# Digital

## Number System

A system of writing , expressing or representation of a number of certain type in known as number system. But more logical meaning of a Number System is Collection of symbols usually numbers on which we can perform operation like Addition, Subtraction , Multiplication and Division and get a singular result.

For Example 2 + 3 = 5, But what is ‘Hello’ + ‘World’ Or ‘Hello’ - ‘World’ is nothing

There are some types of number system

1. Binary
2. Hexadecimal
3. Decimal
4. Octal

### Base of Number System

Each system or collection has some finite amount of unique characters, Like when we are talking about Decimals they have 10 unique character [0 – 9] and Hexadecimals have 16 [0 – F] characters.

So when we are talking about base of a number It is equals to amount of unique characters they have.

For Example

Decimals [0 – 9 ] => 10 unique character so they has base 10,

Hexadecimal [0 – F] => 16 unique characters so they has base 16

Common properties of every number system is that All numbers start with 0 and ends with Base -1

### Calculations

For calculation or performing some arithematical operations . It is required that all numbers are under same system or their bases are same

For Example we will not calculate 0b 0111 1111 and 0x EF

So for calculation of two different numbers system we must convert their bases under same number system

### Conversion Of Number System

All number systems can be converted from 1 form to another this is further discussed when we are discussing each Number System specifically.

### Counting in Number System

Counting uses the **B (**base of number system**)** amount symbols **first** through **last**. Counting begins with the incremental substitution of the least significant digit. When the available symbols for this position are exhausted, the least significant digit is reset to **first**, and the next digit of **higher significance** (one position to the left) is incremented (overflow), and incremental substitution of the low-order digit resumes. This method of reset and overflow is repeated for each digit of significance.

For Example:

Binary [0 - 1] => 0000 , 0001, 0010 , 0011, 0100, 1000 – 1111

Decimal [0 – 9] => 00, 01, 02, 03, … , 09 , 10 , … , 19 , 20 , … 99

### Multiplication in Number System

A simple rule for multiplication of two digits in any base is to multiply them in decimal. If the product is less than the **base**, then we take it as the result. If the product is greater than the base we divide it by the base and take the remainder as the least significant digit. The quotient is taken as carry in the next significant digit.

**For example, (3)4 × (1)4 = (3)4 but (3)4 × (2)4 = (12)4 since 3 × 2 = 6 is decimal and division of 6 by 4 has the remainder 2 and quotient 1.**

### Sign number VS Un-Sign numbers

Sign number are those numbers which have digits along with a plus or minus Sign. I.E Numbers from – Infinity to + Infinity are called sign numbers. While Un-Sign number doent have a sign so they have only positive numbers from 0 to + Infinity.

## Binary

Binary numbers are collection of 0s and 1s and its base is 2, Each of its digit is called bit.

Because of its straight forward implementation In circuitry electronics by using logic gates. It is widely used in Digital Electronics and in Programming Languages which are closer to Electronics like Machine Language and Assembly Language.

Reason for its usage in electronics is that electronics signals only have two states [on ,off] and in binary we have [1,0].

### Sign and Un-Sign Binary Numbers

As we know that binary numbers only contains 0s and 1s, (means no Sign -,+) that dosn’t means binary numbers only contains Un- Sign numbers, they also contains – ve numbers and also +ve numbers.

In Sign numbers the most significance bit is called sign bit if is 0 the number is positive and if is 1 then number is negative

Lets look at example:

0100 , this is a positive number = 4

1100 , this is a negative number = -4

In the above example we saw that 1100 is a negative number but this number is also a positive 12.

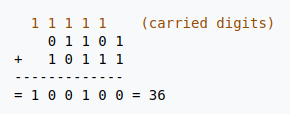
So It is upon us that how we treat binary number as sign number or un-sign number

If we treat number as sign than most significance bit represent a sign.

### Arithmetic Operators

**Addition**  It is a simplest operation in binary.

First we do addition with least significance bit of both numbers,  
If either of them is 1 then we write 1 if both is 0 we write 0 , if both are 1 we write 1 and carry shifted towards left bit.



