1. Sistemas de Numeração

1.1 Introdução – Os Números

Acredita-se que a necessidade de criação de números veio com a necessidade de contar. Seja o número de animais, alimentos, ou coisas do tipo. Como a evolução nos legou algumas características, como os cinco dedos em cada mão e cinco dedos em cada pé, seria muito natural que os primeiros sistemas de numeração fizessem uso das bases 10 (decimal) e 20 (vigesimal).

Em eletrônica e Computação, as bases mais utilizadas para sistemas de numeração são:

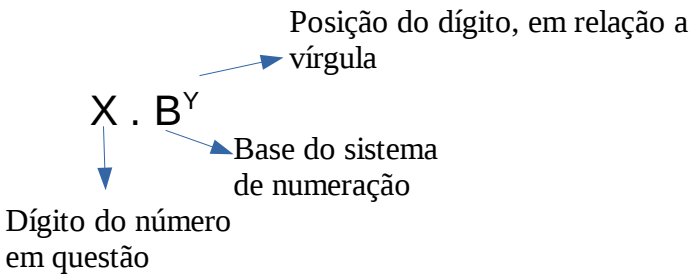
- Decimal (Base 10)
- Binária (Base 2)
- Octal (Base 8)
- Hexadecimal (Base 16)

1.2 Sistema de Numeração Decimal

O sistema de numeração normalmente utilizado, o sistema decimal, apresenta dez dígitos (algarismos), são eles: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. No sistema decimal, 10 é à base do sistema.
Obs.: para um sistema de base N, os dígitos vão de 0 à N-1.

Ex.: $328451_{10} = 3 \times 10^5 + 2 \times 10^4 + 8 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 1 \times 10^0$
$= 300000 + 20000 + 8000 + 400 + 50 + 1$
$= 328451$ ➔ Grandeza

Descrição de formação do número:



1.3 Sistema de Numeração Binário

Este sistema de numeração, como o próprio nome sugere, apresenta base 2. Os números 0 e 1 são os dígitos deste sistema.

Para representarmos à quantidade **zero**, utilizamos o algarismo (0), para representarmos a quantidade **um** utilizamos o algarismo (1). E para representarmos a quantidade **dois**, se nós não possuímos o algarismo (2) nesse sistema ? Basta lembrar-se de como é obtido o número **dez** no sistema de numeração decimal, onde os dígitos vão de 0 a 9.

Representamos a quantidade de uma dezena utilizando a algarismo 1 (um) seguido do algarismo 0 (zero). Neste caso, o algarismo 1 (um) significa que temos um grupo de uma dezena e o algarismo 0 (zero) nenhuma unidade, o que significa dez.

No sistema binário agimos da mesma forma, para representarmos a quantidade dois, utilizamos o algarismo (1) seguido do algarismo (0). Sendo assim, a numeração em binário vai tornar-se:

Decimal	Binário
0	0
1	1
2	10
3	11
4	100
5	101
.	.
.	.
.	.

O sistema binário é de grande importância, pois apresenta correspondência direta com os estados de um sistema digital. Por exemplo: para o dígito 0 pode-se atribuir o valor de tensão 0 V (GND, COM) e para o dígito 1 pode-se atribuir o valor de tensão de + 5 V.

Ex.: $1001101_2 = 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
$= 64 + 0 + 0 + 8 + 4 + 0 + 1$
$= 77_{10}$

1.3.1 Conversão de um número no sistema binário para o equivalente no sistema decimal.

Regra geral: multiplica-se cada dígito pelo valor da base elevada a uma dada potência, definida pela posição do dígito, e finalmente realiza-se a soma.

Ex.: $11001101_2 = 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
$= 128 + 64 + 0 + 0 + 8 + 4 + 0 + 1$
$= 205_{10}$

1.4 Sistema Octal de Numeração

A base de um sistema numérico é igual o número de dígitos que ela usa. Portanto, o sistema octal, que apresenta base 8, tem 8 dígitos a saber: 0, 1, 2, 3, 4, 5, 6, 7 (base N = 8 $\rightarrow$ dígitos 0 $\rightarrow$ N-1 = 7).

Sua utilidade nos sistemas digitais vem do fato de que, associando-se os algarismos de um número binário (bits) em grupos de três, obtém-se uma correspondência direta com os dígitos do sistema octal. Observaremos nitidamente este mais adiante.

