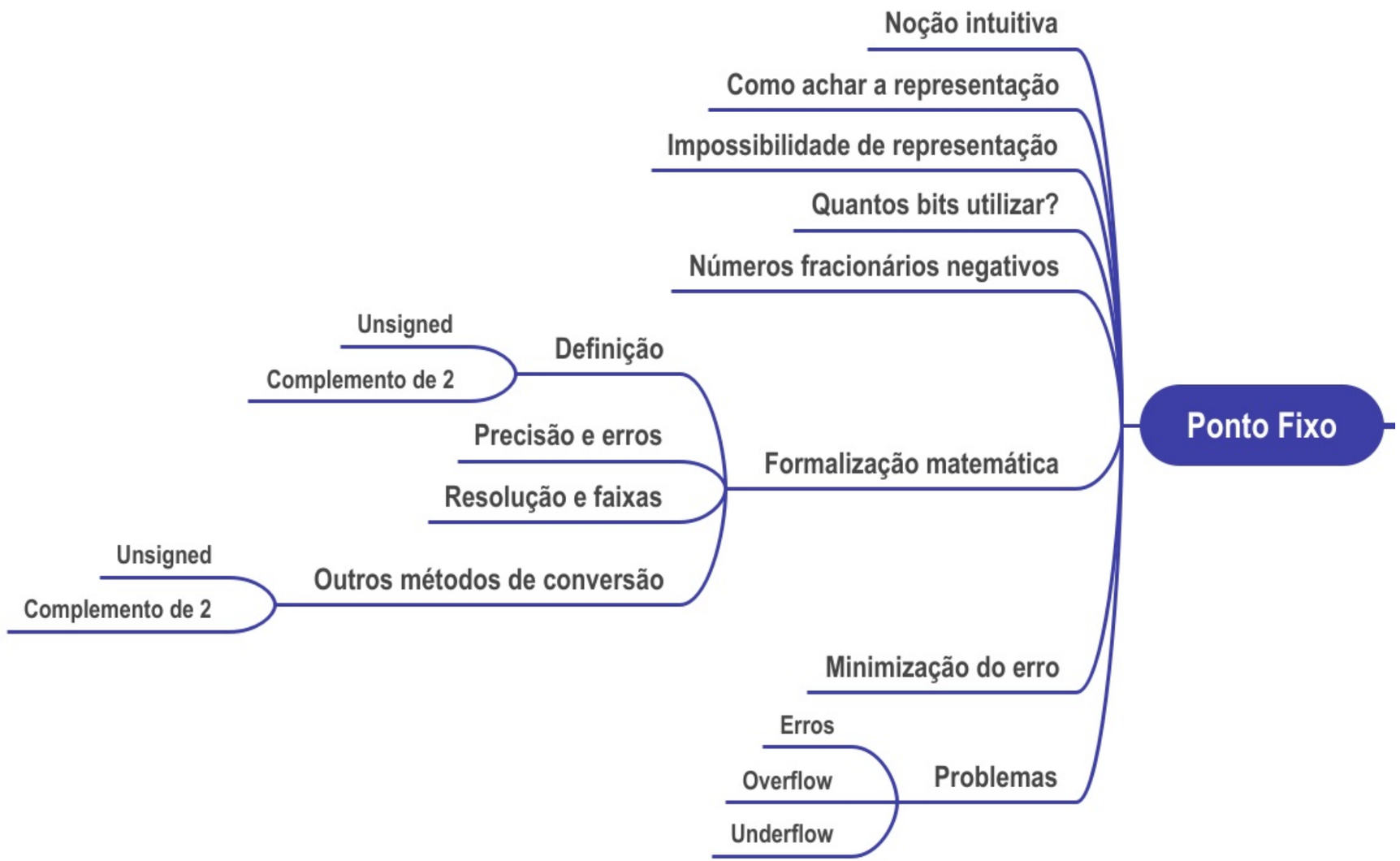
FUNDAMENTOS DA COMPUTAÇÃO

Representação de números fracionários em binário

Notação de Ponto Fixo



Notação de Ponto Fixo: semelhança com o sistema decimal

Semelhante à notação decimal para números fracionários:

- **Alguns algarismos representam a parte inteira**
- **Alguns algarismos representam a parte fracionária**
- **Há uma separação entre a parte inteira e a parte fracionária**
  (para visualização, nós usamos vírgula; computadores usam ponto)

Exemplos:

34.54	**-34.54**
0.94893	**-0.94893**
100.01	**-100.01**
1000.99	**-1000.99**
1.1	**-1.1**

Notação de Ponto Fixo: no sistema decimal

$$n_i \times 10^i$$

Valor posicional de números fracionários no sistema decimal:

- **Mesma coisa até a posição zero;**
- **Continua depois da posição zero, cada vez diminuindo uma posição**

$$19222.4357$$

$$19222.4357 = \left(1 \times 10^4 + 9 \times 10^3 + 2 \times 10^2 + 2 \times 10^1 + 2 \times 10^0\right) + \left(4 \times 10^{-1} + 3 \times 10^{-2} + 5 \times 10^{-3} + 7 \times 10^{-4}\right)$$

$$= (10000 + 9000 + 200 + 20 + 2) + \left(4 \times \frac{1}{10^1} + 3 \times \frac{1}{10^2} + 5 \times \frac{1}{10^3} + 7 \times \frac{1}{10^4}\right)$$

$$= 19222 + \left(\frac{4}{10} + \frac{3}{100} + \frac{5}{1000} + \frac{7}{10000}\right)$$

$$= 19222 + (0.4 + 0.03 + 0.005 + 0.0007)$$

$$= 19222 + 0.4357$$

$$= 19222.4357$$

Notação de Ponto Fixo: no sistema binário

$$n_i \times 2^i$$

Valor posicional de números fracionários no sistema binário:

- **Mesma coisa até a posição zero;**
- **Continua depois da posição zero, cada vez diminuindo uma posição**

$$0110.1100$$

$$0110.1100 = \left(0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0\right) + \left(1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4}\right)$$

$$= (0 + 4 + 2 + 0) + \left(1 \times \frac{1}{2^1} + 1 \times \frac{1}{2^2} + 0 \times \frac{1}{2^3} + 0 \times \frac{1}{2^4}\right)$$

$$= 6 + \left(1 \times \frac{1}{2} + 1 \times \frac{1}{4} + 0 \times \frac{1}{8} + 0 \times \frac{1}{16}\right)$$

$$= 6 + (1 \times 0.5 + 1 \times 0.250 + 0 \times 0.125 + 0 \times 0.0625)$$

$$= 6 + (0.5 + 0.250 + 0 + 0)$$

$$= 6 + 0.750$$

$$= 6.75$$

Notação de Ponto Fixo: exemplo

Outro exemplo: que número está representado abaixo, sabendo-se que é um binário em ponto fixo?

0000.0110

0,375

Notação de Ponto Fixo: como representar um número fracionário?

Para representar um número em binário com ponto fixo, devemos realizar dois processos:

- **Parte inteira:** **divisões sucessivas por 2**
- **Parte fracionária:** **multiplicações sucessivas por 2**

Ex.: representar o número 6,375 com 4 bits na parte inteira e 4 bits na parte fracionária:

0110.0110

Notação de Ponto Fixo: como representar um número fracionário?

Para representar um número em binário com ponto fixo, devemos realizar dois processos:

- **Parte inteira:** **divisões sucessivas por 2**
- **Parte fracionária:** **multiplicações sucessivas por 2**

Ex.: representar o número 27,09375 com 8 bits na parte inteira e 8 bits na parte fracionária:

00011011.00011000

Notação de Ponto Fixo: como representar um número fracionário?

Potência	Decimal
2^{0}	1
2^{-1}	0,5
2^{-2}	0,25
2^{-3}	0,125
2^{-4}	0,0625
2^{-5}	0,03125
2^{-6}	0,015625
2^{-7}	0,0078125
2^{-8}	0,00390625
2^{-9}	0,001953125
2^{-10}	0,0009765625
2^{-11}	0,00048828125
2^{-12}	0,000244140625
2^{-13}	0,0001220703125
2^{-14}	0,00006103515625
2^{-15}	0,000030517578125
2^{-16}	0,0000152587890625
2^{-17}	0,00000762939453125
2^{-18}	0,000003814697265625
2^{-19}	0,0000019073486328125
2^{-20}	0,00000095367431640625

Notação de Ponto Fixo: impossibilidade de representação

Da mesma maneira que não é possível representar todos os números decimais fracionários (ex.: 1/3), não é possível representar diversos números fracionários em binário. Por exemplo: vamos tentar representar 0,1 em binário:

Potência	Decimal	Bit i	Valor Posicional
2^{0}	1	0	0,0000000000000000
2^{-1}	0,5	0	0,0000000000000000
2^{-2}	0,25	0	0,0000000000000000
2^{-3}	0,125	0	0,0000000000000000
2^{-4}	0,0625	1	0,0625000000000000
2^{-5}	0,03125	1	0,0312500000000000
2^{-6}	0,015625	0	0,0000000000000000
2^{-7}	0,0078125	0	0,0000000000000000
2^{-8}	0,00390625	1	0,0039062500000000
2^{-9}	0,001953125	1	0,0019531250000000
2^{-10}	0,0009765625	0	0,0000000000000000
2^{-11}	0,00048828125	0	0,0000000000000000
2^{-12}	0,000244140625	1	0,0002441406250000
2^{-13}	0,0001220703125	1	0,0001220703125000
2^{-14}	0,00006103515625	0	0,0000000000000000
2^{-15}	0,000030517578125	0	0,0000000000000000
2^{-16}	0,0000152587890625	1	0,0000152587890625
2^{-17}	0,00000762939453125	1	0,0000076293945313
2^{-18}	0,000003814697265625	0	0,0000000000000000
2^{-19}	0,0000019073486328125	0	0,0000000000000000
2^{-20}	0,00000095367431640625	1	0,0000009536743164
			0,0999994277954102

$$0.0\overline{0011}\,\overline{0011}\,\overline{0011}\,\overline{0011}\ldots$$

Notação de Ponto Fixo: impossibilidade de representação

Vamos tentar representar 2,3 em binário:

Potência	Decimal	Bit i	Valor Posicional
2^{0}	**1**	0	0,0000000000000000
2^{-1}	**0,5**	0	0,0000000000000000
2^{-2}	**0,25**	1	0,2500000000000000
2^{-3}	**0,125**	0	0,0000000000000000
2^{-4}	**0,0625**	0	0,0000000000000000
2^{-5}	**0,03125**	1	0,0312500000000000
2^{-6}	**0,015625**	1	0,0156250000000000
2^{-7}	**0,0078125**	0	0,0000000000000000
2^{-8}	**0,00390625**	0	0,0000000000000000
2^{-9}	**0,001953125**	1	0,0019531250000000
2^{-10}	**0,0009765625**	1	0,0009765625000000
2^{-11}	**0,00048828125**	0	0,0000000000000000
2^{-12}	**0,000244140625**	0	0,0000000000000000
2^{-13}	**0,0001220703125**	1	0,0001220703125000
2^{-14}	**0,00006103515625**	1	0,0000610351562500
2^{-15}	**0,000030517578125**	0	0,0000000000000000
2^{-16}	**0,0000152587890625**	0	0,0000000000000000
2^{-17}	**0,00000762939453125**	1	0,0000076293945313
2^{-18}	**0,000003814697265625**	1	0,0000038146972656
2^{-19}	**0,0000019073486328125**	0	0,0000000000000000
2^{-20}	**0,00000095367431640625**	0	0,0000000000000000
			0,2999992370605470

$$10.0\overline{1001}\,\overline{1001}\,\overline{1001}\,\overline{1001}\ldots$$

Notação de Ponto Fixo: impossibilidade de representação

Vamos tentar representar 0,66 em binário:

Potência	Decimal	Bit i	Valor Posicional
2^{0}	1	0	0,0000000000000000
2^{-1}	0,5	1	0,5000000000000000
2^{-2}	0,25	0	0,0000000000000000
2^{-3}	0,125	1	0,1250000000000000
2^{-4}	0,0625	0	0,0000000000000000
2^{-5}	0,03125	1	0,0312500000000000
2^{-6}	0,015625	0	0,0000000000000000
2^{-7}	0,0078125	0	0,0000000000000000
2^{-8}	0,00390625	0	0,0000000000000000
2^{-9}	0,001953125	1	0,0019531250000000
2^{-10}	0,0009765625	1	0,0009765625000000
2^{-11}	0,00048828125	1	0,0004882812500000
2^{-12}	0,000244140625	1	0,0002441406250000
2^{-13}	0,0001220703125	0	0,0000000000000000
2^{-14}	0,00006103515625	1	0,0000610351562500
2^{-15}	0,000030517578125	0	0,0000000000000000
2^{-16}	0,0000152587890625	1	0,0000152587890625
2^{-17}	0,00000762939453125	1	0,0000076293945313
2^{-18}	0,000003814697265625	1	0,0000038146972656
2^{-19}	0,0000019073486328125	0	0,0000000000000000
2^{-20}	0,0000009536743164062 5	0	0,0000000000000000
			0,6599998474121090

$$0.\overline{10101000111101011100}0\ldots$$

Notação de Ponto Fixo: impossibilidade de representação

Nenhum computador consegue representar todos os números fracionários em notação de ponto fixo, sempre haverá um erro para muitos números que tentarmos representar em binário.

Em resumo, alguns números fracionários podem ter representação exata em ponto fixo, mas outros (a maioria!) não podem.

Quanto mais bits usarmos para a parte fracionária, menor o erro.

Por exemplo:

0,1	**0.00011**	**0,09375**
	0.000110011	**0,099609375**
	0.0001100110011	**0,0999755859375**
	0.000110011100110011	**0,0999984741210937 5**

Notação de Ponto Fixo: quantos bits usar?

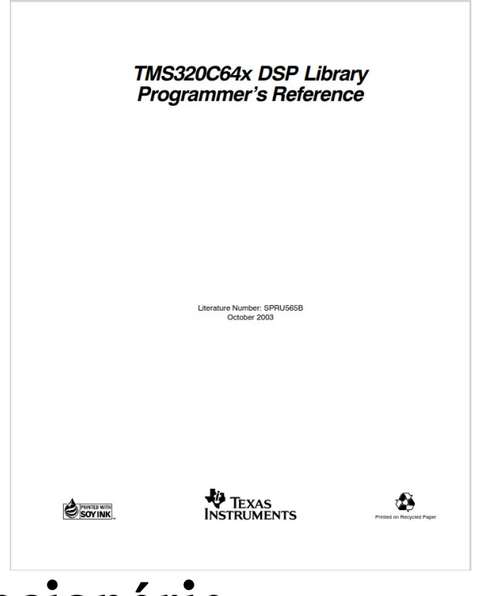
Como qualquer outra representação numérica em binário, números em ponto fixo são apenas uma coleção de bits. Não existe nenhuma maneira de determinar a localização do ponto binário, exceto através de algum acordo de interpretação e do contexto no qual o número está sendo utilizado.

Exemplo: o número abaixo está em notação de ponto fixo. Que número é esse?

01101100

0,84375 **1,6875** **3,375** **6,75** **13,5** **27,0**

Notação de Ponto Fixo: quantos bits usar?



Costumamos indicar a quantidade de bits na forma "p.f", onde:

p = número de bits da parte inteira

f = número de bits da parte fracionária (precisão)

Além disso usamos alguns modificadores, sendo os mais comuns:

Up.f binário unsigned com p bits na parte inteira e f bits na parte fracionária

Qp.f binário em complemento de 2, com p bits na parte inteira, incluindo o bit de sinal, e f bits na parte fracionária

Usos comuns da notação de ponto fixo:

- processamento de sinais digitais (Q1.15; Q1.31)

- transformada rápida de Fourier (Q1.31)

- conversores analógico/digital (U8.8)

Imagem: TECMOSLOJA (https://www.nanoeletronica.com.br/MLB-1784099632-processador-de-sinal-digital-tms320c6416glz-original-uni-_JM#&gid=1&pid=2)
Manual: Texas Instrument (https://www.ti.com/lit/ug/spru565b/spru565b.pdf)

Notação de Ponto Fixo: números fracionários negativos

O mais comum é utilizar complemento de 2 (sinal/magnitude é utilizado em algumas situações).

Exemplo: -2.375 em Q4.4:

1101.1010

Notação de Ponto Fixo: números fracionários negativos

Exemplo em Q2.3:

Binário	Decimal
10.000	-2,000
10.001	-1,875
10.010	-1,750
10.011	-1,625
10.100	-1,500
10.101	-1,375
10.110	-1,250
10.111	-1,125
11.000	-1,000
11.001	-0,875
11.010	-0,750
11.011	-0,625
11.100	-0,500
11.101	-0,375
11.110	-0,250
11.111	-0,125
00.000	0,000
00.001	0,125
00.010	0,250
00.011	0,375
00.100	0,500
00.101	0,625
00.110	0,750
00.111	0,875
01.000	1,000
01.001	1,125
01.010	1,250
01.011	1,375
01.100	1,500
01.101	1,625
01.110	1,750
01.111	1,875

Exemplo em Q1.3:

Binário	Decimal
1.000	-1,000
1.001	-0,875
1.010	-0,750
1.011	-0,625
1.100	-0,500
1.101	-0,375
1.110	-0,250
1.111	-0,125
0.000	0,000
0.001	0,125
0.010	0,250
0.011	0,375
0.100	0,500
0.101	0,625
0.110	0,750
0.111	0,875

Notação de Ponto Fixo: aritmética binária

A aritrmética binária em Qp.f funciona.

Exemplo: calcular 0,75 - 0,625 (0,125) em Q4.4:

0000.0010

Notação de Ponto Fixo: formalização matemática

Formalização: (unsigned)

Seja x um número binário fracionário positivo formado por n algarismos, em notação de **ponto fixo unsigned**, com p algarismos na parte inteira e f algarismos na parte fracionária, com $n = p + f$. Representaremos o vetor de algarismos desse número por $\vec{x}$, ou então por $[x_{p-1}, x_{p-2}, \ldots, x_1, x_0 \,.\, x_{-1}, x_{-2}, \ldots, x_{-f+1}, x_{-f}]$ para denotar os algarismos individuais dentro do vetor. A posição que um determinado algarismo ocupa em $\vec{x}$ será denominada por i, e a notação $\stackrel{\text{def}}{=\!=}$ significa "é definido por". Assim, a conversão $\mathrm{B}_{\mathrm{pf_u}}\mathrm{D}$ (binário em notação de ponto fixo unsigned para decimal) de um vetor binário $\vec{x}$ com tamanho n é dada por:

$$\mathrm{B}_{\mathrm{pf_u}}\mathrm{D} \stackrel{\text{def}}{=\!=} \sum_{i=-f}^{p-1} x_i 2^i$$

Exemplo: U1.3

$1.101_2 = 1{,}625_{10}$

Notação de Ponto Fixo: formalização matemática

Formalização: (complemento de 2)

Seja x um número binário fracionário formado por n algarismos, em notação de **ponto fixo em complemento de 2**, com p algarismos na parte inteira (incluindo o bit que representa o sinal) e f algarismos na parte fracionária, com $n = p + f$. Representaremos o vetor de algarismos desse número por $\vec{x}$, ou então por $[x_{p-1}, x_{p-2}, \ldots, x_1, x_0 \,.\, x_{-1}, x_{-2}, \ldots, x_{-f+1}, x_{-f}]$ para denotar os algarismos individuais dentro do vetor. A posição que um determinado algarismo ocupa em $\vec{x}$ será denominada por i, e a notação $\stackrel{\text{def}}{=\!=}$ significa "é definido por". Assim, a conversão $\mathrm{B}_{\mathrm{pf_{c2}}}\mathrm{D}$ (binário em notação de ponto fixo em complemento de dois para decimal) de um vetor binário $\vec{x}$ com tamanho n é dada por:

$$\mathrm{B}_{\mathrm{pf_{c2}}}\mathrm{D} \stackrel{\text{def}}{=\!=} -(2^{p-1})x_{p-1} + \left(\sum_{i=-f}^{p-2} x_i 2^i\right)$$

Exemplo:

Q1.3

1.101_2 = $-0{,}375_{10}$

Notação de Ponto Fixo: formalização matemática

Precisão:
(representação exata)

Terão representação exata, em ponto fixo, somete os números cuja parte fracionária puder ser escrita no formato

$$q \times 2^s$$

com $q, s \in \mathbb{Z}$ e $s \leq -1$. Outros números só poderão ser aproximados com algum grau de erro determinado pela quantidade de algarismos na parte fracionária.

