UNIT 1

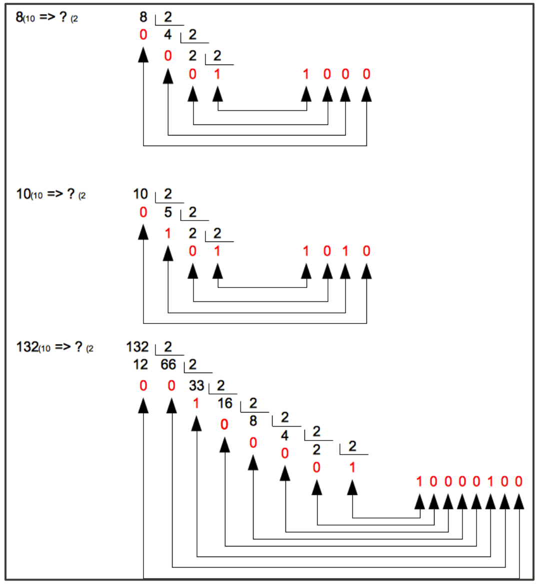
INFORMATION REPRESENTATION

NUMERAL SYSTEMS

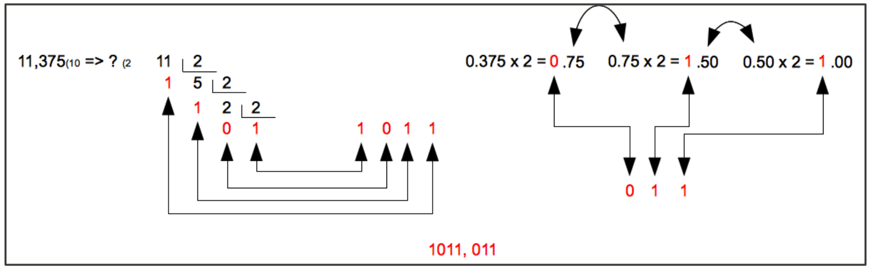
A numeral system is a set of sorted symbols used to represent quantities. The number of symbols is called system base.

- Binary
- Decimal
- Octal
- Hexadecimal

DECIMAL TO BINARY



Number with fractional part:



BINARY TO DECIMAL

$$N = a_{n-1}B^{n-1} + a_{n-2}B^{n-2} + \ldots + a_1B^1 + a_0B^0 + a_{-1}B^{-1} + \ldots + a_{-p}B^{-p} = \sum_{i=-p}^{n-1} a_i\, B^i$$

$101001_{(2} \Rightarrow\ ?_{(10}$

$$1\,0\,1\,0\,0\,1 \Rightarrow 1{*}2^5 + 0{*}2^4 + 1{*}2^3 + 0{*}2^2 + 0{*}2^1 + 1{*}2^0 = 41$$

Number with fractional part:

$10{,}01_{(2} \Rightarrow\ ?_{(10}$

$$1\,0,\,0\,1 \Rightarrow 1{*}2^1 + 0{*}2^0 + 0{*}2^{-1} + 1{*}2^{-2} = 2{,}25$$

MAXIMUM NUMBER

One of the typical questions when handling a binary number is to know what is the maximum decimal value that can be represented by a certain bits number.

The **answer** is easy: 2^n , where **n** is the bit number.

For instance, with 4 bits we can represent 16 values, from 0 to 15 (0000-1111)

OPERATION WITH BINARY

Addition:

0 + 0 = 0

1 + 0 = 1

0 + 1 = 1

1 + 1 = 0 (carry 1)

```
     11               111111
   10011010              1011
+  01001100         +  111101
   11100110           1001000
```

Subtraction:

0 - 0 = 0

1 - 0 = 1

0 - 1 = 1 (carry 1 to *subtrahend*)

1 - 1 = 0

```
  101101             11101
  1                   11
-  10101          -  00111
  011000             10110
```

OPERATION WITH BINARY

Multiplication:

0 * 0 = 0

1 * 0 = 0

0 * 1 = 0

1 * 1 = 1

Division:

0 / 0 = Undefined

1 / 0 = Boundless

1 / 1 = 1

0 / 1 = 0

```
101010    ⌊110
-110       111
 1001
  1
-1110
 00110
   110
   000
```

NEGATIVE NUMBERS

When we need to represent a negative binary number, we have several options although four are the most important.