1.4.1 Conversão de Octal em Decimal

$1247,235_8 = ?_{10}$

$1 \times 8^3 + 2 \times 8^2 + 4 \times 8^1 + 7 \times 8^0 + 2 \times 8^{-1} + 3 \times 8^{-2} + 5 \times 8^{-3}$
$512 + 128 + 32 + 7 + 1/8 + 3/64 + 5/512$

$1247,235_8 = 679,1816406_{10}$

1.5 Sistema de Numeração Hexadecimal

Este sistema apresenta base igual a 16. Portanto 16 dígitos distintos. São usados os dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Como no sistema de numeração octal, o hexadecimal apresenta equivalência direta entre seus dígitos e grupos de **quatro** dígitos binários. A tabela a seguir mostra esta equivalência.

Decimal	Binário	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

1.5.1 Conversão de Hexadecimal para Decimal

A regra é a mesma da conversão de qualquer sistema de numeração para o decimal.
$AFC0,7D_{16} = ?_{10}$
$A \times 16^3 + F \times 16^2 + C \times 16^1 + 0 \times 16^0 + 7 \times 16^{-1} + D \times 16^{-2}$
$10 \times 16^3 + 15 \times 16^2 + 12 \times 16^1 + 0 \times 16^0 + 7 \times 16^{-1} + 13 \times 16^{-2}$
$44992,48828_{10}$

1.6 Conversão de decimal para qualquer outra base

1.6.1 Conversão de decimal para binário.

Ex.: Conversão do número 23_{10} para binário.

$23 = 2 \times 11 + 1$

$23 = 2 \times (2 \times 5 + 1) + 1 = 5 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

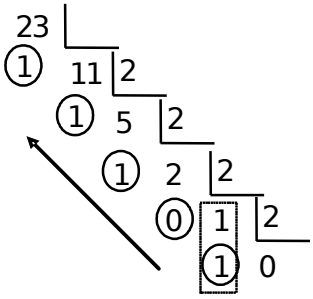
$23 = (2 \times 2 + 1) \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$23 = (1 \times 2) \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 23_{10}$

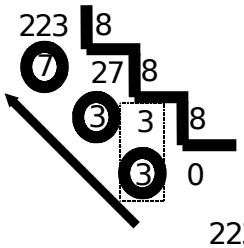
Regra prática:



$10111_2 = 23_{10}$

1.6.2 Conversão de Decimal para Octal

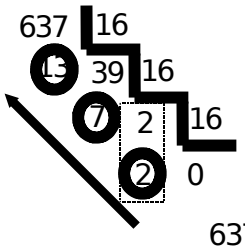
Converter o número 223 da base decimal para a octal.



$223_{10} = 337_8$

1.6.3 Conversão de Decimal para Hexadecimal

$637{,}33_{10} = ?_{16}$



$637_{10} = 27D_{16}$

1.7 Conversão de números fracionários

Regra de formação:

Decimal: $197{,}526_{10} = 1x10^2 + 9x10^1 + 7x10^0 + 5x10^{-1} + 2x10^{-2} + 6x10^{-3}$

Binário: $101101{,}101 = 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3}$

1.7.1.1 Conversão de binário, octal ou hexadecimal para decimal

$1101{,}111_2 = 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2} + 1x2^{-3}$
$= 8 + 4 + 0 + 1 + 0{,}5 + 0{,}25 + 0{,}125$
$= 13 + 0{,}875$
$= 13{,}875_{10}$

1.7.1.2 Conversão de decimal para binário, octal ou hexadecimal

$35{,}625_{10} = ?_2$

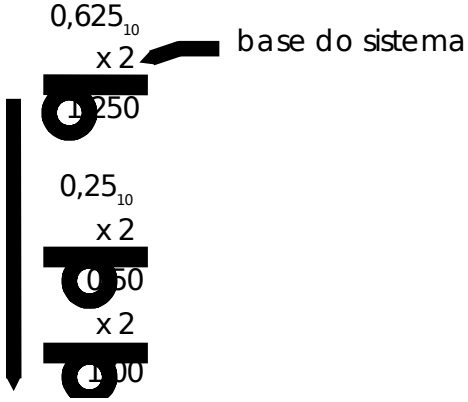
$35{,}625_{10} = \underbrace{35_{10}}_{\text{parte inteira}} + \underbrace{0{,}625_{10}}_{\text{parte fracionária}}$

A conversão da parte inteira segue o procedimento já descrito:

$35_{10} = 100011_2$

A conversão da parte fracionária segue a seguinte regra prática:

- Multiplica-se a parte fracionária pelo valor da base.
- O número resultante a esquerda da vírgula é o dígito (0 ou 1) procurado.
- Se o dígito à esquerda for 0 (zero) continuar a multiplicação pela base.
- Se o dígito à esquerda for 1 este é retirado e prossegue-se a multiplicação.
- O processo continua até obter-se 0 (zero) como resultado ou atingir-se a resolução estabelecida, no caso de dízima.
- A leitura dos dígitos, ao contrário do caso da parte inteira, é feita de cima para baixo.