**Substraction**  The formula of subtraction is simple. If A and B are two sign binary digits then

A-B = A + NOT B + 1

A = 0011 = 3

B = 0010 = 2

A - B = A NOT B + 1

A - B = 0011 + NOT 0010 + 1

A - B = 0011 + 1101 + 1

0011

1101

+

-------

10000 because we are using only 4 digit numbers so we neglect 5th

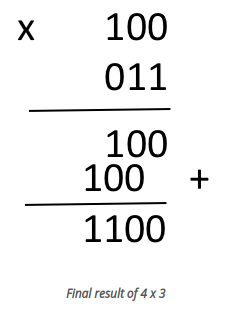
A - B = 0000 + 0001

A - B = 1;

### Multiplication Of Binary

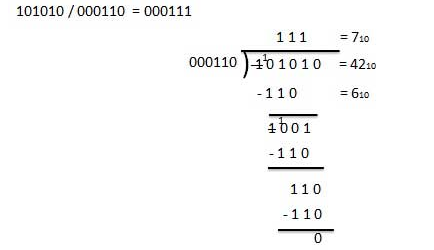
The Process of multiplying two numbers with each other is that

1. Multiplication of two bits is done by doing bitwise AND operation
2. Take LSB (Least Significant bit) from 2ndnumber multiply it bit by bit from 1st number.
3. Write down answer 1 line
4. Repeat the process until you iterate whole bits of 2nd number and writing answer line by line and also keep adding 0 to the tail of number (append right side of LSB)
5. finally add all answer you are writing line by line



### Divisions in Binary Numbers

Divisions in Binary is same as Decimals. It is also called long division process.



**In Binary there is a lot of things we will discuss some of them in Boolean Algebra Topic**

## Octal

Octal numbers are 3 bit representation of binary numbers.

Octal numbers are from 0 – 7 having base 8, Octal numbers can be made from binary by groping three bits 1111111 = 001111111 = 001 , 111, 111 = 1778

### Conversion of numbers to octal

In Mathematics we know that 16 , 8 , 2 are of same base 2. means we will write 16 , 8 , 2 as 24  , 23 , 21  respectively. And we also know that in number system Hexadecimal, Octal , Binary have bases 16 , 8 , 2 respectively.  
So it is correct to say that converting Hexadecimal, Octal , Binary to each other is easy then converting them to decimal.

# Binary to Octal Conversion

As we already discuss converting binary to decimal is easy by grouping bits into 3 digits

1111111 = 001111111 = 001 , 111, 111 = 1778

Why we are grouping them to 3 digits? Or From Where 3 is come from?

As we know binary has base 2 and octal has base 8 and 23  = 8, because of 2 exp 3 is equal to 8.

We also use wise versa process to convert octal to decimal

# Hexadecimal to Octal Conversion

The easy process is that First convert Hexadecimal to binary then binary to Octal

Converting Hexadecimal to binary is again easy we can write each digit of hexadecimal to 4 bit binary and finally combine them.

For Example : FF as we know F = 1111 so we can write 1111 1111.

now group these binary number to 3 bits 011 111 111 = 3778

# Decimal to Octal

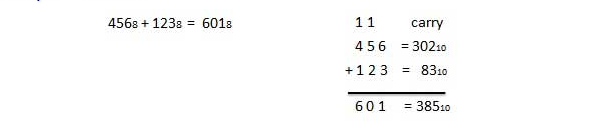
Converting Decimal to octal is simple just divide the decimal number and its quotient and append the reminder after MSB.

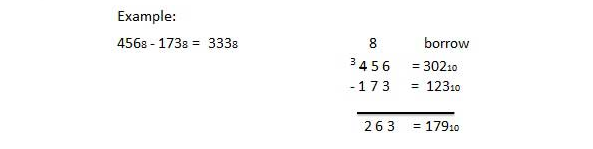
Here is an example of using repeated division to convert 1792 decimal to octal:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Decimal Number | Operation | Quotient | Remainder | Octal Result |
| 1792 | ÷ 8 = | 224 | 0 | 0 |
| 224 | ÷ 8 = | 28 | 0 | 00 |
| 28 | ÷ 8 = | 3 | 4 | 400 |
| 3 | ÷ 8 = | 0 | 3 | 3400 |
| 0 | done |  | | |

Arithmetic in Octal  
Arithmetic in Octal is same as Arithmetic in decimal, or binary or in other number system.

**Addition:**  Addition is same as decimals but after seven there must be 10 not 8

**Subtraction:** In subtraction all rules are same in all number system. The only variation is in binary we borrow group of 2, 8, 10 and 16 for binary, octal , decimals and hexadecimal respectively.

**Multiplication:**

As we discussed before the rules of mathematics in any number system in Multiplication in Number System.

Here is an example of multiplication in octal number system.

