Capítulo 2 - Sistemas de Numeração

Objetivos

- Fornecer o conceito de número e de base de sistema de numeração;
- Mostrar os sistemas de numeração decimal, binário, octal e hexadecimal;
- Mostrar técnicas para conversão de números em bases diferentes;
- Mostrar as operações aritméticas básicas em outras bases.

Notação posicional e o sistema decimal

- Sistema decimal

- Base: 10
- Algarismos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- Notação posicional

O número da base não pode ser representado por um único algarismo. Forma-se mediante uma combinação de outros algarismos disponíveis nesta base.

A regra básica de formação permite escrever qualquer valor utilizando-se dos algarismos e de suas posições relativas às potências da base.

Exemplos:

$$1986 = 1x10^3 + 9x10^2 + 8x10^1 + 6x10^0$$

ou, em outra notação:

$$1986 = \sum_{i=0}^{3} a_i x10^i \qquad \text{para } a_3 = 1,\ \ a_2 = 9,\ \ a_1 = 8,\ \ a_0 = 6.$$

para valores com parte fracionária:

$$0{,}1986 = 1x10^{-1} + 9x10^{-2} + 8x10^{-3} + 6x10^{-4}$$

ou, também, em outra notação:

$$0{,}1986 = \sum_{j=0}^{-4} a_j x10^{-j} \qquad \text{para } a_1 = 1,\ \ a_2 = 9,\ \ a_3 = 8,\ \ a_4 = 6.$$

Generalizando a notação para qualquer número:

$$N = \sum_{i=0}^{p} a_i xb^i + \sum_{j=1}^{q} a_j xb^{-j}$$

ou, de forma mais simplificada:

$$N = \sum_{k=q}^{p} a_k x b^k \quad (q \in Z \mid q \leq 0)$$

onde

N - uma quantidade qualquer;
p - maior potência positiva, na qual existe um algarismo significativo (diferente de zero);
q - menor potência negativa, na qual existe um algarismo significativo (diferente de zero);
a - um algarismo qualquer [0:(b-1)] da base (b);
b - base de um sistema de numeração.

Com tal generalização, é possível representar qualquer quantidade, em qualquer base, desde que conhecidos os algarismos e suas posições relativas à base.

Sistema binário

- Base: 2
- Algarismos: 0, 1

A partir da regra básica de formação pode-se escrever qualquer valor, usando apenas os elementos desta base. Qualquer número na base 10 pode ser representado por um equivalente na base 2.

Exemplos:

$$13_{(10)} = 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 1101_{(2)}$$

$$0{,}625_{(10)} = 1x2^{-1} + 0x2^{-2} + 1x2^{-3} = 0{,}101_{(2)}$$

Observação:

Os números abaixo representam valores diferentes:

$10_{(10)} = 10$ e $10_{(2)} = 2$

Exercício - Escrever em notação posicional e o valor decimal:

a) $1110101_{(2)}$
b) $0{,}01101_{(2)}$
c) $110{,}101_{(2)}$
d) $010{,}010_{(2)}$
e) $111{,}001_{(2)}$

Sistema octal

- Base: 8
- Algarismos: 0, 1, 2, 3, 4, 5, 6, 7

A partir da regra básica de formação pode-se escrever qualquer valor, usando apenas os elementos desta base.

Exemplos:

$$13_{(10)} = 1x8^{1} + 5x8^{0} = 15_{(8)}$$

$$0{,}625_{(10)} = 5x8^{-1} = 0{,}5_{(8)}$$

Sistema hexadecimal

- Base: 16
- Algarismos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
  A(=10),B(=11),C(=12),D(=13),E(=14),F(=15)

A partir da regra básica de formação pode-se escrever qualquer valor, usando apenas os elementos desta base.

Exemplos:

$$1986 = 7x16^{2} + Cx16^{1} + 2x16^{0} = 7C2_{(16)}$$

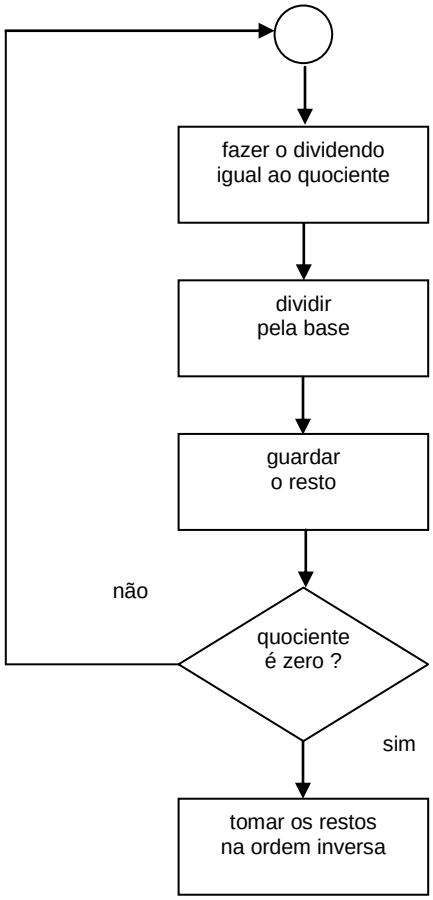
$$13{,}625 = D,A_{(16)}$$

Conversão entre bases

- Conversão de base 10 para uma base (b)

  - Parte inteira

    Dividir o número decimal pela base (b), tomando o resto e o quociente separadamente. Continuar a dividir o quociente obtido pela base, guardando os novos resto e quociente, até que o quociente seja igual a zero (0). Tomar os restos na ordem inversa em que forem sendo calculados.