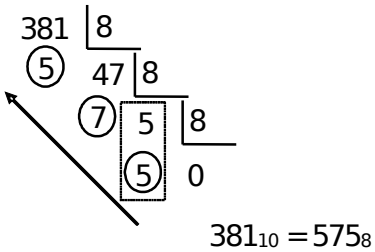


$0{,}625_{10} = 0{,}101_2$

$35{,}625_{10} = 100011{,}101_2$

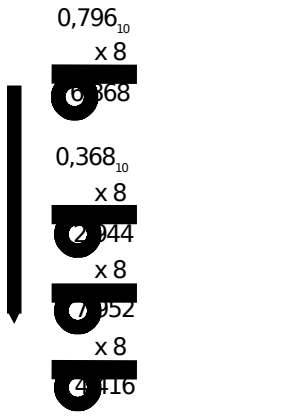
Converter o número fracionário 381,796 da base decimal para octal (4 casas decimais após a vírgula).
$381{,}796_{10} = 381_{10} + 0{,}796_{10}$

Parte inteira:



$381_{10} = 575_8$

Parte fracionária:

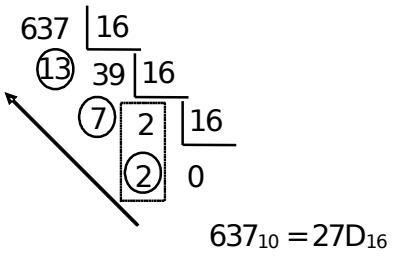


$0{,}796_{10} \approx 0{,}6274_8$ (aproximado)

Converter o número fracionário 637,33 da base decimal para hexadecimal (4 casas decimais após a vírgula).

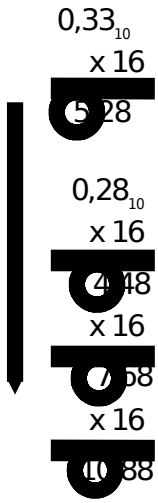
$637{,}33_{10} = ?_{16}$
$637{,}33_{10} = 637_{10} + 0{,}33_{10}$

Parte inteira



$637_{10} = 27D_{16}$

Parte Fracionária



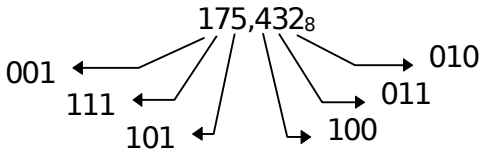
$0{,}33_{10} \approx 0{,}547A_{16}$ (aproximado)

1.8 Conversão de Octal em Binário

Para converter um número expresso em uma determinada base é normal convertermos o primeiro para um número na base 10 e, em seguida, fazer a conversão para a base desejada. Entretanto, como já foi dito, no caso do octal para o binário (e vice-versa) podemos fazer a conversão diretamente, sem passar pelo sistema decimal, já que, 8 é **terceira** potência de 2 e, portanto, são múltiplos e tem correspondência direta um com o outro.

Regra: Cada dígito octal, a partir da vírgula, é representado pelo equivalente a **três** dígitos binários. A tabela de equivalência é mostrada a seguir.

Octal	*Binário*
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111



$175{,}4328_{10} = 001\ 111\ 101{,}100\ 011\ 010_2$

1.9 Conversão de Binário em Octal

Agrega-se os dígitos binários, a partir da vírgula, em grupos de **três** e converte-se para o equivalente em octal. Caso os dígitos extremos, da direita ou esquerda, não formarem um grupo completo de **três**, adiciona-se zeros até que isto ocorra.
Converter os seguintes números de binário para octal.

$101110{,}011101_2$ = 101 110 , 011 101 $_2$
5 6 , 3 5 $_8$

$1011{,}11101_2$ = 001 011 , 111 010 $_2$
1 3 , 7 2 $_8$

Converter o número 677_{10} para binário.
1ª alternativa: dividir 677_{10} sucessivamente por 2. Solução bastante extensa.
2ª alternativa: converter 677_{10} para octal e, em seguida, converter para binário. Solução menos trabalhosa).