We have 6 × 3 = 18 in decimal, which when divided by 8 gives a remainder 2 and carry 2. Again 6 × 2 = 12 in decimal, and 12 + 2 = 14. This when divided by 8 gives a remainder 6 and a carry 1.

|  |  |
| --- | --- |
| Hence 68 × 238 = 1628 | 6 × 3 = 18  18/8 = 2 with remainder 2 → l,s,d,   6 × 2 = 12 + 2 (carry) = 14  14/8 = 1 with remainder 6. |

## Hexadecimals

There for Hexadecimal are more human readable 4 bit representation of binary numbers in computer science.

Because computer are used to store data in memory registers, ram, … Counted on bytes, words

and Hexadecimal are perfect representation as for word we use single digit hexadecimal

and for byte we use double digit hexadecimal.

### Conversion To Hexadecimal

# Hexadecimal - Binary

Conversion Between Hexadecimal and binary is easy

Write binary in group of 4 bits and convert each 4 bit in its Hexadecimal equilent.

For Example: 1010 1111 = AF

Its wise versa convert Hexadecimal to binary.

# Hexadecimal – Octal

Again conversion from Hexadecimal to Octal is an easy stuff.

Convert Octal to Binary , Group bits by 4 and convert to hexadecimal

2578  = 010 101 1112 = 0101011112 = 0000 1010 11112 = 0AF16  = AF16

# Hexadecimal – Decimals

There is a formula to convert Hexadecimal to Decimal

Hn-1 x 16n-1 + Hn-2 x 16n-2 + …..+ H2 x 162  + H1 x 161 + H0 x 160

Put Every digit of decimal equivalent Hexadecimal into Hn with respect to its position, put LSB to position 0 and you will get its decimal equivalent.

For Example: Here is a hexadecimal number 0d47a116

In this case positions are 05 d4 43 72 a1 10 this is a six digit number h5 – h0

convert each digit into its decimal equivalent 05 134 43 72 101 10

now put each digit into formula 0 \* 165  + 13\*164 + 4\*163 + 7\*162 + 10\*161 + 1\*160 = 87030510

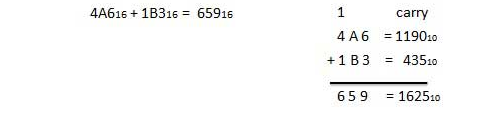
The method to convert decimal to hexadecimal is Repeating Divide and Remainder method

Lets convert base 10 188 to base 16

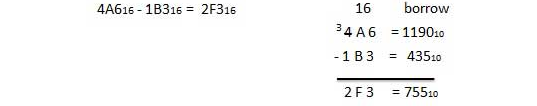
|  |  |  |
| --- | --- | --- |
| **DIVISION** | **RESULT** | **REMAINDER (in HEX)** |
| 188 / 16 | 11 | C (12 decimal) |
| 11 / 16 | 0 | B (11 decimal) |
|  |  |  |
| ANSWER |  | BC |

### Arithmetic Operations in Hexadecimal

Hexadecimal borrows same rules for arithmetics in hexadecimal

**Addition:**

**Substraction:**

****

## Decimals

Decimal number system is that we are using it in our daily life. It has [0-9] 10 unique symbols to represent an amount , Therefore its base in 10.

# Conversion To Decimals

For **n** amount of digit having **Dn-1**to **D0**having base **b**

Dn-1 x Bn-1 + Dn-2 x Bn-2 + …..+ D2 x B2  + D1 x B1 + D0 x B0

For Example :

10112 = 1x 23 + 0 x 22 + 1 x 21 + 1 x 20 = 8+0+2+1 = 11

FF16 = 15 x 161 + 15 x 160 = 240 + 16 = 255

755**8** = 7x 82 + 5 x 81 + 5 x 80 = 448+40+5 = 493

# Conversion From Decimals

To convert any number system from decimal is same repeated dividing and writing remainder process

For any decimal number **d** convertingto base **B.**

Take a **d** and divide it by **B** and write symbol for equivalent remainder in LSB

continue Dividing **d** and writing its remainder right along with LSB to get your converted number.

See Example in Conversion To Hexadecimal

## Gates

In every number system we have some kind operators. In binary we have two types of operators

1. Arithmetic Operators
2. Bitwise Operators

Bitwise operators or logic gates are used when we have only 2 states of a symbol I.e [true, false] Or [0 , 1]

There are some brief defination of bitwise operators

**OR**  If either operand is 1 it will give 1 otherwise 0

**AND**  If either operand is 0 it will give 0 otherwise 1

**NOT**  It is a singular operator it change the state of symbol Not 1 will give 0 and Not 0 will give 1

**XOR**  If operands are different it will give 1 otherwise 0

**XNOR**  If operands are same it will give 1 otherwise 0

## Boolean Algebra

Boolean Algebra is used to simplify circuit logic in digital circuits, also used in software logic and design in conditions.