Exemplo:

	Quociente	Resto
13 ÷ 2 =	6	(+ 1)
06 ÷ 2 =	3	(+ 0)
03 ÷ 2 =	1	(+ 1)
01 ÷ 2 =	0	(+ 1)

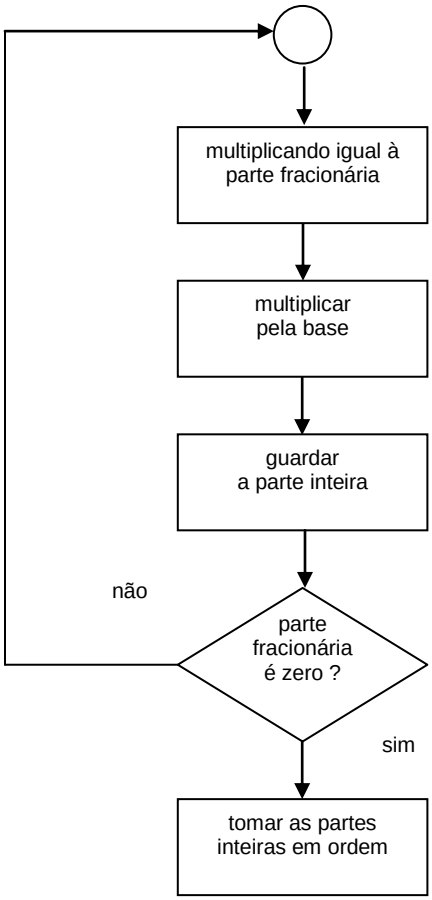
$13_{(10)} = 1101_{(2)}$

```
13 | 2
 1   6 | 2
     0   3 | 2
         1   1 | 2
             1   0
```

$13_{(10)} = 1101_{(2)}$

- Parte fracionária

Multiplicar a parte fracionária do número decimal pela base (b), tomando apenas a parte inteira produzida. Continuar o processo até obter apenas zeros na parte fracionária, ou obter um número cuja representação esteja dentro de uma precisão aceitável. Esta última condição atende ao fato de que alguns números terem como representação uma dízima. Tomar as partes inteiras na ordem em que forem sendo calculadas.



Exemplo:

0,3520 x 2 = 0 + 0,7040
0,7040 x 2 = 1 + 0,4080
0,4080 x 2 = 0 + 0,8160
0,8160 x 2 = 1 + 0,6320
0,6320 x 2 = 1 + 0,2640

Supondo uma aproximação razoável de 5 dígitos:

$$0{,}3520_{(10)} = 0{,}01011_{(2)}$$

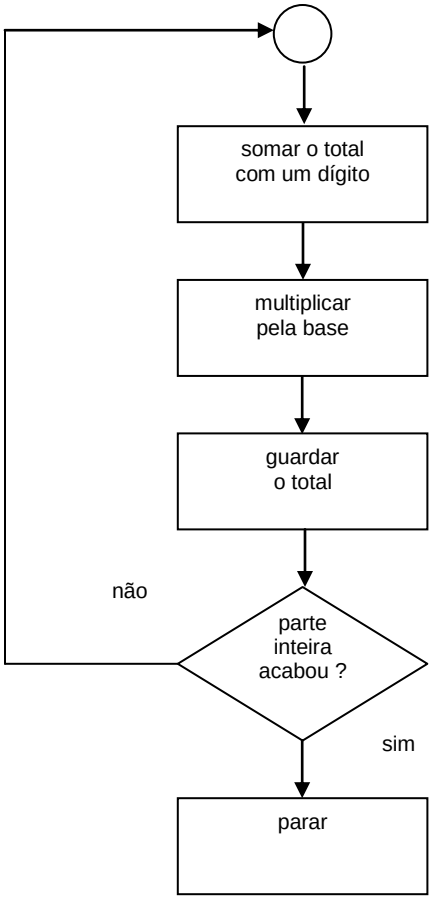
Exercício - Converter para os binários equivalentes:

a) 1234
b) 0,35
c) 12,7
d) 25,25
e) 103,412

- Conversão de uma base (b) para a base 10:

- Parte inteira

Multiplicar o dígito mais a esquerda do número na base (b) pela própria base. Somar o próximo dígito ao produto, e continuar o processo até que o último dígito à direita seja somado. O número decimal será o resultado destas operações.



Exemplo:

$1101_{(2)} = 13_{(10)}$

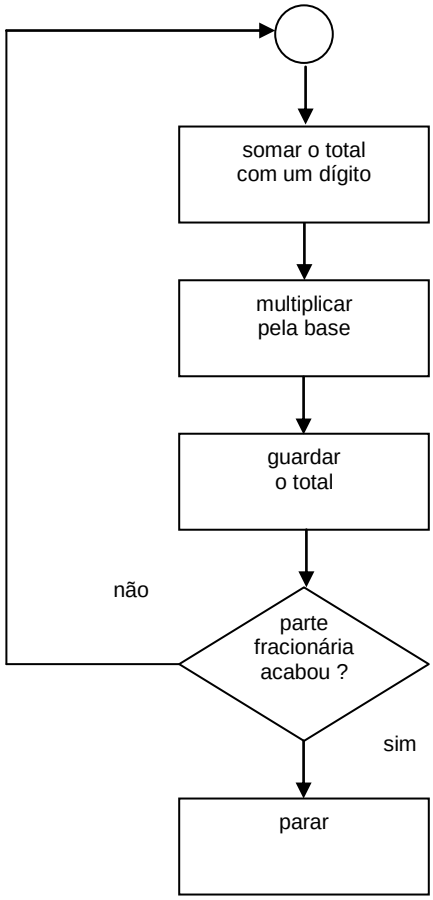
(0 + 1)	x 2 =	2
(2 + 1)	x 2 =	6
(6 + 0)	x 2 =	12
12 + 1	=	13

ou

$$1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 13_{(10)}$$

- Parte fracionária

Multiplicar o dígito mais a direita do número na base (b) pelo inverso da própria base (1/b). Somar o próximo dígito, à esquerda, ao produto, e continuar o processo até que o último dígito à esquerda seja somado, e a soma multiplicada por (1/b). O número decimal será o resultado destas operações.