Erro na representação:

Seja d um número decimal, R uma função que converte um número decimal em binário em ponto fixo, e V uma função que converte de binário em ponto fixo para decimal. O erro absoluto (e_a) e o erro relativo (e_r) da representação de d em ponto fixo são dados por:

$$e_a = | \, V(R(d)) - d \, |$$

$$e_r = \left| \frac{V(R(d)) - d}{d} \right|$$

0,1 **0.00011** **0,09375**

0.0001100110011001100 **0,099998474121093750**

Notação de Ponto Fixo: formalização matemática

Resolução:
- **menor intervalo**
- **valor mais próx. de 0**

O número de bits na parte fracionária, f, determina a **resolução** (r) da representação em ponto fixo, ou seja, o menor intervalo que podemos distinguir (cada incremento no número binário aumenta o valor por r) e, assim, também representa o valor mais próximo de 0 que conseguimos representar. A resolução é dada por:

$$r = 2^{-f}$$

Faixa Up.f: $[0\ ;\ 2^p - r]$

Faixa Qp.f: $[-2^{p-1}\ ;\ 2^{p-1} - r]$

Notação de Ponto Fixo: formalização matemática

Outros modos de fazer a conversão de binário em ponto fixo unsigned para decimal:

Seja x um número binário fracionário positivo formado por n algarismos, em notação de **ponto fixo unsigned**, com p algarismos na parte inteira e f algarismos na parte fracionária, com $n = p + f$. Representaremos o vetor de algarismos desse número por $\vec{x}$, ou então por $[x_{p-1}, x_{p-2}, \ldots, x_1, x_0 \,.\, x_{-1}, x_{-2}, \ldots, x_{-f+1}, x_{-f}]$ para denotar os algarismos individuais dentro do vetor. Se w representa uma constante que informa a posição do ponto binário em bits a partir da esquerda do número, e se considerarmos i como a posição unsigned do número (contada da direita para a esquerda, começando em 0 e ignorando o ponto binário), então a conversão $\mathrm{B}_{\mathrm{pf_u}}\mathrm{D}$ (binário em notação de ponto fixo unsigned para decimal) de um vetor binário $\vec{x}$ com tamanho n pode ser dada por:

$$\mathrm{B}_{\mathrm{pf_u}}\mathrm{D} = 2^w \sum_{i=0}^{n-1} x_i 2^{-n+i}$$

Exemplo:
U1.3

$1.101_2 = 1{,}625_{10}$

Seja x um número binário fracionário positivo formado por n algarismos, em notação de **ponto fixo unsigned**, com p algarismos na parte inteira e f algarismos na parte fracionária, com $n = p + f$. Representaremos o vetor de algarismos desse número por $\vec{x}$, ou então por $[x_{p-1}, x_{p-2}, \ldots, x_1, x_0 \,.\, x_{-1}, x_{-2}, \ldots, x_{-f+1}, x_{-f}]$ para denotar os algarismos individuais dentro do vetor. Se r representa a resolução (2^{-f}) e se considerarmos i como a posição unsigned do número (contada da direita para a esquerda, começando em 0 e ignorando o ponto binário), então a conversão $\mathrm{B}_{\mathrm{pf_u}}\mathrm{D}$ (binário em notação de ponto fixo unsigned para decimal) de um vetor binário $\vec{x}$ com tamanho n pode ser dada por:

$$\mathrm{B}_{\mathrm{pf_u}}\mathrm{D} = r \sum_{i=0}^{n-1} x_i 2^i$$

Notação de Ponto Fixo: formalização matemática

Outros modos de fazer a conversão de binário em ponto fixo em comp. de 2 para decimal:

Seja x um número binário fracionário formado por n algarismos, em notação de **ponto fixo em complemento de 2**, com p algarismos na parte inteira (incluindo o bit que representa o sinal) e f algarismos na parte fracionária, com $n = p + f$. Representaremos o vetor de algarismos desse número por $\vec{x}$, ou então por $[x_{p-1}, x_{p-2}, \ldots, x_1, x_0 \,.\, x_{-1}, x_{-2}, \ldots, x_{-f+1}, x_{-f}]$ para denotar os algarismos individuais dentro do vetor. Se w representa uma constante que informa a posição do ponto binário em bits a partir da esquerda do número, e se considerarmos i como a posição unsigned do número (contada da direita para a esquerda, começando em 0 e ignorando o ponto binário), então a conversão $\mathrm{B}_{\mathrm{pf_{c2}}}\mathrm{D}$ (binário em notação de ponto fixo em complemento de dois para decimal) de um vetor binário $\vec{x}$ com tamanho n pode ser dada por:

$$\mathrm{B}_{\mathrm{pf_{c2}}}\mathrm{D} = 2^w \left[-(2^{-n+(n-1)})x_{n-1} + \sum_{i=0}^{n-2} x_i 2^{-n+i} \right]$$

Exemplo:

Q1.3

$1.101_2 = -0{,}375_{10}$

Seja x um número binário fracionário positivo formado por n algarismos, em notação de **ponto fixo em complemento de 2**, com p algarismos na parte inteira (incluindo o bit que representa o sinal) e f algarismos na parte fracionária, com $n = p + f$. Representaremos o vetor de algarismos desse número por $\vec{x}$, ou então por $[x_{p-1}, x_{p-2}, \ldots, x_1, x_0 \,.\, x_{-1}, x_{-2}, \ldots, x_{-f+1}, x_{-f}]$ para denotar os algarismos individuais dentro do vetor. Se r representa a resolução (2^{-f}) e se considerarmos i como a posição unsigned do número (contada da direita para a esquerda, começando em 0 e ignorando o ponto binário), então a conversão $\mathrm{B}_{\mathrm{pf_{c2}}}\mathrm{D}$ (binário em notação de complemento de dois unsigned para decimal) de um vetor binário $\vec{x}$ com tamanho n pode ser dada por:

$$\mathrm{B}_{\mathrm{pf_{c2}}}\mathrm{D} = r \left[-(2^{n-1})x_{n-1} + \sum_{i=0}^{n-2} x_i 2^i \right]$$

Notação de Ponto Fixo: minimização do erro

Para minimizar o erro na representação de números fracionários Qp.f, ao invés de fazermos a multiplicação sucessiva por 2, podemos usar o seguinte procedimento:

1) Multiplicar o número por 2^f
2) Arredondar para o inteiro mais próximo
3) Converter o inteiro encontrado para binário unsigned
4) Ajustar a posição do ponto binário
5) Converter para complemento de 2, se necessário

Ex.: 1.484 em Q2.3

Notação de Ponto Fixo: problemas

Erro na representação!

Overflow!

Ex.: -2.484 em Q2.3

Underflow!

O número a ser representado é tão próximo de zero que a notação escolhida não consegue diferenciar o número a ser representado do próprio zero. Ex.: 0.001 em Q2.3.

"Desperdício"!

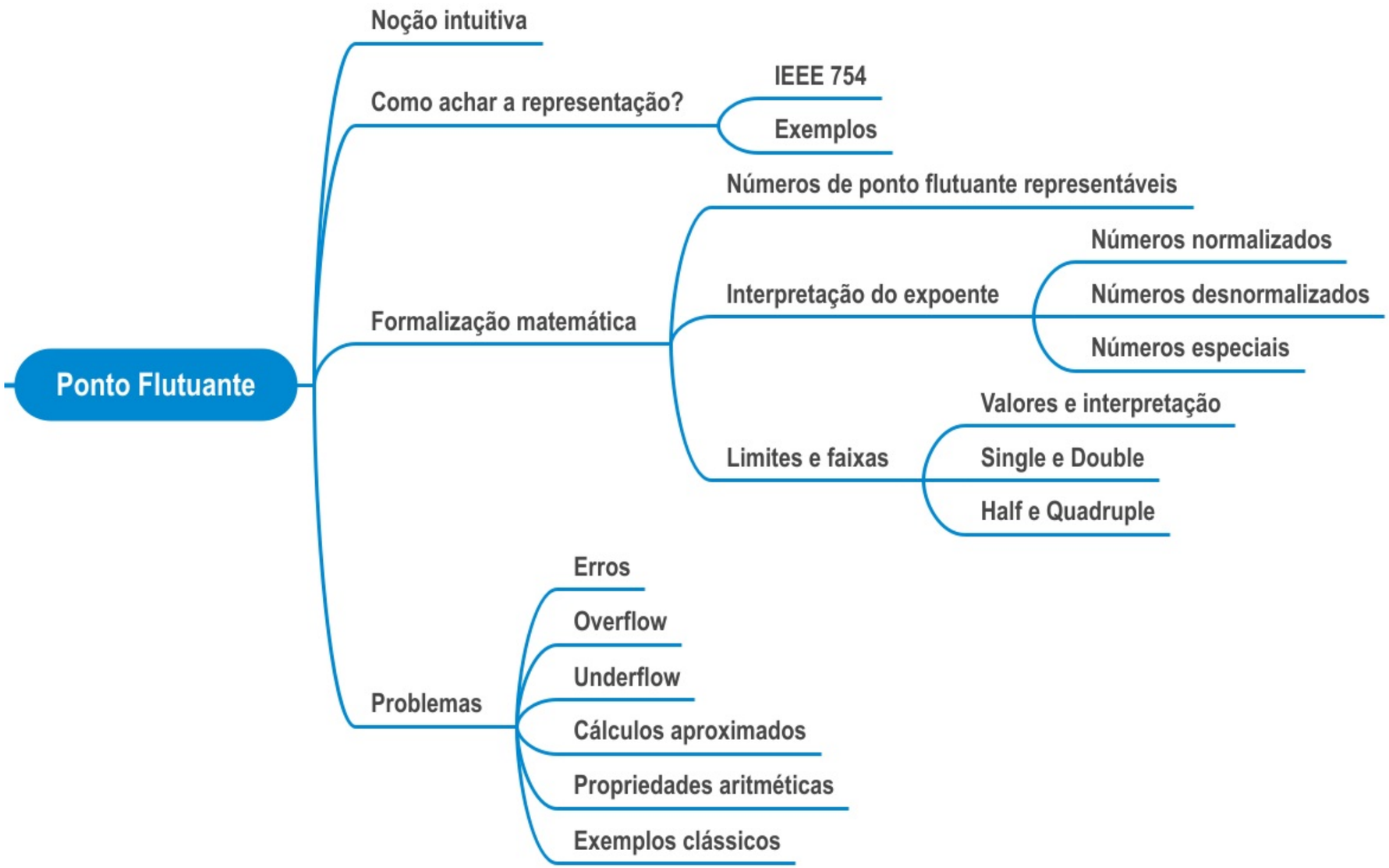
Com os computadores de hoje, com word de 64 bits, utilizar Up.f ou Qp.f pode ser desperdício de bits de armazenamento.

Limitações!

Não representa números muito grandes ou pequenos.

$$0.0000000001$$

Notação de Ponto Flutuante



Notação de Ponto Flutuante: semelhança com a notação científica

A notação de ponto flutuante foi criada para podermos representar os números reais muito maiores do que zero ou muito próximos a zero (positivos ou negativos). É uma notação semelhante à notação científica que usamos no sistema decimal.