- Signed Magnitude
- Ones’ complement
- Two’s complement
- Excess-K

SIGNED MAGNITUDE

The idea is to keep the MSB to indicate the sign of the number: 0 positive, 1 negative.

Decimal	Binary	Positive Binary	Negative Binary
5	101	**0**101	**1**101

As can be seen, we need a bit to indicate the sign, so that what in normal representation values would be 0 to 15 in this case, to use the sign, becomes of –7 to +7 (1111 – 0111)

This system is simple to understand but complex to use when performing mathematical operations.

There are two ways to define the 0_{10} : 0000 and 1000

ONES' COMPLEMENT

This option also uses the first bit as sign indicator, but in this case the negative number is achieved complemented positive number (changing ones' by zeros and vice versa).

In this option is required to give the number of bits to encode, in such a way that if in the previous example we use 8 bits to encode:

Decimal	Binary	Positive Binary	Negative Binary
5	101	00000101	11111010

There are two ways to define the 0_{10} : 0000 and 1111

TWO’S COMPLEMENT

Although ones' complement simplifies the mathematical operations, they do much more with the use of two's complement. That is why it is the most used method.

Two's complement consists in to apply a ones' complement and then, to add 1. For instance, two's complement of 5 encoded with 8 bits is:

$$\boxed{5_{(10)} \rightarrow 101_{(2)} \rightarrow \text{(encoded in 8 bits) } 00000101_{(2)} \rightarrow \text{(1's complement) } 1111010_{(2)} \rightarrow \text{(+1) } 11111011_{(2)}}$$

Decimal	Binary	Positive Binary	Negative Binary
5	101	**0**0000101	**1**1111011

TWO'S COMPLEMENT

What decimal number represents a number in two's complement?. Easy. We have to preform the same process:

$11111011_{(2)} \rightarrow$ (1's complement) $\rightarrow 00000100_{(2)} \rightarrow$ (+1) $00000101_{(2)} \rightarrow 5_{(10)}$

The great advantage of the two's complement method is that it allows subtraction as if they were adds. This is because subtract two binary numbers is the same as adding to the minuend the complement of the subtrahend..

$101101_{(2)}$ ($45_{(10)}$) – $010101_{(2)}$ ($21_{(10)}$) // $010101_{(2)}$ (1's complement) $\rightarrow 101010_{(2)} \rightarrow$ (+1) 101011

101101 + 101011 = 1011000 (the last carry over is rejected)$\rightarrow 11000_{(2)} = 24_{(10)}$

EXCESS-K

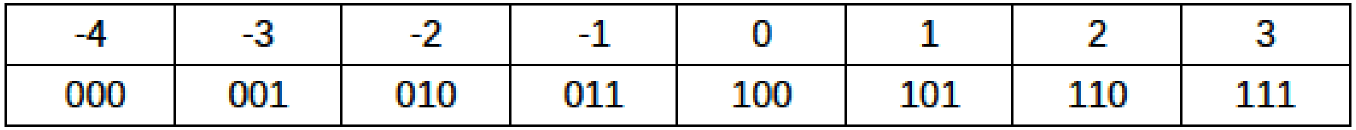
If we have a number in Excess-K and we know its decimal value, we need to subtract the value of the excess to the decimal value. For instance, if n = 8 and is K = 2^{n-1} = 128.

$11001100 \rightarrow 204\ ;\ 204 - 128 = 76$ or $00111100 \rightarrow 60\ ;\ 60 - 128 = -68$

EXCESS-K

Depending on the number of bits available, mid-range is dedicated for negative numbers and the other half (minus 1) to the positives (the zero value is in the middle). The new range will be [-K,K-1], where we can calculate by K = 2^{n-1}. Once we have the permissible range, the smallest number is who has all its bits to 0.

Example: We have 3 bits for representing the number so we can represent 2^3 numbers, the range [0,7]. In this case, K will be $2^{3-1} = 2^2 = 4$, so the range with negative numbers will be [-4,3]. The smallest number -4 will be 000 and the biggest 3 will be 111. The complete board will be:



-4	-3	-2	-1	0	1	2	3
000	001	010	011	100	101	110	111