Exemplo:
Supondo uma precisão de 10^{-4} :

$$0{,}1101_{(2)} = 0{,}8125_{(10)}$$

(0 + 1)	x 1/2 =	0,5
(0,5 + 1)	x 1/2 =	0,25
(0.25 + 0)	x 1/2 =	0,625
(0,625 + 1	=	0,8125

ou

$$1x2^{-1} + 1x2^{-2} + 0x2^{-3} + 1x2^{-4} = 0{,}8125_{(10)}$$

Exercício - Converter para os decimais equivalentes:

a) $1010110_{(2)}$
b) $0{,}01101_{(2)}$
c) $110{,}111_{(2)}$
d) $110{,}01011_{(2)}$
e) $11{,}0011_{(2)}$

- Conversão de uma base (b) para uma base (b^n)

- Parte inteira

Agrupar os dígitos do número na base (b) em grupos de (n) dígitos, a partir da direita, completando com zeros, à esquerda, os dígitos que faltarem. Converter cada um dos grupos para o algarismo correspondente na outra base.

Exemplo:

Converter o número $11111000001_{(2)}$ para a base (4):

01	11	11	00	00	$01_{(2)}$
1	3	3	0	0	$1_{(4)}$

Portanto,

$$11111000001_{(2)} = 133001_{(4)}$$

- Parte fracionária

Agrupar os dígitos do número na base (b) em grupos de (n) dígitos, a partir da vírgula, completando com zeros, à direita, os dígitos que faltarem. Converter cada um dos grupos para o algarismo correspondente na outra base.

Exemplo:

Converter o número $0{,}0011001011_{(2)}$ para a base (8):

001	100	101	$100_{(2)}$
1	4	5	$4_{(8)}$

Portanto $0{,}0011001011_{(2)} = 0{,}1454_{(8)}$

- Parte inteira e fracionária

Agrupar os dígitos do número na base (b) em grupos de (n) dígitos, completando com zeros os dígitos que faltarem, tanto á esquerda quanto á direita. Converter cada um dos grupos para o algarismo correspondente na outra base.

Exemplo:

Converter o número $11111000001{,}0011001011_{(2)}$ para a base (16):

0111	1100	0001	, 0011	0010	$1100_{(2)}$
7	C(=12)	1	, 3	2	$C_{(16)}$

Portanto $11111000001{,}001100101100_{(2)} = 7C1{,}32C_{(16)}$

Exercício - Fazer as conversões de base:

a) $102130122_{(4)}$ para a base 8
b) $111010111_{(2)}$ para a base 16
c) $122122320_{(4)}$ para a base 16
d) $AB50{,}F97C_{(16)}$ para a base 8

O quadro abaixo resume as primeiras representações equivalentes em cada base:

base 10/16		2^3	2^2	2^1	2^0		base
0		0	0	0	0		
1		0	0	0	1		2
2		0	0	1	0		
3		0	0	1	1		4
4		0	1	0	0		
5		0	1	0	1		
6		0	1	1	0		
7		0	1	1	1		8
8		1	0	0	0		
9		1	0	0	1		
10=A		1	0	1	0		
11=B		1	0	1	1		
12=C		1	1	0	0		
13=D		1	1	0	1		
14=E		1	1	1	0		
15=F		1	1	1	1		16

Operações aritméticas em base 2

- Soma e multiplicação

0 + 0 = 0	0 x 0 = 0
0 + 1 = 1	0 x 1 = 0
1 + 0 = 1	1 x 0 = 0
1 + 1 = 10	1 x 1 = 1

Exemplos:

"vai-um" → 111

```
   111010          1010
+  101100     x     101
_________     ________
  1100110         1010
                 0000 -
               + 1010 - -
               ________
                110010
```

Exercício - Efetuar as operações binárias:

a) 1101010 + 101101
b) 0,11011 + 0,0101
c) 11,1101 x 101
d) 101,11 x 10,1 + 10,101
e) 101,11 x (10,1 + 10,101)

- Subtração

A subtração binária é feita da mesma forma que a decimal, lembrando que, se o minuendo for menor que o subtraendo, pode-se tomar uma unidade emprestada à casa seguinte, se houver.

Exemplo:

```
                1011
              - 0110
              ______
                0101

                          -1+1x2    "vem-um"
 101| 1     10| 1 | 1    1| 0 | 11   0|10 | 11    0|10 | 11
- 011| 0   - 01| 1 | 0  - 0| 1 | 10 - 0|  1 | 10  - 0|  1 | 10
_______  __________  _________ _________ _________
   | 1      | 0 | 1      | 01     |  1 | 01     0|  1 | 01
```

Exercício - Efetuar as subtrações binárias:

a) 1001100 - 1111
b) 0,11001 - 0,01
c) 10,1011 - 1,11

- Divisão

A divisão pode ser feita com subtrações sucessivas.

Exemplo:

```
 1011010        | 10____
-10_____        101101
 0011
-  10___
   010
-    10_
     0010
-      10
       00
```

Exercício - Efetuar as divisões binárias:

a) 1101110: 100
b) 0,11011: 0,11
c) 11,1011: 11

Representação de um número em ponto fixo

A representação de qualquer símbolo em um computador digital é feita através de uma seqüência de dígitos (0 ou 1). Esta representação é ambígua, servindo tanto para números, como para letras, ou outros sinais. Além disso, o tamanho desta seqüência é bastante limitado.