As sete **constantes fundamentais** do Sistema Internacional de Unidades (SI) permitem que qualquer outra unidade possa ser escrita através de uma constante por si mesma, ou através de produtos ou quocientes dessas constantes. O sistema internacional de unidades é o sistema no qual

$$
\begin{aligned}
\text{Frqüência de transição hiperfina do césio 137, } \Delta V_{Cs} &= 9.192\,631\,770 \times 10^{9}\,\mathrm{Hz} \\
\text{Velocidade da luz no vácuo, } c &= 2.997\,924\,580 \times 10^{8}\,\mathrm{m\,s^{-1}} \\
\text{Constante de Planck, } h &= 6.626\,070\,150 \times 10^{-34}\,\mathrm{J\,s} \\
\text{Carga elementar, } e &= 1.602\,176\,634 \times 10^{-19}\,\mathrm{C} \\
\text{Constante de Boltzmann, } k &= 1.380\,649\,000 \times 10^{-23}\,\mathrm{J\,K^{-1}} \\
\text{Constante de Avogadro, } N_A &= 6.022\,140\,760 \times 10^{23}\,\mathrm{mol^{-1}} \\
\text{Eficácia luminosa, } K_{cd} &= 6.830\,000\,000 \times 10^{2}\,\mathrm{lm\,W^{-1}},
\end{aligned}
$$

onde o hertz (Hz), joule (J), coulomb (C), lumen (lm) e watt (W), são relacionados às unidades segundo (s), metro (m), kilograma (kg), ampere (A), kelvin (K), mol (mol) e candela (cd), de acordo com: $\mathrm{Hz} = \mathrm{s^{-1}}$, $\mathrm{J} = \mathrm{kg\,m^2\,s^{-2}}$, $\mathrm{C} = \mathrm{A\,s}$, $\mathrm{lm} = \mathrm{cd\,sr}$, e $\mathrm{W} = \mathrm{kg\,m^2\,s^{-3}}$.

Notação de Ponto Flutuante: semelhança com a notação científica

Notação científica: expressa números muito grandes ou pequenos para serem escritos convenientemente na forma decimal. Base é 10.

A forma geral é

$$m \times 10^n \qquad (\pm m \times 10^{\pm n})$$

onde: $(m \in \mathbb{R}^*) \wedge (n \in \mathbb{Z})$

m = significando (mantissa): número real diferente de zero

n = expoente: número inteiro

A forma normalizada é aquela onde n é escolhido de tal forma que o valor absoluto de m seja maior do que ou igual a 1 e menor do que 10. Assim, na forma normalizada, temos:

$$(m \in \mathbb{R}^* \ \wedge \ 1 \leq |m| < 10) \wedge (n \in \mathbb{Z})$$

Notação de Ponto Flutuante: semelhança com a notação científica

Na notação de ponto flutuante, a base é 2:

A forma geral é

$$m \times 2^n \qquad (\pm m \times 2^{\pm n})$$

onde: $(m \in \mathbb{R}^*) \wedge (n \in \mathbb{Z})$

- **m** = **significando** (mantissa): **número real diferente de zero**
- **n** = **expoente**: **número inteiro**

A forma normalizada é aquele onde n é escolhido de tal forma que o MSB de m (em binário) seja igual a 1:

$$(m \in \mathbb{R}^* \ \wedge \ \mathrm{MSB}(m_2) = 1) \wedge (n \in \mathbb{Z})$$

$$0.0625_{10} = 0.0001_2$$

$$5.6875_{10} = 101.1011_2$$

Notação de Ponto Flutuante: como representar em bits?

Problema: a partir da notação científica em base 2, como criar um padrão de bits para representar:

- **significando** (que pode ser positivo ou negativo)
- **expoente** (que pode ser positivo ou negativo)

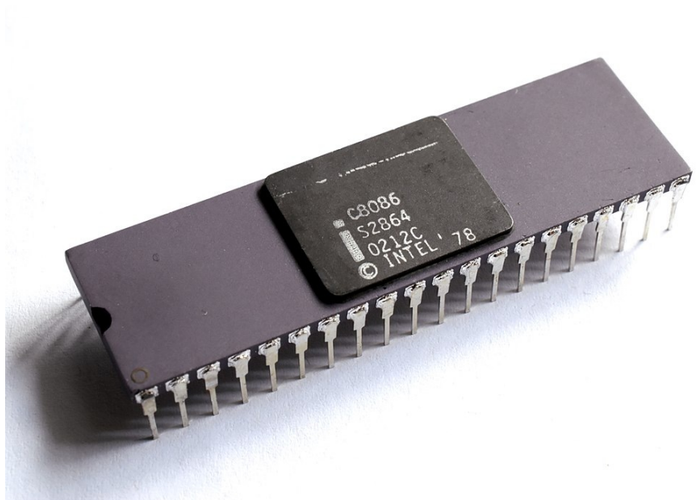
Até meados da década de 1980, cada fabricante:

- **criava seus próprios padrões de bits para representar a notação de ponto flutuante**
- **criava suas próprias operações**
- **não se importavam muito com a exatidão dos cálculos, priorizando a velocidade e facilidade de implementação**

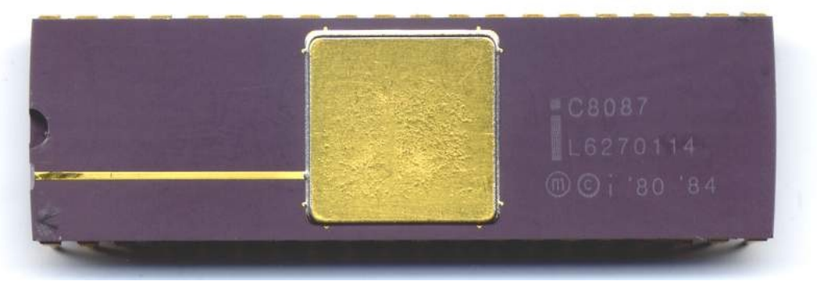
Notação de Ponto Flutuante: como representar em bits?

Em 1976 a Intel estava projetando seu chip 8087, que era uma unidade auxiliar (co-processador) para dar suporte de ponto flutuante ao processador 8086. Para ajudar a projetar um padrão de ponto flutuante para o 8087, a Intel contratou o professor William Morton Kahan (Universidade da Califórnia, em Berkeley). A Intel permitiu que Kahan se juntasse com um comitê do Institute of Electrical and Electronics Engineers (IEEE) que adotou o padrão criado por Kahan, com pequenas modificações, criando assim o padrão IEEE 754.

George M. Bergman, na Wikipedia (https://en.wikipedia.org/wiki/File:William_Kahan_2008.jpg)



Thomas Nguyen, na Wikipedia (https://en.wikipedia.org/wiki/File:Intel_C8086.jpg)



Dirk Oppelt, na Wikipedia (https://en.wikipedia.org/wiki/File:Intel_C8087.jpg)

Notação de Ponto Flutuante: IEEE 754

IEEE 754: Standard for Floating-Point Arithmetic

- Publicado em 1985, com revisões em 2008 e 2019.
- Praticamente todos os computadores atuais.
- Define os **formatos** (padrões de bits), **operações**, **conversões** e **exceções** dos números de ponto flutuante.
- É uma **aproximação** dos números reais e por isso:
  - representa um **subconjunto finito** dos reais
  - **algumas propriedades aritméticas não valem**
- Especificação:

$$\{-\infty \ldots 0 \ldots +\infty\} \qquad (-1)^s \times m \times b^e$$

$$\Downarrow$$

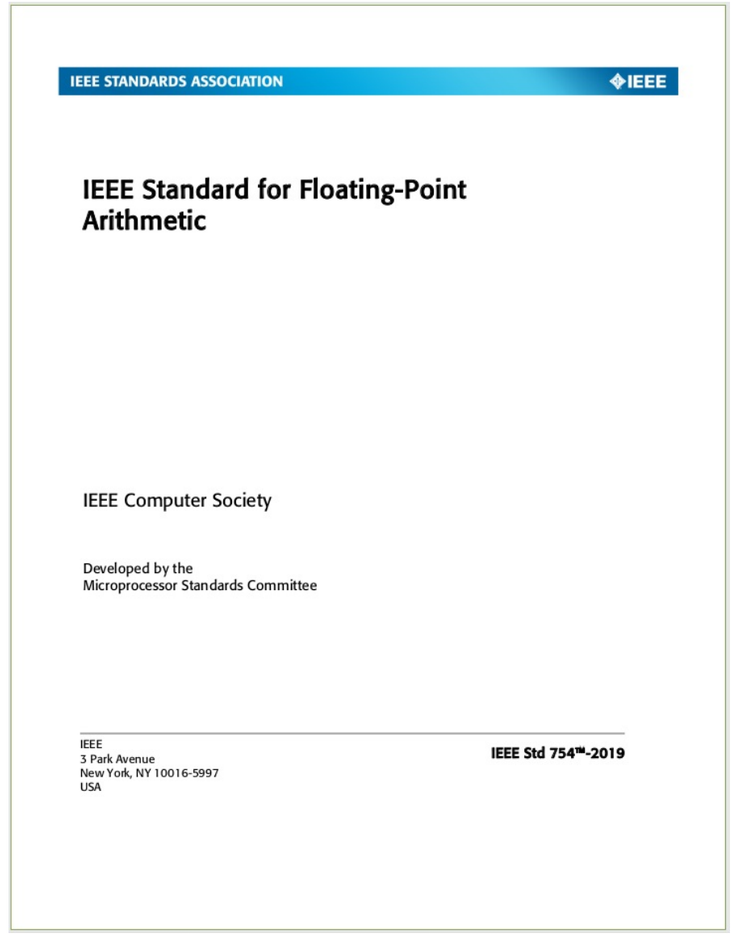
$$\{-\infty \ldots -0\} \cup \{+0 \ldots +\infty\} \cup \mathrm{NaN}$$

$$\Downarrow$$

$$(\textbf{sinal}, \textbf{expoente}, \textbf{significando}) \cup \{-\infty, +\infty\} \cup \mathrm{qNaN} \cup \mathrm{sNaN}$$

$$\Downarrow$$

$$0111000\ldots$$



IEEE STANDARDS ASSOCIATION IEEE

IEEE Standard for Floating-Point Arithmetic

IEEE Computer Society

Developed by the
Microprocessor Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 754™-2019

Notação de Ponto Flutuante: IEEE 754

O formato padrão de sinal, expoente e significando para representar

$$(\text{sinal}, \text{expoente}, \text{significando}) \cup \{-\infty, +\infty\} \cup \text{qNaN} \cup \text{sNaN}$$

é dado por:

$$(-1)^s \times m \times b^e$$

Mas, como a base será sempre 2, então temos:

$$(-1)^s \times m \times 2^e$$

Assim:

- **s** sinal do significando (0 = positivo; 1 = negativo)
- **m** significando (número binário fracionário)
- **e** expoente (número inteiro positivo ou negativo)

Atenção! O expoente e tem 3 funções diferentes, e é calculado de modo diferente dependendo da função:

- numérica para números normalizados
- numérica para números não normalizados (subnormais)
- indicação de números especiais

Notação de Ponto Flutuante: IEEE 754

O padrão estabeleceu 4 formatos com graus de precisão diferente. A precisão é dada pela quantidade de algarismos do significando: quanto mais algarismos, mais preciso é o número.