$677_{10} = 1245_8 = 1010100101_2$

1.10 Conversão de Hexadecimal em Binário

Da mesma forma que no sistema octal, não é necessário converter o número para o sistema decimal e depois para binário. Basta representar cada dígito hexadecimal, a partir da vírgula, em grupos de **quatro** dígitos binários equivalentes. A base 16 é a **quarta** potência da base 2. A tabela de equivalência é a que foi apresentada acima.

$FACA{,}CACA_{16} = ?_2$

F A C A , C A C A $_{16}$
1111 1010 1100 1010 , 1100 1010 1100 1010 $_2$

$FACA{,}CACA_{16} = 1111101011001010{,}1100101011001010_2$

1.11 Conversão de Binário para Hexadecimal

Como no caso da conversão de binário para octal, agrega-se os dígitos binários, a partir da vírgula, em grupos de **quatro** e converte-se para o equivalente em hexadecimal. Caso os dígitos extremos, da direita ou esquerda, não formarem um grupo completo de **quatro**, adiciona-se zeros até que isto ocorra.

$100101010{,}00111_2 = ?_{16}$
$0001\ 0010\ 1010 , 0011\ 1000_2$
1 2 A , 3 8 $_{16}$

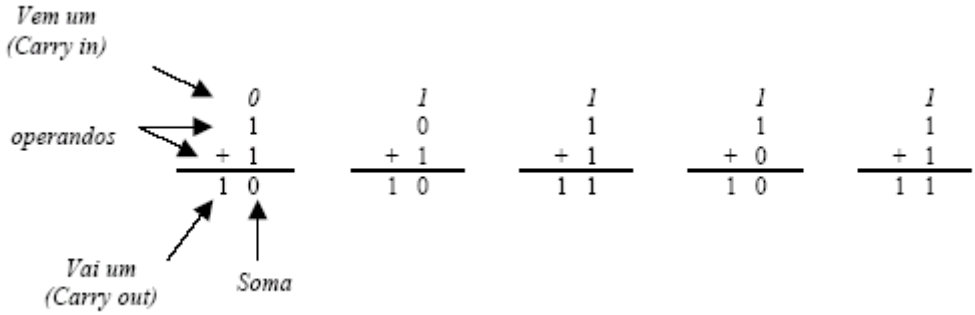
$100101010{,}00111_2 = 12A{,}38_{16}$

1.12 Aritmética Binária

1.12.1 Adição

A soma binária é realizada da mesma maneira que a soma decimal. Os números binários são somados da direita para a esquerda, gerando uma soma e um carry (vai-um) em cada posição de bit. O seguinte exemplo ilustra isto:

Carry	*1 1 1 1 1 0*	
Número A	**1 1 1 0 1**	29_{10}
Número B	+ **1 0 1 1 1**	+ 23_{10}
Soma = A+B	**1 1 0 1 0 0**	52_{10}



Regras:
$0 + 0 = 0_2$
$0 + 1 = 1_2$
$1 + 0 = 1_2$
$1 + 1 = 10_2$
$1 + 1 + 1 = 11_2$

1.12.2 Subtração

A subtração (A-B) entre dois números A e B, é calculada como a soma entre o número A e o negativo do número B (-B). Para tanto, deve-se calcular o negativo do segundo número.

1.12.2.1 Representação de Números Negativos:

A representação binária de números, estudada anteriormente, referia-se a números positivos. Para representar números negativos serão utilizadas 3 representações; (1) sinalmagnitude, (2) complemento de um e (3) complemento de dois.

- **Sinal-magnitude**, neste caso o bit mais à esquerda é utilizado para o sinal (0 quando positivo e 1 quando negativo). Os bits restantes contêm o valor (magnitude) absoluto do valor. O número negativo é formado simplesmente trocando o bit de sinal do número positivo de 0 para 1. Por exemplo, os números $+9_{10}$ e -9_{10} em um formato de 8 bits serão:

  $+9_{10} = 00001001_2$
  $-9_{10} = 10001001_2$

Sendo o formato de 8 bits, é possível representar 28=256 números válidos. No entanto, existem apenas 255 números diferentes pois +0 (00000000_2) e –0 (10000000_2) representam o mesmo número. Assim, os números se estendem no intervalo de –127 até +127.