Tomando a representação de números, em particular, nota-se que não há uma forma padrão que indique o sinal, pois outro símbolo não pode ser empregado. Por isso, desenvolveram-se métodos para o tratamento de sinal em números binários. Cada um deles toma por princípio que o número tenha uma representação com ponto fixo, ou seja, o ponto (ou vírgula) decimal tem uma posição bem definida, e as operações têm que levá-la em consideração. No caso desta posição ser a mais à direita, os números serão quantidades inteiras, positivas ou negativas.

- Representação em sinal e amplitude

Nessa representação, o dígito mais à esquerda indica o sinal do número, segundo a convenção abaixo:

0 - positivo
1 - negativo

Exemplo:

Representar os binários equivalentes aos valores (+5) e (-5) com 6 dígitos e tamanho fixo.

	Sinal	Amplitude	
0 0 0 1 0 1 =	**0**	0 0 1 0 1	= (+ 5)
1 0 0 1 0 1 =	**1**	0 0 1 0 1	= (- 5)

Observação: O zero tem duas representações: 0 00000 e 1 00000.

- Noção de complemento

Considera-se complemento de um número em relação a outro, a diferença entre os dois. É útil para aplicação em computadores.

Exemplo 1:

O complemento de dez de 2 é: 10 - 2 = 8. E a noção de complemento é aplicável à subtração:

9	9	9	17
- 2	+ (10 - 2)	+ 8	- 10
	(x -10)	(x -10)	+ 7 (=x)

Considerando-se o que falta para completar 10, (17-10), o resultado é correto.

Exemplo 2:

O complemento de dez de 9 é: 10 - 9 = 1. Essa noção de complemento aplicada à operação:

2	2	2	3
- 9	+ (10 - 9)	+ 1	- 10
	(x - 10)	(x -10)	- 7 (=x)

Considerando-se o que falta para completar 10, (3-10), o resultado é correto.

Exemplo 3:

O complemento de dez de 9 é: 10 - 9 = 1. E aplicando, novamente, a noção de complemento:

- 2	(10 - 2)	8	9
- 9	+ (10 - 9)	+ 1	- 20
	(x - 20)	(9-20)	- 11 (=x)

Considerando-se o que falta para completar cada complemento de 10, resultado é correto.

- Representação em complemento de 2

Seja (m) o tamanho fixo da representação, e 2^m um valor não representável nesse tamanho.

Os números positivos estarão no intervalo $[0,2^{m-1} - 1]$, ou seja, todos os números entre zero e o maior valor capaz de ser representado, sem contar com uma casa para o sinal.

Os números negativos estarão no intervalo $[-2^{m-1}, -1]$. Sua representação será feita tomando o equivalente binário para o resultado da soma deste número com o valor máximo.

Exemplo:

Representar os binários equivalentes a (+5) e (-5) com 6 dígitos e tamanho fixo.

	sinal	amplitude	
0 0 0 1 0 1 =	**0**	0 0 1 0 1	= (+ 5)
1 1 1 0 1 1 =	**1**	1 1 0 1 1	= (- 5)

A representação de (- 5) é obtida por $2^6 - 5 = 64 - 5 = 59_{(10)} =$ **1** $11011_{(2)}$

Observação: O zero tem representação única: **0** 00000.

No entanto, não existe simetria: -2^{m-1} tem representação, mas 2^{m-1} não.

Regra prática:

Para converter um número binário para a seu simétrico, em complemento de 2, trocar todos os zeros por uns, e uns por zeros, acrescentando mais uma unidade ao resultado.

Exemplo:

0 0 0 1 0 1 =	**0**	0 0 1 0 1 =	(+ 5)
1 1 1 0 1 0 =	**1**	1 1 0 1 0	(trocados 0's e 1's)
1 1 1 0 1 1 =	**1**	1 1 0 1 1 =	(- 5)

- Representação em complemento de 1

Seja (m) o tamanho fixo da representação, e 2^m um valor não representável nesse tamanho.

Os números positivos estarão no intervalo $[0,2^{m-1} - 1]$, ou seja, todos os números entre zero e o maior valor capaz de ser representado, sem contar com uma casa para o sinal.

Os números negativos estarão no intervalo $[-2^{m-1} +1, -1]$. Sua representação será feita tomando o equivalente binário para o resultado da soma deste número com o valor máximo.

Exemplo:

Representar os binários equivalentes a (+5) e (-5) com 6 dígitos e tamanho fixo.

	sinal	amplitude	
0 0 0 1 0 1 =	**0**	0 0 1 0 1	= (+ 5)
1 1 1 0 1 1 =	**1**	1 1 0 1 0	= (- 5)

A representação de (- 5) é obtida por $(2^6 - 1) - 5 = 63 - 5 = 58_{(10)} =$ **1** $11010_{(2)}$

Observação: O valor zero terá duas representações: **0** 00000 e **1** 11111.

Regra prática:

Para converter um número binário para a seu simétrico, em complemento de 1, trocar os zeros por uns, e uns por zeros.

Exemplo:

0 0 0 1 0 1 = **0** 0 0 1 0 1 = (+ 5)
1 1 1 0 1 0 = **1** 1 1 0 1 0 = (- 5)

A subtração pode ser feita utilizando-se a representação em complemento de 2, somando-se o número e seu complemento, e desprezando-se o "vai-um" na última casa, se houver.