Quadruple Precision: 128 bits no total; 1 bit para o sinal, 15 bits para o expoente com viés de 16383, e 112 bits para a fração.

127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
s	expoente															fração																																															
(1)	(15 bits, viés de 16383)															(112 bits: bits 111 até 64)																																															

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
fração (continuação)																																																															
(112 bits: bits 63 até 0)																																																															

Double Precision: 64 bits no total; 1 bit para o sinal, 11 bits para o expoente com viés de 1023, e 52 bits para a fração.

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	expoente											fração																																																		
(1)	(11 bits, viés de 1023)											(52 bits)																																																		

Single Precision: 32 bits no total; 1 bit para o sinal, 8 bits para expoente com viés de 127, e 23 bits para a fração.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	expoente								fração																						
(1)	(8 bits, viés de 127)								(23 bits)																						

Half Precision: 16 bits no total; 1 bit para sinal, 5 bits para expoente com viés de 15, e 10 bits para a fração.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	expoente					fração									
(1)	(5 bits, viés de 15)					(10 bits)									

Half precision:	**11**	**bits de precisão,**	**16 bits no total**
Single precision:	**24**	**bits de precisão,**	**32 bits no total**
Double precision:	**53**	**bits de precisão,**	**64 bits no total**
Quadruple precision:	**113**	**bits de precisão,**	**128 bits no total**

Notação de Ponto Flutuante: exemplo em Half Precision

Para entender como a notação de ponto flutuante funciona, vamos representar o número 228,0. Faremos diversos refinamentos, começando com versões simples, que não seguem totalmente o padrão IEEE 754, até chegar na versão correta.

Half Precision: 16 bits no total; 1 bit para sinal, 5 bits para expoente com viés de 15, e 10 bits para a fração.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	expoente					fração									
(1)	(5 bits, viés de 15)					(10 bits)									

Notação de Ponto Flutuante: exemplo em Half Precision

Agora vamos ajustar o significando. Como todos os números estarão normalizados, o MSB do significando sempre será 1. Assim, para ganhar um bit de precisão, não representamos o MSB, ele fica implícito, e só a fração é representada!

Half Precision: 16 bits no total; 1 bit para sinal, 5 bits para expoente com viés de 15, e 10 bits para a fração.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	expoente					fração									
(1)	(5 bits, viés de 15)					(10 bits)									

$$\text{Forma normalizada} = \underbrace{\underbrace{1}_{\text{MSB}} \cdot \overbrace{bbb\ldots bbb}^{\text{fração}}}_{\text{significando}} \times 2^e$$

Notação de Ponto Flutuante: exemplo em Half Precision

Agora vamos ajustar o expoente. Note que o sinal s é apenas para o significando. E como fazer para representar expoentes negativos? Complemento de 1? Complemento de 2? Notação enviesada? O padrão IEEE 754 utiliza notação enviesada para o expoente e!

Half Precision: 16 bits no total; 1 bit para sinal, 5 bits para expoente com viés de 15, e 10 bits para a fração.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	expoente					fração									
(1)	(5 bits, viés de 15)					(10 bits)									

$$\text{Viés} = 2^{(\text{bits do } e)-1} - 1$$

Notação de Ponto Flutuante: exemplo em Single Precision

Representar -0,75:

Single Precision: 32 bits no total; 1 bit para o sinal, 8 bits para expoente com viés de 127, e 23 bits para a fração.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	expoente								fração																						
(1)	(8 bits, viés de 127)								(23 bits)																						

Notação de Ponto Flutuante: outro exemplo em Single Precision

Que número é esse?

Single Precision: 32 bits no total; 1 bit para o sinal, 8 bits para expoente com viés de 127, e 23 bits para a fração.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s	expoente								fração																						
(1)	(8 bits, viés de 127)								(23 bits)																						

Notação de Ponto Flutuante: outro exemplo em Single Precision

Represente a carga elétrica elementar em precisão simples, sabendo que:

$$1.602\,176\,634 \times 10^{-19}\,\mathrm{C} \approx 1.011\,110\,100\,100\,110\,110\,100\,010\,100\,100\,001\,100\,000\,101\,100\,101\,001\,1 \times 2^{-63}\,\mathrm{C}$$

Single Precision: 32 bits no total; 1 bit para o sinal, 8 bits para expoente com viés de 127, e 23 bits para a fração.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	expoente								fração																						
(1)	(8 bits, viés de 127)								(23 bits)																						

Notação de Ponto Flutuante: formalização matemática

O conjunto finito de números de ponto flutuante que será representável em um formato é determinado pelos seguintes parâmetros:

- b, a base, 2 ou 10
- p, o número de dígitos do significando m (precisão)
- $emax$, o expoente e máximo
- $emin$, o expoente e mínimo, definido como $1 - emax$ em todos os formatos

Parâmetros	Formatos Binários (b = 2)				Formatos Decimais (b = 10)	
	binary16	binary32	binary64	binary128	decimal64	decimal128
	Half	Single	Double	Quadruple		
p	11	24	53	113	16	34
emax	+15	+127	+1023	+16383	+384	+6144
emin	-14	-126	-1022	-16382	-383	-6143

Dentro de cada formato os seguintes números de ponto flutuante devem ser representados:

- +0, −0 e números de ponto flutuante diferentes de zero no formato "$(-1)^s \times m \times b^e$", onde:
  - s é 0 ou 1
  - e é qualquer inteiro $emin \leq e \leq emax$
  - m é a mantissa representada por uma seqüência de bits com a forma "$d_0 \bullet d_1 d_2 \ldots d_{p-1}$", onde d_i é um número inteiro $0 \leq d_i < b$ (assim, $0 \leq m < b$)
- dois infinitos: $+\infty$ e $-\infty$
- dois tipos de NaN: qNaN e sNaN

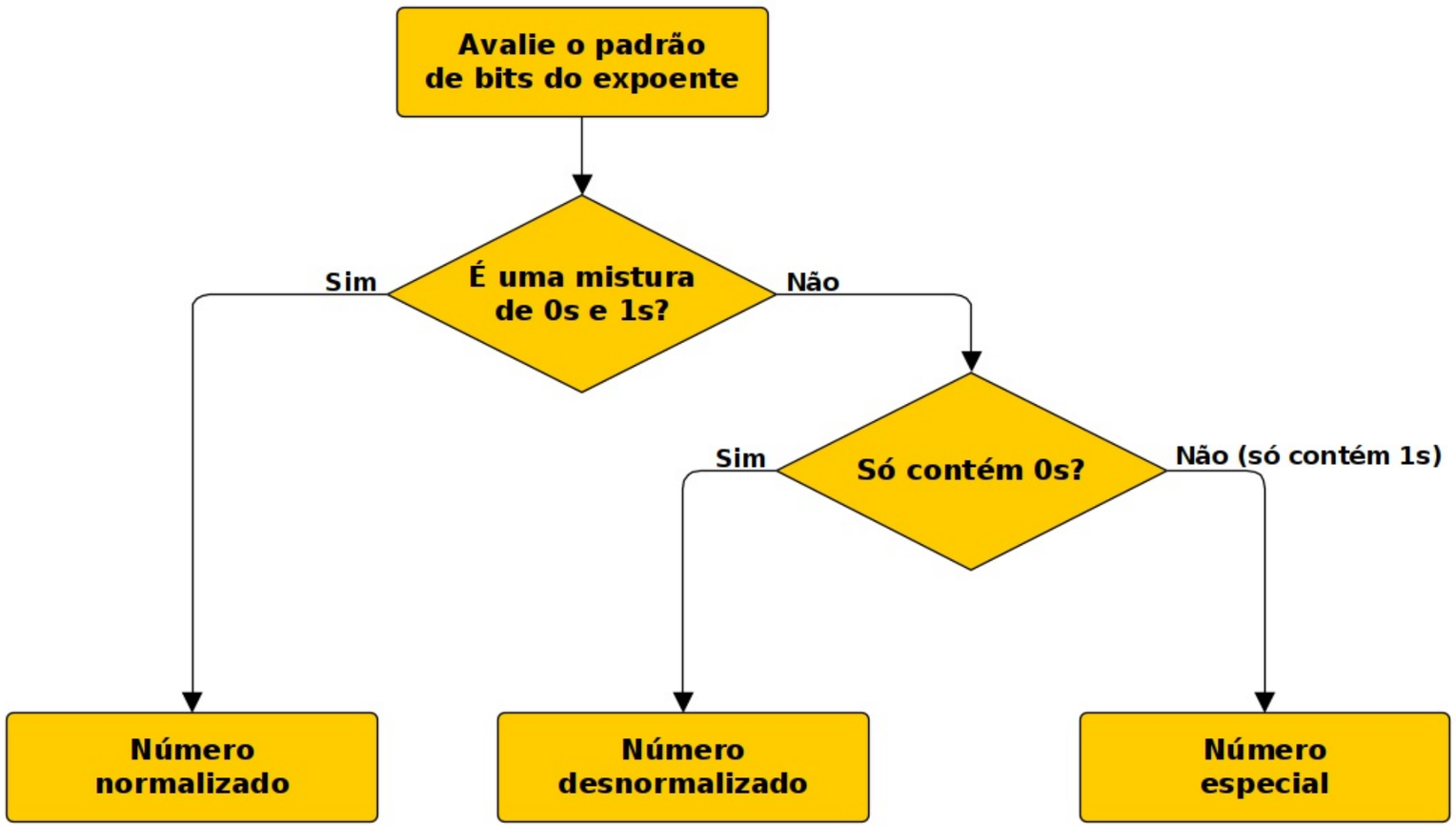
$$\{-\infty \ldots -0\} \cup \{+0 \ldots +\infty\} \cup \mathrm{NaN}$$

Notação de Ponto Flutuante: formalização matemática

Em cada formato, dependendo do padrão de bits do expoente, temos 3 situações diferentes:

- **número normalizados**
- **números não normalizados (desnormalizados, subnormais)**
- **números especiais**

Dependendo da situação, o expoente e é calculado de modo diferente e a interpretação do formato muda.



