**Number Systems**

Table of Contents

[Introduction 2](#_Toc65539855)

[Decimal Number System 3](#_Toc65539856)

[Converting to Decimal Number System 3](#_Toc65539857)

[Converting to Other Bases Systems 4](#_Toc65539858)

[Binary Arithmetic 6](#_Toc65539859)

[Addition 6](#_Toc65539860)

[Subtraction 6](#_Toc65539861)

## Introduction

The computer translates any letters or digits we type into numbers as it can only understand numbers. It understands the positional number system, where different values are represented by the digit and the position it occupies.

## Decimal Number System

Humans are used to using the decimal number system. It is read from right to left, multiplying each number by to power its position, starting with , and adding them together.

So, is read as .

This system uses the digits – . In any number system, the number is read in a similar manner, from right to left, multiplying each number by its base to the power of the position of the number. Each system uses numbers starting from and ending just digit before its base. The binary number system used by all computers is a base , using the digits and . The octal number system is a base , using digits – , and the hexadecimal number system is a base , using digits – and characters – . For the Hexadecimal Number System, the characters – represent numbers – resepectively.

### Converting to Decimal Number System

### Converting to Other Bases Systems

Generally, the number in the decimal numbering system is divided by the base of the new numbering system, until 0 has been reached. For each step, the remainder is the value of the position in the new numbering system, read from right to left.

remainder

remainder

remainder

remainder

remainder

Thus,

For number systems that are multiples of each other (like , and ), a shortcut method can be used.

From binary to octal for example, the decimal equivalent of each -digit set in binary represents digit in the octal system.

The same can be done for conversion of the -base system to the -base system.

For conversion of octal to binary, the opposite is done.

## Binary Arithmetic

The following is how computers go through addition and subtraction.

### Addition

Carry

Result

### Subtraction

For the computer, subtraction is just the addition of a negative number.

Negative numbers are stored in two ways, using the Signed Magnitude System and the Two’s Complement System.

In the Signed Magnitude System, the leftmost digit is reserved to store a sign. For a -bit system, the maximum value is normally . If the left most digit is reserved, the range becomes to , thus still allowing numbers.

In the 2’s Complement System, the leftmost value is fixed as a negative place value. For a -bit system, this means . Thus, .

Conversion of a positive number a negative number in the 2’s Complement system requires going through the 1’s Complement system.

This is reverted to the 1’s Complement System, by flipping the value of each position, thus giving . is then added to the least significant bit (rightmost bit).

Thus, is the result in the 2’s Complement system.

Sometimes, addition of numbers using the 2’s Complement System can cause overflow.

The least value that can be stored in a -bit system is , so cannot be stored. This is proven in the binary code as the most significant bit (the leftmost bit) must still carry forward a , but there is no place for it to carry the to. This is called an overflow, and it will cause an error that must be fixed by the programmer manually.

A quick way to check for an overflow is to check the MSB position. If the result is negative, the MSB position should have value , but it has the value in the above example, which is proof that it is wrong. The opposite is true for positive results.