Exemplo:

	1 1 1
0 110 (+6)	0 110
- **0** 101 (+5)	+ 1 011
	(1)0 001

0 111(+5 $\rightarrow$ **1** 011 (complemento de 2 de **0** 101)

Exemplo 2:

0 101 (+5)	0 101
- **0** 110 (+6)	+ 1 010
	1 111

0 110 (+6) $\rightarrow$ **1** 010 (complemento de 2 de **0** 110)

Recorrendo à relação inversa do complemento de 2, subtraindo um do número, trocando zeros por uns, e vice-versa:

1 111
\- 1
1 110 ~ **0** 001 (+1)

portanto, **1** 111 é representação de (-1), o que também pode ser verificado assim:

1 111 = 15 $\Rightarrow$ $15 - 2^4 = (-1)$

Exemplo 3:

	1 1
1 011 (- 5)	1 011
- **0** 010 (+5)	+ 1 110
	(1)1 001

0 111(+5 $\rightarrow$ 1 011 (complemento de 2 de 0 101)

Recorrendo à relação inversa do complemento de 2:

1 001
\- 1
1 000 ~ **0** 111 (+7)

portanto, **1** 001 é representação de (-7). O que também pode ser verificado assim:

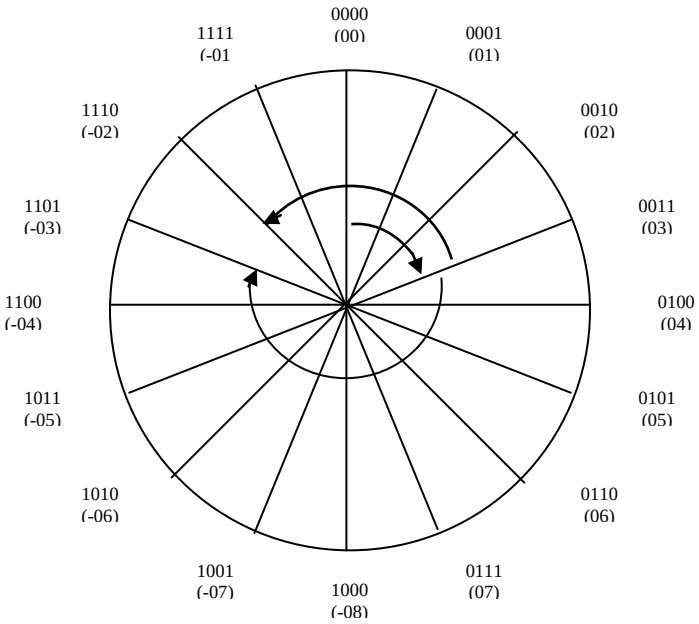
1 001 = 9 $\Rightarrow$ $9 - 2^4 = (-7)$

- Comparação entre as duas representações em complemento

Exemplo:

Para um tamanho de representação igual a 4, em complemento de 2 tem-se que:

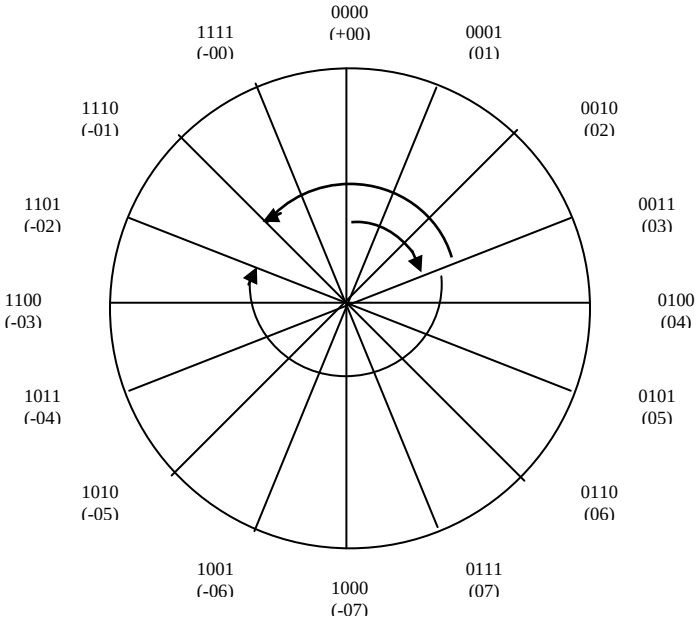
$$3 - 5 = 0011_{(2)} - 0101_{(2)} = 0011_{(2)} + 1011_{(2)} = 3\nabla + 11\nabla$$



Exemplo:

Supondo um tamanho de representação igual a 4, em complemento de 1 tem-se que:

$$3 - 5 = 0011_{(2)} - 0101_{(2)} = 0011_{(2)} + 1010_{(2)} = 3\nabla + 10\nabla$$



Corrige-se a diferença, acrescentando-se mais um deslocamento, o que também é feito no complemento de 2 adicionando-se mais uma unidade.

Exercício:

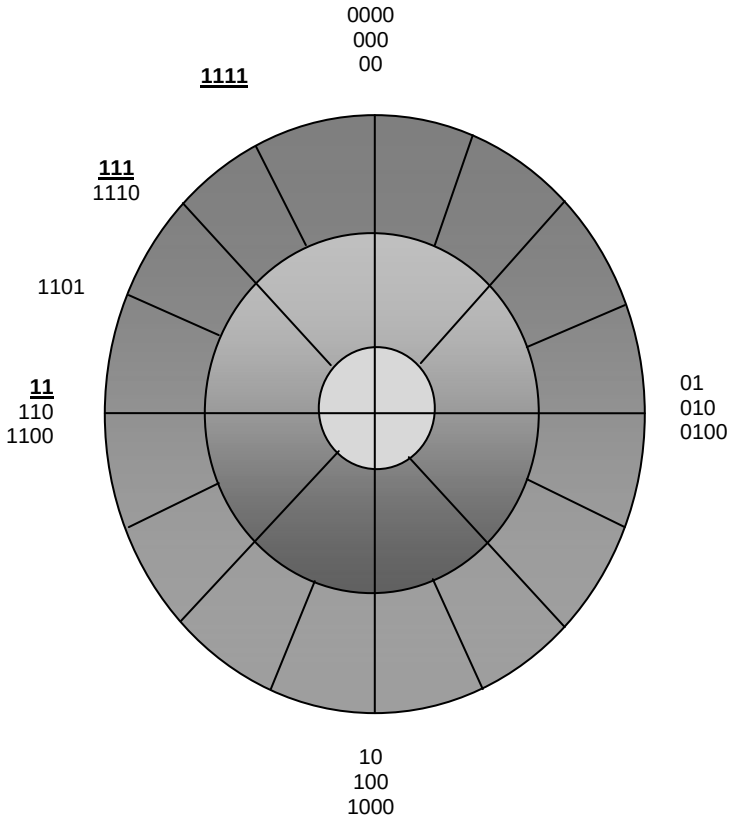
Efetuar as subtrações propostas anteriormente utilizando complemento de 2.