Notação de Ponto Flutuante: formalização matemática

Se o número é normalizado:

- **O expoente é uma mistura de 0s e 1s**
- **Representa números diferentes de zero, desde que esses números não estejam próximos de 0.0:** $\{x \in \mathbb{R} \mid x \neq 0 \ \wedge x \text{ não esteja próximo de } 0.0\}$
- **O expoente tem viés de:** $2^{(\text{bits do } e)-1} - 1$
- **O significando tem sempre MSB = 1 e é implícito, ou seja não é representado no formato. Como só representamos a fração f, o significando deve ser interpretado como:** $m = 1 + f$

$$(-1)^s \times m \times 2^e$$

$$(-1)^s \times (1 + f) \times 2^{e - \text{viés}}$$

Half Precision: 16 bits no total; 1 bit para sinal, 5 bits para expoente com viés de 15, e 10 bits para a fração.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	1	0	1	0	0	1	1	0	1	1	1
s	expoente					fração									
(1)	(5 bits, viés de 15)					(10 bits)									

Notação de Ponto Flutuante: formalização matemática

Se o número é desnormalizado:

- **O expoente só tem 0s**
- **Representa 0 e números próximos:** $\{x \in \mathbb{R} \mid x = +0.0 \ \vee \ x = -0.0 \ \vee \ x \text{ esteja próximo de } 0.0\}$
- **O expoente é diferente:** $e = 1 - \text{viés} = 1 - (2^{(\text{bits do } e)-1} - 1)$
- **O significando tem sempre MSB = 0 e é implícito, ou seja não é representado no formato. Como só representamos a fração f, o significando deve ser interpretado como:** $m = 0 + f = f$

$$(-1)^s \times m \times 2^e$$

$$(-1)^s \times f \times 2^{1-\text{viés}}$$

Half Precision: 16 bits no total; 1 bit para sinal, 5 bits para expoente com viés de 15, e 10 bits para a fração.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
s	expoente					fração									
(1)	(5 bits, viés de 15)					(10 bits)									

Notação de Ponto Flutuante: formalização matemática

Se o número é especial:

- **O expoente só tem 1s**
- **Representa infinito e NaN:** $\{-\infty, +\infty, \mathrm{NaN}\}$
- **O expoente não tem nenhum valor**
- **O significando não tem nenhum valor, é interpretado como:**
  - **se todos os bits forem 0: infinito (+ ou - de acordo com o sinal)**
  - **se algum bit for 1: NaN (+ ou - de acordo com o sinal)**

Half Precision: 16 bits no total; 1 bit para sinal, 5 bits para expoente com viés de 15, e 10 bits para a fração.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
s	expoente					fração									
(1)	(5 bits, viés de 15)					(10 bits)									

Notação de Ponto Flutuante: formalização matemática

Limites e faixas de representação

Descrição	s	e	f	Valor	Interpretação
Infinito negativo	1	11...11	0...00	$-\infty$	$-\infty$
Maior normalizado negativo	1	11...10	1...11	$-(2-\epsilon)\times 2^{e-\text{viés}}$	Menor número negativo
Um negativo	1	01...11	0...00	-1×2^{0}	-1.0
Menor normalizado negativo	1	00...01	0...00	$-1\times 2^{e-\text{viés}}$	
Maior desnormalizado negativo	1	00...00	1...11	$-(1-\epsilon)\times 2^{1-\text{viés}}$	
Menor desnormalizado negativo	1	00...00	0...01	$-2^{-w}\times 2^{1-\text{viés}}$	Maior número negativo
Zero negativo	1	00...00	0...00	$+0$	-0.0
Zero positivo	0	00...00	0...00	-0	$+0.0$
Menor desnormalizado positivo	0	00...00	0...01	$2^{-w}\times 2^{1-\text{viés}}$	Menor número positivo
Maior desnormalizado positivo	0	00...00	1...11	$(1-\epsilon)\times 2^{1-\text{viés}}$	
Menor normalizado positivo	0	00...01	0...00	$1\times 2^{e-\text{viés}}$	
Um positivo	0	01...11	0...00	1×2^{0}	$+1.0$
Maior normalizado positivo	0	11...10	1...11	$(2-\epsilon)\times 2^{e-\text{viés}}$	Maior número positivo
Infinito positivo	0	11...11	0...00	$+\infty$	$+\infty$

- $(1-\epsilon)$: um número menor do que 1 e maior do que 0, mas extremamente próximo do 1
- $(2-\epsilon)$: um número menor do que 2 e maior do que 1, mas extremamente próximo do 2
- w: quantidade de bits da fração

Notação de Ponto Flutuante: formalização matemática

Limites e faixas de representação: single e double precision

					Single Precision		Double Precision	
Descrição	s	e	f	**Valor**	**Valor**	**Aprox. Decimal**	**Valor**	**Aprox. Decimal**
Infinito negativo	1	11...11	0...00	$-\infty$	$-\infty$		$-\infty$	
Maior normalizado negativo	1	11...10	1...11	$-(2-\epsilon)\times 2^{e-\text{viés}}$	$-(2-\epsilon)\times 2^{127}$	-3.4×10^{38}	$-(2-\epsilon)\times 2^{1023}$	-1.8×10^{308}
Um negativo	1	01...11	0...00	-1×2^{0}	-1×2^{0}	-1.0	-1×2^{0}	-1.0
Menor normalizado negativo	1	00...01	0...00	$-1\times 2^{e-\text{viés}}$	-1×2^{-126}	-1.2×10^{-38}	-1×2^{-1022}	-2.2×10^{-308}
Maior desnormalizado negativo	1	00...00	1...11	$-(1-\epsilon)\times 2^{1-\text{viés}}$	$-(1-\epsilon)\times 2^{-126}$	-1.2×10^{-38}	$-(1-\epsilon)\times 2^{-1022}$	-2.2×10^{-308}
Menor desnormalizado negativo	1	00...00	0...01	$-2^{-w}\times 2^{1-\text{viés}}$	$-2^{-23}\times 2^{-126}$	-1.4×10^{-45}	$-2^{-52}\times 2^{-1022}$	-4.9×10^{-324}
Zero negativo	1	00...00	0...00	$+0$	-0	-0.0	-0	-0.0
Zero positivo	0	00...00	0...00	-0	$+0$	$+0.0$	$+0$	$+0.0$
Menor desnormalizado positivo	0	00...00	0...01	$2^{-w}\times 2^{1-\text{viés}}$	$2^{-23}\times 2^{-126}$	1.4×10^{-45}	$2^{-52}\times 2^{-1022}$	4.9×10^{-324}
Maior desnormalizado positivo	0	00...00	1...11	$(1-\epsilon)\times 2^{1-\text{viés}}$	$(1-\epsilon)\times 2^{-126}$	1.2×10^{-38}	$(1-\epsilon)\times 2^{-1022}$	2.2×10^{-308}
Menor normalizado positivo	0	00...01	0...00	$1\times 2^{e-\text{viés}}$	1×2^{-126}	1.2×10^{-38}	1×2^{-1022}	2.2×10^{-308}
Um positivo	0	01...11	0...00	1×2^{0}	1×2^{0}	1.0	1×2^{0}	1.0
Maior normalizado positivo	0	11...10	1...11	$(2-\epsilon)\times 2^{e-\text{viés}}$	$(2-\epsilon)\times 2^{127}$	3.4×10^{38}	$(2-\epsilon)\times 2^{1023}$	1.8×10^{308}
Infinito positivo	0	11...11	0...00	$+\infty$	$+\infty$		$+\infty$	

- $(1-\epsilon)$: um número menor do que 1 e maior do que 0, mas extremamente próximo do 1
- $(2-\epsilon)$: um número menor do que 2 e maior do que 1, mas extremamente próximo do 2
- w: quantidade de bits da fração

Notação de Ponto Flutuante: formalização matemática

Limites e faixas de representação: single precision

Single Precision:

$$\{\underbrace{-(2-\epsilon)\times 2^{127} \ \ldots \ -2^{-23}\times 2^{-126}}_{\text{negativos}},\ \underbrace{-0,\ +0}_{\text{zero}},\ \underbrace{2^{-23}\times 2^{-126} \ \ldots \ (2-\epsilon)\times 2^{127}}_{\text{positivos}}\} \cup \{-\infty,\ +\infty,\ \mathrm{NaN}\}$$

$$\{\underbrace{-3.4\times 10^{38} \ \ldots \ -1.4\times 10^{-45}}_{\text{negativos}},\ \underbrace{-0,\ +0}_{\text{zero}},\ \underbrace{1.4\times 10^{-45} \ \ldots \ 3.4\times 10^{38}}_{\text{positivos}}\} \cup \{-\infty,\ +\infty,\ \mathrm{NaN}\}$$

Notação de Ponto Flutuante: formalização matemática

Limites e faixas de representação: double precision

Double Precision:

$$\{\underbrace{-(2-\epsilon)\times 2^{1023} \ \ldots \ -2^{-52}\times 2^{-1022}}_{\text{negativos}}, \ \underbrace{-0, \ +0}_{\text{zero}}, \ \underbrace{2^{-52}\times 2^{-1022} \ \ldots \ (2-\epsilon)\times 2^{1023}}_{\text{positivos}}\} \cup \{-\infty, \ +\infty, \ \text{NaN}\}$$

$$\{\underbrace{-1.8\times 10^{308} \ \ldots \ -4.9\times 10^{-324}}_{\text{negativos}}, \ \underbrace{-0, +0}_{\text{zero}}, \ \underbrace{4.9\times 10^{-324} \ \ldots \ 1.8\times 10^{308}}_{\text{positivos}}\} \cup \{-\infty, \ +\infty, \ \text{NaN}\}$$

