + 0 + 0.25 + 0.125 + 0.0625 = 13.437510

**Hexadecimal to decimal**

* Convert D5H to decimal
* (D5)H = (13×161 + 5×160)

= (13 ×16 + 5× 1)

= (208+5)

= 21310

**Hexadecimal to binary**

* Convert (25)H to Binary
* 25H = (0010 0101)2
* Convert (3A.7) to Binary
* (3A.7)H = (0011 1010. 0111)2
* Convert (CD.E8) to Binary
* (CD.E8) = (1100 1101. 1110 1000)2

**OCTAL TO DECIMAL**

|  |  |  |
| --- | --- | --- |
| Decimal | Octal | Binary |
| 0 | 0 | 000 |
| 1 | 1 | 001 |
| 2 | 2 | 010 |
| 3 | 3 | 011 |
| 4 | 4 | 100 |
| 5 | 5 | 101 |
| 6 | 6 | 110 |
| 7 | 7 | 111 |

* Convert (75)8 to decimal
* (75)8 = (7×81 + 5×80)

= (7×8 + 5× 1)

= (56+5)

= 6110

**Try Out:**

* Binary to decimal
* (11101)
* (111.11)
* Hex to decimal
* (50)H
* (3EF)H
* Decimal to Hex
* (48)10
* (1024)10
* Hex to binary (99)H , (FF.E6)H