É importante notar que a representação de um mesmo valor negativo varia de acordo com a quantidade de bits usados.

Exemplo:

(-1) = $11_{(2)}$ com 2 bits
= $111_{(2)}$ com 3 bits
= $1111_{(2)}$ com 4 bits

O diagrama abaixo ilustra essas diferentes representações



Representação de um número em ponto flutuante

A representação em ponto flutuante permite representar valores que dificilmente caberiam em notação de ponto fixo, devido ao tamanho limitado da palavra do computador. Entretanto, ela também tem um número limitado de bits significativos.

Existem normas, como ANSI/IEEE 754/854 (1985/1987), que estabelecem o formato desta representação, entretanto, usaremos uma forma simplificada (com 1+e+m bits):

Sinal	Expoente	Mantissa
S	E	M
(1)	(e)	(m)

$$N = (-1)^S \cdot 2^p \times M \qquad (\tfrac{1}{2} \le M < 1)$$

$p = E - 2^{e-1}$ (o excesso de 2^{e-1}, e $E \geq 0$)

Exemplo 1:

Dado o número (0 1010 1100000), com e = 4 e m = 7, determinar o valor representado.

$$N = +2^{(1010-1000)} \times 0.1100000_{(2)} = +2^{10-2^{4-1}} \times 0.75 = 3.0$$

Exemplo 2:

Dado o número 1.70, com e = 4 e m = 7, representá-lo em ponto flutuante.

$$1.70 = +2^p \times M$$

o menor valor de (p) possível neste caso é 1, logo M = 0.85

que pode ser aproximado com 7 casas por $0.1100110_{(2)}$

e o expoente deve ser $p = 1 \Rightarrow 2^1 = 2^{(E-2^{4-1})} \Rightarrow E = 9$ representado por $1001_{(2)}$

assim a representação final será N = **0 1001** $1110011_{(2)}$

Exemplo 3:

Dado o número (**1 0011** 1100000), com e = 4 e m = 7, determinar o valor representado, com três casas decimais.

$$N = -2^{(0011-1000)} \times 0.1100000_{(2)} = +2^{3-2^{4-1}} \times 0.75 = -0.023$$

Exemplo 4:

Dado o número -0.25, representá-lo em ponto flutuante, em sinal e amplitude.

$$-0.25 = -2^p \times M = -0.01_{(2)} = -2^0 \times 0.01_{(2)}$$

Mas, esse valor não está de acordo com as especificações (não está "normalizado"), pois não pertence ao intervalo [1/2, 1), e o expoente é nulo. Se tomado o menor valor do intervalo,

$M = 0.1000000_{(2)} = 0.5$

o expoente deve ser $p = -1 \Rightarrow 2^{-1} = 2^{(x-2^{4-1})} \Rightarrow x = 7$ representado por $0111_{(2)}$ e

o valor "normalizado" (com o primeiro dígito mais à esquerda diferente de zero na mantissa), para e = 4 e m = 7, será:

N = **1 0111** 1000000.

O quadro abaixo resume as diferentes representações binárias de inteiros e expoentes:

base 10/16		2^3	2^2	2^1	2^0	base		-x [1]	-x [2]	-x [3]	-x [4]		2^x [4]	2^x [3]	2^x [2]	2^x [1]	(e-1) 2
0		0	0	0	0			+	0	0	0		-8	-4	-2	-	
1		0	0	0	1	2		-	1	1	1		-7	-3	-1	+	
2		0	0	1	0			→	-2	2	2		-6	-2	0	←	2
3		0	0	1	1	4			-1	3	3		-5	-1	1		
4		0	1	0	0				→	-4	4		-4	0	←		4
5		0	1	0	1					-3	5		-3	1			
6		0	1	1	0					-2	6		-2	2			
7		0	1	1	1	8				-1	7		-1	3			
8		1	0	0	0					→	-8		0	←			8
9		1	0	0	1						-7		1				
10=A		1	0	1	0						-6		2				
11=B		1	0	1	1						-5		3				
12=C		1	1	0	0						-4		4				
13=D		1	1	0	1						-3		5				
14=E		1	1	1	0						-2		6				
15=F		1	1	1	1	16					-1		7				

Os padrões ANSI/IEEE-754 e ANSI/IEEE-854

O padrão ANSI/IEEE-754 (1985) estabelece uma representação de 32 ***bits*** (1+8+23) para um número em ponto-flutuante com precisão simples (aproximadamente, de 10^{-44} até 10^{38}), e uma de 64 ***bits*** (1+11+52) para a precisão dupla (aproximadamente, de 10^{-323} até 10^{308}). Além disso, aplicam-se as seguintes regras:

- se o expoente for igual a 255 (ou 2047) e a mantissa diferente de zero,
  a representação não será considerada como a de um número válido (**NaN**);
- se o expoente for igual a 255 (ou 2047), a mantissa igual a zero e o sinal igual a 1,
  a representação será considerada como infinito negativo (**-Infinitiy**);
- se o expoente for igual a 255 (ou 2047), a mantissa igual a zero e o sinal igual a 0,
  a representação será considerada como infinito positivo (**+Infinitiy**)
- se o expoente for igual a zero, a mantissa igual a zero e o sinal igual a 1,
  a representação será considerada como zero negativo (**-0**);
- se o expoente for igual a zero, a mantissa igual a zero e o sinal igual a 0,
  a representação será considerada como zero (**0**);
- se o expoente for igual a zero, a mantissa diferente de zero,
  a representação será considerada como ***não normalizada***, e se aplicará a fórmula:

$$N = (-1)^S \times 2^{(-126)} * (0.F)$$

- se o expoente estiver no intervalo (0:255), e a mantissa diferente de zero,
  a representação será considerada como ***normalizada***, e se aplicará a fórmula abaixo,
  onde (***e***) será o número de bits do expoente e a mantissa irá iniciar-se por 1, :

$$N = (-1)^S \times 2^{(E-127)} * (1.F)$$

O padrão ANSI/IEEE-854 (1987) é uma generalização que estabelece normas para as operações em ponto-flutuante, independente da base do sistema de numeração utilizada.

O padrão ANSI/IEEE-754 foi atualizado em 2008 para incluir valores representáveis até com 128 ***bits*** (1+15+112) para precisão quádrupla.

Erros em representação numérica

As representações e as aritméticas de ponto fixo são convenientes para se tratar o conjunto dos números inteiros (Z), ou seja, números sem parte fracionária, dentro de certo intervalo de valores dependente do tamanho da representação (m).

A representação e aritmética em ponto flutuante são convenientes para se tratar o conjunto dos números reais (R), ou seja, números com parte fracionária, dentro de certo intervalo de valores representáveis. Durante as operações aritméticas deve-se manter o cuidado de se "alinhar as vírgulas" ajustando-se as mantissas, segundo os valores dos expoentes, e realizar uma verificação extra do sinal dependendo da comparação de seus valores. O resultado também poderá requerer normalização após ser calculado.

A dificuldade básica na representação numérica é que os valores deverão ser necessariamente armazenados em uma quantidade finita (m) de dígitos binários. Isso significa que há uma limitação quanto ao número de representações distintas, e que cada número representável deverá ter no máximo (m) dígitos significativos. Assim, poderão ocorrer vários tipos de erros relacionados com a representação.

- Intervalo

O número de dígitos significativos limita também o intervalo de valores representáveis, tanto em ponto, fixo como em ponto flutuante, onde o expoente é o principal fator determinante.

Exemplo:

Dispondo-se de uma representação, em complemento de 2, com tamanho (m) de 16 dígitos (1 para o sinal e 15 para a amplitude) pode-se representar valores inteiros dentro do intervalo $[-2^{15}, 2^{15} - 1] = [-32768, 32767]$.

Dispondo-se de uma representação, em complemento de 2, com tamanho (m) de 4 bytes (11 dígitos decimais significativos) pode-se representar valores reais no intervalo $[-10^{38}, 10^{38}]$.

- Arredondamento e truncamento

Tanto em ponto fixo, como em ponto flutuante, podem acontecer erros devido ao tamanho finito da representação.

Exemplo:

Para o valor representado em binário: $0.87 = 01101111010_{(2)}$

considerando uma representação com 8 dígitos: $0.11011110 = 0.8671875_{(2)}$

e precisão de 3 casas decimais:

$0.87 \approx 0.867$ (truncamento)

$0.87 \approx (0.867 + 0.005) \approx 0.872$ (arredondamento)

Mas, para o valor representado em binário a: $0.86 = 0.1101110000_{(2)}$

considerando uma representação com 8 dígitos: $0.11011100_{(2)} = 0.859375$

e precisão de 3 casas decimais:

$0.86 \approx 0.859$ (truncamento)

$0.86 \approx (0.859 + 0.005) \approx 0.864$ (arredondamento)

- Precisão

A precisão está relacionada com o número de dígitos significativos. No caso de números inteiros, limita-se aos (m-1) dígitos da palavra de representação; no caso de números com parte fracionária limita-se a 1/(m-1) para representação em ponto fixo.

Exemplo:

Para o valor abaixo representado em binário: $0.87 = 0.1101111010_{(2)}$
considerando uma representação com apenas 8 dígitos: $0.11011110_{(2)} = 0.8671875$
a precisão será de $\pm 2^{-8} = 3.90625 \times 10^{-3}$.

- Transbordamento

Os erros de transbordamento ocorrem, geralmente, durante a utilização da representação numérica para fins aritméticos.

Exemplo:

Somar X e Y binários com representação em complemento de dois, com seis dígitos:

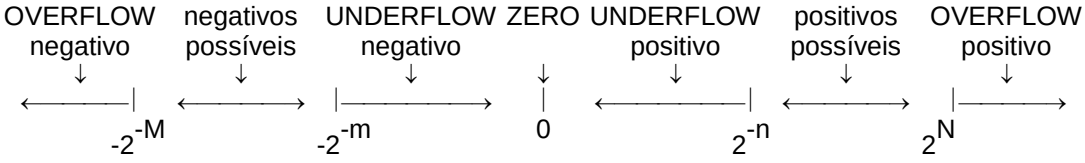
	1 1 1	
X =	**0** 11111	= +31
+ Y =	**0** 01100	= +12
	1 01011	= -21 (ERRADO !!!)