Notação de Ponto Flutuante: formalização matemática

Limites e faixas de representação: half e quadruple precision

					Half Precision		Quadruple Precision	
Descrição	s	e	f	**Valor**	**Valor**	**Aprox. Decimal**	**Valor**	**Aprox. Decimal**
Infinito negativo	1	11...11	0...00	$-\infty$	$-\infty$		$-\infty$	
Maior normalizado negativo	1	11...10	1...11	$-(2-\epsilon)\times 2^{e-\text{viés}}$	$-(2-\epsilon)\times 2^{15}$	-65504	$-(2-\epsilon)\times 2^{16383}$	-1.2×10^{4932}
Um negativo	1	01...11	0...00	-1×2^{0}	-1×2^{0}	-1.0	-1×2^{0}	-1.0
Menor normalizado negativo	1	00...01	0...00	$-1\times 2^{e-\text{viés}}$	-1×2^{-14}	-6.1×10^{-5}	-1×2^{-16382}	-3.4×10^{-4932}
Maior desnormalizado negativo	1	00...00	1...11	$-(1-\epsilon)\times 2^{1-\text{viés}}$	$-(1-\epsilon)\times 2^{-14}$	-6.1×10^{-5}	$-(1-\epsilon)\times 2^{-16382}$	-3.4×10^{-4932}
Menor desnormalizado negativo	1	00...00	0...01	$-2^{-w}\times 2^{1-\text{viés}}$	$-2^{-10}\times 2^{-14}$	-6.0×10^{-8}	$-2^{-122}\times 2^{-16382}$	-6.5×10^{-4966}
Zero negativo	1	00...00	0...00	$+0$	-0	-0.0	-0	-0.0
Zero positivo	0	00...00	0...00	-0	$+0$	$+0.0$	$+0$	$+0.0$
Menor desnormalizado positivo	0	00...00	0...01	$2^{-w}\times 2^{1-\text{viés}}$	$2^{-10}\times 2^{-14}$	6.0×10^{-8}	$2^{-112}\times 2^{-16382}$	6.5×10^{-4966}
Maior desnormalizado positivo	0	00...00	1...11	$(1-\epsilon)\times 2^{1-\text{viés}}$	$(1-\epsilon)\times 2^{-14}$	6.1×10^{-5}	$(1-\epsilon)\times 2^{-16382}$	3.4×10^{-4932}
Menor normalizado positivo	0	00...01	0...00	$1\times 2^{e-\text{viés}}$	1×2^{-14}	6.1×10^{-5}	1×2^{-16382}	3.4×10^{-4932}
Um positivo	0	01...11	0...00	1×2^{0}	1×2^{0}	1.0	1×2^{0}	1.0
Maior normalizado positivo	0	11...10	1...11	$(2-\epsilon)\times 2^{e-\text{viés}}$	$(2-\epsilon)\times 2^{15}$	65504	$(2-\epsilon)\times 2^{16383}$	1.2×10^{4932}
Infinito positivo	0	11...11	0...00	$+\infty$	$+\infty$		$+\infty$	

- $(1-\epsilon)$: um número menor do que 1 e maior do que 0, mas extremamente próximo do 1
- $(2-\epsilon)$: um número menor do que 2 e maior do que 1, mas extremamente próximo do 2
- w: quantidade de bits da fração

Notação de Ponto Flutuante: formalização matemática

Limites e faixas de representação: half precision

Half Precision:

$$\{\underbrace{-(2-\epsilon)\times 2^{15} \ \ldots \ -2^{-10}\times 2^{-14}}_{\text{negativos}},\ \underbrace{-0,\ +0}_{\text{zero}},\ \underbrace{2^{-10}\times 2^{-14} \ \ldots \ (2-\epsilon)\times 2^{15}}_{\text{positivos}}\} \cup \{-\infty,\ +\infty,\ \text{NaN}\}$$

$$\{\underbrace{-65504 \ \ldots \ -6.0\times 10^{-8}}_{\text{negativos}},\ \underbrace{-0, +0}_{\text{zero}},\ \underbrace{6.0\times 10^{-8} \ \ldots \ 65504}_{\text{positivos}}\} \cup \{-\infty,\ +\infty,\ \text{NaN}\}$$

Notação de Ponto Flutuante: formalização matemática

Limites e faixas de representação: quadruple precision

Quadruple Precision:

$$\{\underbrace{-(2-\epsilon)\times 2^{16383} \ \ldots \ -2^{-112}\times 2^{-16382}}_{\text{negativos}},\ \underbrace{-0,\ +0}_{\text{zero}},\ \underbrace{2^{-112}\times 2^{-16382} \ \ldots \ (2-\epsilon)\times 2^{16383}}_{\text{positivos}}\} \cup \{-\infty,\ +\infty,\ \text{NaN}\}$$

$$\{\underbrace{-1.2\times 10^{4932} \ \ldots \ -6.5\times 10^{-4966}}_{\text{negativos}},\ \underbrace{-0,+0}_{\text{zero}},\ \underbrace{6.5\times 10^{-4966} \ \ldots \ 1.2\times 10^{4932}}_{\text{positivos}}\} \cup \{-\infty,\ +\infty,\ \text{NaN}\}$$

Notação de Ponto Flutuante: formalização matemática

Limites e faixas de representação: resumo (somente faixa positiva)

Formato	De	Até
Half	6.0×10^{-8}	65504
Single	1.4×10^{-45}	3.4×10^{38}
Double	4.9×10^{-324}	1.8×10^{308}
Quadruple	6.5×10^{-4966}	1.2×10^{4932}

Notação de Ponto Flutuante: problemas

Erro na representação!

Overflow!

Underflow!

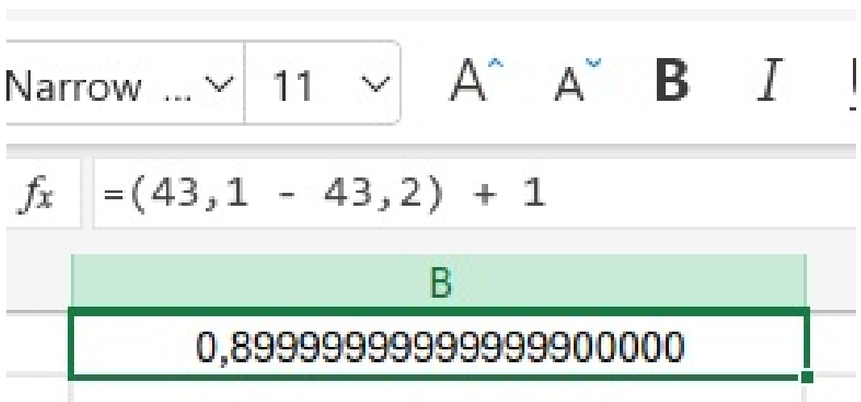
O número a ser representado é tão próximo de zero que o formato escolhido não consegue diferenciar o número a ser representado do próprio zero (o expoente não cabe no padrão de bits).

Limitações!

Cálculos são aproximações.
Arredondamentos.
Algumas propriedades aritméticas não valem.

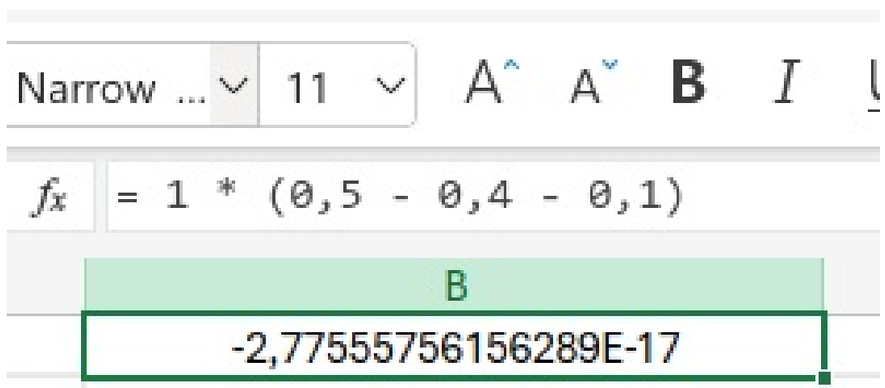
Notação de Ponto Flutuante: problemas

$$(43.1 - 43.2) + 1 = 0.9$$



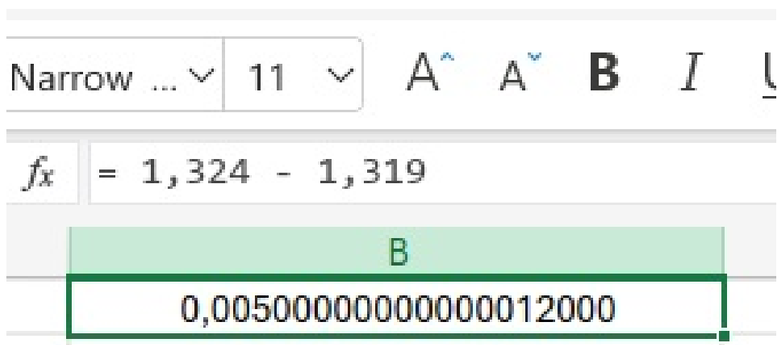
Notação de Ponto Flutuante: problemas

$$1 * (0.5 - 0.4 - 0.1) = 0$$



Notação de Ponto Flutuante: problemas

$$1.324 - 1.319 = 0.005$$



Notação de Ponto Flutuante: problemas

floats.c - GNU Emacs at cosmos

```c
#include <stdio.h>

int main(void)
{
    float x = 0.0;
    double y = 0.0;

    for (int i = 0; i < 10000; i++)
    {
        x = x + (1.0/10000);
        y = y + (1.0/10000);
    }

    printf("x (single precision) = %2.30f\n", x);
    printf("y (double precision) = %2.30f\n", y);

    return 0;
}
```

```
[abrantesasf@cosmos ~/cr6100b]$ ./floats
x (single precision) = 1.000053524971008300781250000000
y (double precision) = 0.999999999999906186154419174272
```

Notação de Ponto Flutuante: problemas

A imprecisão nos cálculos ocorre por:

- **Limitação na representação**
- **Arredondamento**

IEEE 754 define 4 métodos de arredondamento:

- **Mais próximo ou par**
- **Em direção ao zero**
- **Para baixo**
- **Para cima**

Obs.: calculadoras fazem outro tipo de arredondamento, por exemplo: as calculadoras da Hewlett Packard fazem

- **0-4 para baixo**
- **5-9 para cima**

	IEEE 754				Hewlett Packard
Número	Mais próximo ou par	Direção ao zero	Para baixo	Para cima	Calculadoras
1,40	1	1	1	2	1
1,60	2	1	1	2	2
1,50	2	1	1	2	2
2,50	2	2	2	3	3
-1,50	-2	-1	-2	-1	-2

Notação de Ponto Flutuante: problemas

Algumas propriedades aritméticas não valem.