- **Complemento de um**, o complemento de um de um número binário é obtido trocando todos os zeros por uns e os uns por zeros. utilizado para o sinal (0 quando positivo e 1 quando negativo). Por exemplo, os números $+9_{10}$ e -9_{10} em um formato de 8 bits serão:

  $+9_{10} = 00001001_2$
  $-9_{10} = 11110110_2$

O bit mais á esquerda do número é 1 quando o número é negativo, e 0 quando o número é positivo. Novamente, em um formato de 8 bits existem +0 (00000000_2) e –0 (11111111_2) representam o mesmo número e os números se estendem no intervalo de –127 até +127.

- **Complemento de dois,** o complemento de dois de um número binário é obtido calculando primeiro o complemento de 1 do número e depois somando 1. Por exemplo, para os números $+9_{10}$ e -9_{10} em um formato de 8 bits, soma-se 1 ao número obtido no exemplo anterior (11110110_2) :

  $+9_{10} = 00001001_2$

$-9_{10} = 11110111_2$

O bit mais á esquerda do número também é 1 quando o número é negativo, e 0 quando o número é positivo. No formato de 8 bits, é possível representar 2^8=256 números válidos, pois +0 (00000000_2) e –0 (00000000_2) são representados pela mesma seqüência binária. Os números, neste caso, se estendem no intervalo de –128 até 127. Esta é a representação mais freqüentemente utilizada.

A tabela 1 mostra as três representações de números em três bits.

Decimal	Sem Sinal	Sinal Magnitude	Complemento de 1	Complemento de 2
+7	111	-	-	-
+6	110	-	-	-
+5	101	-	-	-
+4	100	-	-	-
+3	011	011	011	011
+2	010	010	010	010
+1	001	001	001	001
+0	000	000	000	000
-0	-	100	111	000
-1	-	101	110	111
-2	-	110	101	110
-3	-	111	100	101
-4	-	-	-	100

Tabela 1: Representação de números de três bits.

Para realizar a subtração entre dois números, é necessário calcular o complemento de dois do subtraindo e somar com o minuendo. Isto resulta em economia de hardware.

Exemplos:

$5_{10} - 2_{10}$

```
 +5        0101
 -2   +    1110
 ---      -----
 +3       10011   = 0011
            ^
            |
         Ignora-se
```

$-5_{10} - 2_{10}$

```
 -5        1011
 -2   +    1110
 ---      -----
 -7       11001   = 1001
            ^
            |
         Ignora-se
```

$5_{10} - (-2_{10})$

```
 +5        0101
-(-2) +    0010
 ---      -----
 +3       0111    = 0111
```

$-5_{10} - (-2_{10})$

```
 -5        1011
-(-2) +    0010
 ---      -----
 -3       1101    = 1101
```

1.12.3 Adição e Subtração no Sistema de Numeração Octal e Hexadecimal

A forma mais rápida e prática de efetuar uma operação aritmética em um número octal ou hexadecimal é transformá-lo em binário, efetuar a operação e depois reconvertê-lo para octal ou hexadecimal.
Exemplos:

$$\begin{array}{r} 147_8 \\ +654_8 \\ \hline 1023_8 \end{array} \qquad \begin{array}{r} 001100\,111_2 \\ +110\,101\,100_2 \\ \hline \underbrace{1}\,\underbrace{000}\,\underbrace{010}\,\underbrace{011}_2 \\ 1\quad 0\quad 2\quad 3_8 \end{array}$$

Exemplos: Transformar os números octais para binário e verificar se o resultado da operação está correto:

147	100	6234
- 121	+37	- 2351
26	41	3663

Exemplos: Transformar os números hexadecimais para binário e verificar se o resultado da operação está correto:

F0FC	900	F731	BEBE
+A73	+CA1	- 11	+62DEB
FB6F	15A1	F720	6ECA9

1.12.4 – Over Flow

Over Flow é a mudança no sinal do resultado devido a realização de operações com números que levam ao estouro da capacidade do registrador (seqüência de bits). Esta situação ocorre quando realiza-se operações equivalentes de soma de dois números positivos ou de dois números negativos.
Exemplos: Utilizando um registrador de 4 bits, considerando representação em complemento de dois

a) 3+2

```
  0011
+ 0010
  0101  (5 resultado correto)
```

b) 5+4

```
  0101
+ 0100
  1001 (resultado errado. Número um no bit mais significativo indica número negativo, portanto pela representação de complemento de dois o resultado obtido foi –7.)
```

c) –3 –2

```
  1101
+ 1110
  1011 (-5, resultado correto)
```

d) –5 –4

```
  1011
+ 1100
  0111 (resultado errado. Número zero no bit mais significativo indica número positivo portanto o resultado obtido foi +7)
```