O erro verificado deve-se a uma situação de transbordamento (*overflow*) em relação ao tamanho da representação. Não há casas da amplitude suficientes para conter o resultado. Por isso, o "vai-um" na casa do sinal ao invés de servir para corrigir o sinal do resultado, passará a ser um dígito significativo (fará parte da representação da quantidade).

O mesmo tipo de erro poderá ocorrer quando se operam números binários negativos:

	1	(em complemento de dois)
X =	**1** 00001	= -31
+ Y =	**1** 10100	= -12
	0 10101	= +21 (ERRADO !!!)

Na subtração poderá ocorrer erro semelhante devido à insuficiência, ou falta (*underflow*).



- Não-associatividade

É possível na aritmética computacional que, para alguns valores (não todos):

$(a + b) + c \neq a + (b + c)$
$(a \times b) \times c \neq a \times (b \times c)$

Poderão ocorrer erros de transbordamento por excesso (*overflow*), ou falta (*underflow*).

- Acumulação

Para sequências de operações, os erros de acumulação poderão ocorrer com grande rapidez, anulando-se ou não, de forma imprevisível.

Representação de letras e símbolos

Para a representação de letras e outros símbolos utilizam-se tabelas de equivalências, onde cada letra, ou símbolo, corresponde a um código numérico.

Há vários códigos:

- BCD (Binary Coded Decimal)
- EBCDIC (Extended Binary Coded Decimal Interchange Code)
- ASCII (American Standard Code for Information Interchange)
- UNICODE

Por exemplo, em ASCII (tabela hexadecimal):

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	^@	^A	^B	^C	^D	^E	^F	^G	^H	^I	^J	^K	^L	^M	^N	^O
1	^P	^Q	^R	^S	^t	^U	^V	^W	^X	^Y	^Z	^[	^\	^]	^^	^
2		!	“	#	$	%	&	‘	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	←

Nesse padrão tem-se:

		hexadecimal					decimal				
A S C I I	=	41	53	43	49	49	= 65	83	67	73	73
a s c i i	=	61	73	63	69	69	= 97	115	99	105	105
1 9 8 6	=	31	39	38	36		= 49	57	56	54	

Observação importante:

Reparar que os símbolos que representam os algarismos têm, em código ASCII, valores diferentes de seus equivalentes binários; por exemplo, zero será representado por $30_{(16)} = 48$.

Exercícios propostos

1. Construir uma tabela:

   - de 0 a 15, nas bases 2, 4, 8, 10 e 16,
   - acrescentar a representação em complemento de 1 e 2, de cada número.

2. Fazer as conversões de base:

   a) $1324_{(5)}$ para a base 10
   b) $3573_{(8)}$ para a base 16
   c) $8957_{(11)}$ para a base 7
   d) $2201_{(3)}$ para a base 9
   e) $B8C0_{(13)}$ para a base 14

3. Verificar a divisibilidade dos números abaixo:

   a) $0111\ 1000_{(2)}$ por 4
   b) $3132_{(4)}$ por 8
   c) $160_{(8)}$ por 16
   d) $1B9A_{(16)}$ por 2

4. Efetuar as operações abaixo:

   a) $2412_{(5)} + 5627_{(8)}$
   b) $0,13_{(4)} \times 0,11_{(2)}$
   c) $1,24_{(7)} - 1,14_{(5)}$
   d) $1D4F_{(16)} \div 8131_{(9)}$

5. Efetuar as operações abaixo em binário:

   a) $100010_{(2)} + 11101_{(2)}$
   b) $101010_{(2)} - 10110_{(2)}$
   c) $1101_{(2)} \times 101_{(2)}$
   d) $101100_{(2)} \div 100_{(2)}$

6. Efetuar as operações abaixo em binário:

   a) $100,110_{(2)} + 11011_{(2)}$
   b) $1010,10_{(2)} - 10,11_{(2)}$
   c) $11,01_{(2)} \times 1,01_{(2)}$
   d) $101,100_{(2)} \div 10,1_{(2)}$

7. Resolver a equação abaixo em binário:

   $(101110_{(2)} - 10011_{(2)}) \times 11_{(2)} + 11100_{(2)} = X_{(2)}$

8. Resolver a equação abaixo:

   $(101110_{(2)} - 32_{(8)}) \times A_{(16)} + 2013_{(4)} = X_{(2)}$

9. Representar em complemento de 2 os números binários abaixo:

a.) **0** 1101
b.) **0** 10110
c.) **0** 111011
d.) **0** 1001010

10.Converter a representação em sinal e amplitude, e em ponto flutuante, abaixo para decimal:

0 10101 .1111001

11.Converter o valor decimal abaixo para ponto flutuante, em complemento de 2, com a menor representação:

-30.25

12.Converter o número decimal abaixo para ponto flutuante padrão IEEE-754:

30.25

13. Representar o número abaixo em binário com precisão de 4 casas decimais:

0.374

14.Qual a precisão decimal do número binário abaixo:

$1101{,}11011_{(2)}$

15.Qual dos números abaixo apresenta a melhor precisão decimal:

a.) $14{,}56_{(8)}$
b.) $101{,}010111_{(2)}$
c.) $1A{,}14_{(16)}$
d.) $32{,}331_{(4)}$

16.Decifrar a frase abaixo, codificada em ASCII:

4E616F20686120736567265646F20616C67756D2021

17.Codificar em ASCII a expressão abaixo:

((1+2+3)-(2+2))*2 = 4

18.Codificar em ASCII a frase abaixo:

O computador é uma máquina.

19.Identificar os símbolos contidos no seguinte padrão de bytes em código ASCII:

FE 81 80 80 8F 81 81 FF

20.Codificar a letra (F) em um padrão de 8 x 8 bits equivalentes a ***pixels***.