## Rule in Boolean Algebra

Following are the important rules used in Boolean algebra.

* Variable used can have only two values. Binary 1 for HIGH and Binary 0 for LOW.
* Complement of a variable is represented by an overbar (-). Thus, complement of variable B is represented as. Thus if B = 0 then Not B= 1 and B = 1 then Not B = 0.
* ORing of the variables is represented by a plus (+) sign between them. For example ORing of A, B, C is represented as A + B + C.
* Logical ANDing of the two or more variable is represented by writing a dot between them such as A.B.C. Sometime the dot may be omitted like ABC.

## Postulates and Basic Laws of Boolean Algebra

**Or Laws:**

x + 0 = x

x + 1 = 1

x + x = x

x + x’ = 1

**And Laws:**

x.1 = x

x.0 = 0

x.x = x

x.x’ = 0

## Boolean Laws

# Commutative law

This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.

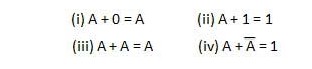
# Distributive law

# 

# And Law



# Or Law



# INVERSION law

This law uses the NOT operation. The inversion law states that double inversion of a variable results in the original variable itself.



# Boolean Expression

Boolean algebra deals with binary logic, A boolean function is described by boolean expresion



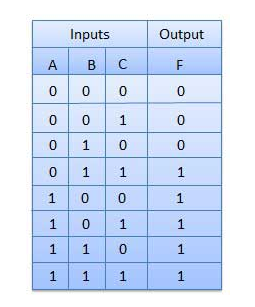
## Truth Table Formation

A truth table represents a table having all combinations of inputs and their corresponding result.

Consider the following equation F(A,B,C) = A + BC

In this case the answer is high if A is high or BC or Both, BC is high if both is high

For above equation we have following truth table.



### DeMorgan’s Theorem

1. This theorem is usefull in finding the complement of boolean function.
2. It states that
3. The Complement of logical Oring of atleast two variable is equal to the complement of complement and to two variables

DeMorgan’s theorem with 2 Boolean variables x and y can be represented as

(x + y)’ = x’.y’

The dual of the above Boolean function is

(x.y)’ = x’ + y’

### Duality Theorem

|  |  |
| --- | --- |
| Group1 | Group2 |
| x + 0 = x | x.1 = x |
| x + 1 = 1 | x.0 = 0 |
| x + x = x | x.x = x |
| x + x’ = 1 | x.x’ = 0 |
| x + y = y + x | x.y = y.x |
| x + (y + z) = (x + y) + z | x.(y.z) = (x.y).z |
| x.(y + z) = x.y + x.z | x + (y.z) = (x + y).(x + z) |

### Methods to simplify the boolean function

The methods used for simplifying the Boolean function are as follows −

* Karnaugh-map or K-map, and
* NAND gate method.

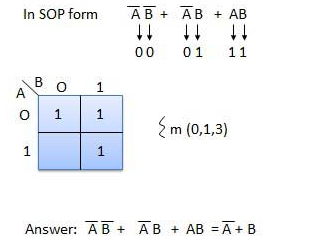
### Karnaugh-map or K-map

Boolean Logic , theorems and De-Morgan’s theorems are useful in manipulating the logic. We can realize the no of gates used in this equation.  
We can reduce the requirement of gates using K-Map method

# Sum of Products (SOP) Form

Its name describes it very well when equation is written in such form where we should first done anding of variable than oring it.

For Example

AB + AC + BC

Example 1

Let us **simplify** the Boolean function, f = p’qr + pq’r + pqr’ + pqr

We can simplify this function in two methods.

Method 1

Given Boolean function, f = p’qr + pq’r + pqr’ +pqr.

**Step 1** − In first and second terms r is common and in third and fourth terms pq is common. So, take the common terms by using **Distributive law**.

⇒ f = (p’q + pq’)r + pq(r’ + r)

**Step 2** − The terms present in first parenthesis can be simplified to Ex-OR operation. The terms present in second parenthesis can be simplified to ‘1’ using **Boolean postulate**

⇒ f = (p ⊕q)r + pq(1)

**Step 3** − The first term can’t be simplified further. But, the second term can be simplified to pq using **Boolean postulate**.

⇒ f = (p ⊕q)r + pq

Therefore, the simplified Boolean function is **f = (p⊕q)r + pq**