Adição:

- **Associativa:** **não**
- **Comutativa:** **sim**
- **Inverso:** **sim**
- **Fechamento:** **sim**
- **Monotonicidade:** **sim**

Multiplicação:

- **Associativa:** **não**
- **Comutativa:** **sim**
- **Distributiva:** **não**
- **Inverso:** **sim**
- **Fechamento:** **sim**
- **Monotonicidade:** **sim**

Notação de Ponto Flutuante: problemas

associativa.c - GNU Emacs at cosmos

```c
#include <stdio.h>

int main(void)
{
    printf("(-1e20 + 1e20) + 3.14 = %.2f\n\n",  (-1e20 + 1e20) + 3.14);
    printf("-1e20 + (1e20 + 3.14) = %.2f\n\n\n", -1e20 + (1e20 + 3.14));

    printf("(1e20 x 1e20) x 1e-20 = %.2f\n\n",   (float) (1e20 * 1e20) * 1e-20);
    printf("1e20 x (1e20 x 1e-20) = %.2f\n\n\n", (float) 1e20 * (1e20 * 1e-20));

    return 0;
}
```

```
[abrantesasf@cosmos ~/cr6100b]$ ./associativa
(-1e20 + 1e20) + 3.14 = 3.14

-1e20 + (1e20 + 3.14) = 0.00


(1e20 x 1e20) x 1e-20 = inf

1e20 x (1e20 x 1e-20) = 100000002004087734272.00
```

Notação de Ponto Flutuante: problemas

distributiva.c - GNU Emacs at cosmos

```c
#include <stdio.h>

int main(void)
{
    printf("1e20 x (1e20 - 1e20) = %.2f\n\n",
           (float) -1e20 * (float) (1e20 - 1e20));

    printf("1e20 x 1e20 - 1e20 x 1e20 = %.2f\n\n\n",
           (float) (1e20 * 1e20) - (float) (1e20 * 1e20));

    return 0;
}
```

```
[abrantesasf@cosmos ~/cr6100b]$ ./distributiva
1e20 x (1e20 - 1e20) = -0.00

1e20 x 1e20 - 1e20 x 1e20 = -nan
```

Notação de Ponto Flutuante: problemas

Exemplo clássico: Patriot missile

U. S. Army, na Wikimedia Commons
(https://commons.wikimedia.org/wiki/File:Patriot_missile_launch_b.jpg)

25/01/1991, Guerra do Golfo

O exército do Iraque lançou um míssil Scud contra um acampamento do exército americano, na Arábia Saudita.

O exército americano lançou seu míssil Patriot, de interceptação terra-ar, para destruir o Scud. Alguns segundos depois o Patriot não acertou o Scud, que caiu no acampamento e matou 28 militares americanos, ferindo mais várias dezenas.

Investigações mostraram que o problema ocorreu em uma rotina de cálculo de tempo: o Patriot utilizava uma constante multiplicativa de 1/10 (0,1), que não tem representação exata em ponto flutuante. Assim, a cada multiplicação, o erro na contagem do tempo era acumulado. No momento do lançamento o erro era de 0,342 seguntos, fazendo que o Patriot fosse desviado em 687 metros, errando o Scud.

Notação de Ponto Flutuante: problemas

Exemplo clássico: Ariane-5 rocket

CNN, arquivado no Internet Archive
(https://web.archive.org/web/20000819090542/http://www.cnn.com/WORLD/9606/04/rocket.explode/)

04/06/1996, Guiana Francesa

O foguete Ariane-5, da Agência Espacial Européia, explodiu depois de 40 segundos da decolagem, causando um prejuízo de 7,5 bilhões de dólares.

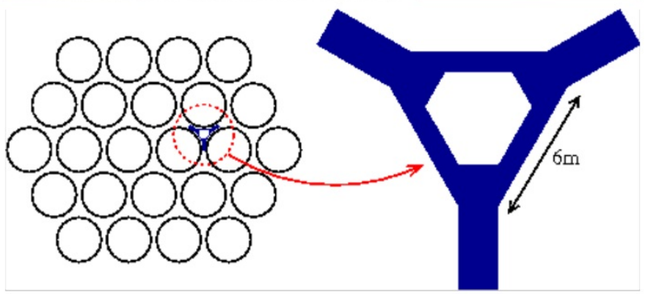
Investigações mostraram que o problema foi um simples overflow: um dos sistemas de direcionamento realizava uma conversão de um número de ponto flutuante de 64 bits para um número inteiro signed de 16 bits (2^15 = 32.768 valores possíveis) no computador de controle. Durante os primeiros segunda após a decolagem a aceleração era baixa, então a conversão podia ser realizada. Com o aumento da aceleração chegou um momento em que a conversão causou um overflow no inteiro signed de 16 bits, causando o total descontrole do foguete que acabou explodindo.

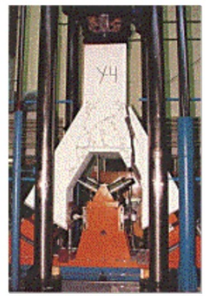
O programa tinha sido testado e utilizado com sucesso no Ariane-4, mas o Ariane-5 tinha um motor mais potente, com maior aceleração, gerando números maiores para o computador de controle. Ninguém percebeu que isso causaria um overflow!

Notação de Ponto Flutuante: problemas

Exemplo clássico: Plataforma Sleipner A





Douglas Arnold, Universidade de Minnesota
(https://www-users.cse.umn.edu/~arnold/disasters/sleipner.html)

23/08/1991, Mar do Norte

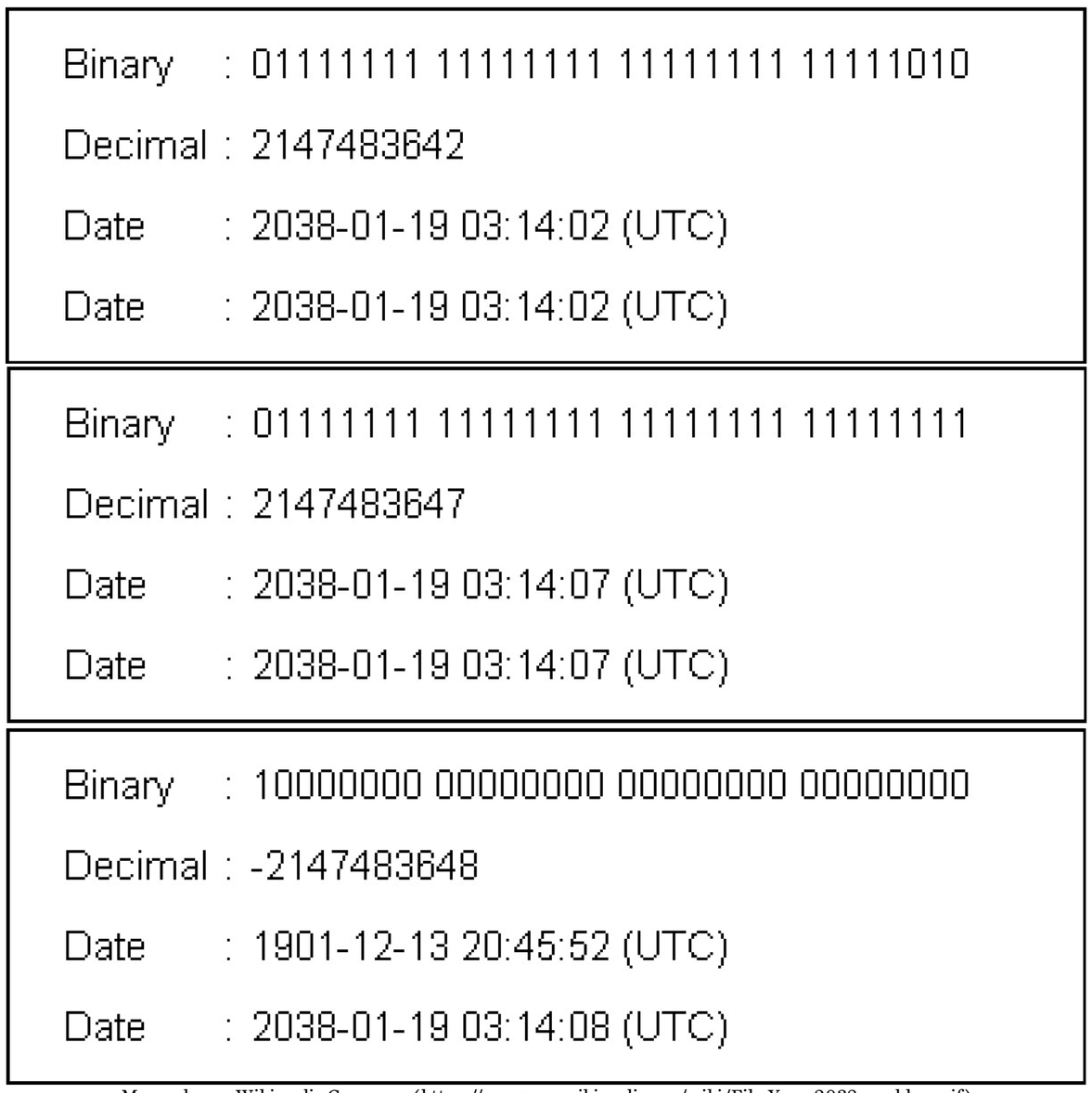
A plataforma "Sleipner A" se partiu e afundou durante a fase final de construção. Em um momento, quando o caso foi abaixado para 62 metros de profundidade, uma parede das células de sustentação rachou e o mar começou a inundar a plataforma. Quando a plataforma atingiu 210 metros as câmaras de flutuabilidade implodiram e os escombros atingiram o leito do oceano causando um terremoto de magnitude 3 na escala Richter.

Investigações mostraram que o erro ocorreu no projeto das células de sustentação, devido à imprecisão de cálculos executados pelo software NASTRAN (NASA Structural Analysis). Essa imprecisão fez com que as paredes de concreto fossem projetadas muito finas para resistir à pressão, subestimando a carga em cerca de 50% em algumas áreas críticas!

O software NASTRAN era utilizado há anos, com sucesso até mesmo em aplicações semelhantes, mas o algoritmo não foi suficiente para o projeto complexo dessa plataforma.

Notação de Ponto Flutuante: problemas

Exemplo clássico: o bug do ano 2038 ("epocalipse")



Monaneko, na Wikimedia Commons (https://commons.wikimedia.org/wiki/File:Year_2038_problem.gif)

Os computadores Unix e derivados, medem o tempo como o número de segundos decorridos desde o marco "Unix epoch", (1970-01-01 00:00:00 UTC).

O problema é que essa contagem é armazenada, desde a década de 1970, em um número inteiro signed de 32 bits, que pode armazenar valores de -2^31 até 2^31 - 1. Esse limite será alcançado em 2038-01-19 03:14:07 UTC.

O próximo segundo após esse limite causará um overflow no sistema de contagem de tempo, e os computadores passarão a entender a data como 1901-12-13 20:45:52 UTC, causando problemas em diversos sistemas.

Por sorte os sistemas mais novos já utilizam um inteiro signed de 64 bits para contar o tempo (novo "epocalipse" somente daqui a 292 bilhões de anos!) mas os sistemas antigos e legados, se não forem atualizados, podem sofrer o "epocalipse" de 2038.

Binários Fracionários: em resumo