Insper

Sistemas Hardware-Software

Aula 2 – Números fracionários

2020 – Engenharia

Igor Montagner, Fábio Ayres

Números fracionários

- Representamos inteiros usando bits, sendo que cada posição equivale a uma multiplicação por potência de 2.

Unsigned

$$B2U(X) = \sum_{i=0}^{w-1} x_i \cdot 2^i$$

Two's Complement

$$B2T(X) = -x_{w-1} \cdot 2^{w-1} + \sum_{i=0}^{w-2} x_i \cdot 2^i$$

Bit de sinal

Exercício - precisão

Usando Python, teste o seguinte:

```
>> 0.1
>> format(0.1, '.24f')
```

```
>>0.9
>> format(0.9, '.24f')
```

```
>> 0.5
>> format(0.5, '.24f')
```

Pergunta: todos números são representáveis de maneira exata?

Exercício - dígitos finitos

Calcule (na mão) a seguinte soma:

$$9 + 0.9 + \overbrace{0.1 + 0.1 + \ldots + 0.1}^{17 \text{ vezes}}$$

Regra: sempre que fizer uma soma, arredonde o resultado para 2 dígitos apenas. Exemplos: $2.34 \rightarrow 2.3$, $10.1 \rightarrow 10$

a) Some da esquerda para direita

a) Some da direita para esquerda

O que aconteceu?

Números fracionários

- Precisam ser escritos usando bits
- Usam número fixo de bits (32 ou 64)
- Precisão é um problema: o espaçamento entre números consecutivos pode mudar conforme números ficam maiores

Alguma ideia de como resolver isso?

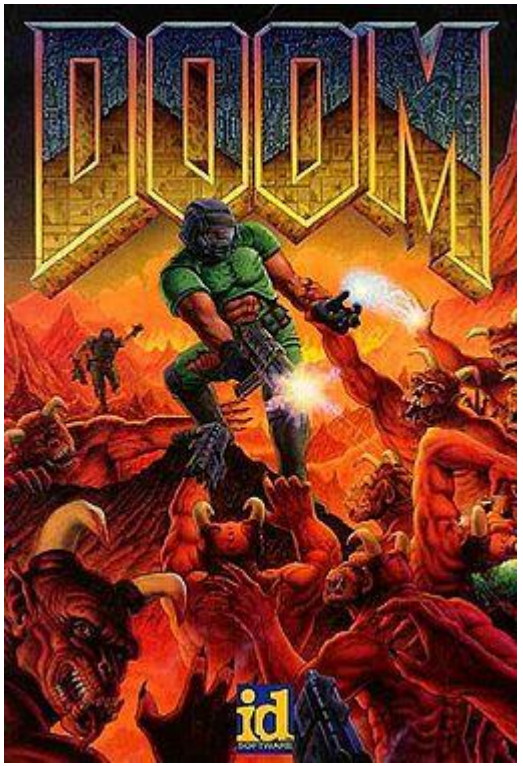
Ponto fixo

Número fixo de bits para a parte decimal

- 16 bits para a parte inteira
- 16 bits para a parte fracionária

Equivalente a dividir a representação binária por uma fator escala!

Expoente é fixo!



Insper

Revisão: números decimais fracionários

$$314.15$$

$$3x10^{2} + 1x10^{1} + 4x10^{0} + 1x10^{-1} + 5x10^{-2}$$

$$-314.15$$

$$-(3x10^{2} + 1x10^{1} + 4x10^{0} + 1x10^{-1} + 5x10^{-2})$$

Números binários fracionários

Já sabemos o que significam números inteiros representados em base 2:

$10110_2 = 1x2^4 + ... + 0x2^0 = 22_{10}$

$-1101_2 = -(1x2^3 + ... + 1x2^0) = -13_{10}$

Sabemos também o que significam números fracionários decimais

O que seriam números fracionários em base 2?

$$11.0110_2 = ?$$

Números binários fracionários

Já sabemos o que significam números inteiros representados em base 2:

$10110_2 = 1x2^4 + ... + 0x2^0 = 22_{10}$

$-1101_2 = -(1x2^3 + ... + 1x2^0) = -13_{10}$

Sabemos também o que significam números fracionários decimais

O que seriam números fracionários em base 2?

$$11.0110_2 = ?$$

Resposta:

$$11.0110 = 1x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2} + 1x2^{-3} + 0x2^{-4}$$
$$= 2 + 1 + 0 + 0.25 + 0.125 + 0$$
$$= 3.37510$$

Exercício

1. Calcule o valor dos seguintes valores binários em decimal:

a) 101.11_2

b) -0.001_2

c) 11.0_2

Exercício

1. Calcule o valor dos seguintes valores binários em decimal:

a) 101.11_2 5,75

b) -0.001_2 -0,125

c) 11.0_2 3

Números decimais em notação científica

Expoente

$$314.15 = 3.1415 \times 10^{2}$$

Mantissa Base

$$-314.15 = -1 \times 3.1415 \times 10^{2}$$

$$= (-1)^{1} \times 3.1415 \times 10^{2}$$

Bit de sinal

Números decimais em notação científica

Todo número real pode ser arredondado para um número real escrito da seguinte forma:

$$x = (-1)^s \times M \times 10^E$$

onde:

- s é o bit de sinal (0: positivo, 1: negativo)
- M é a mantissa
- E é o expoente

Números binários em notação científica

Vamos fazer a mesma coisa que fizemos para os números decimais:

$$x = (-1)^s \times M \times 2^E$$

onde:

- s é o bit de sinal (0: positivo, 1: negativo)
- M é a mantissa
- E é o expoente

Exemplo

$$10.1_2$$
$$= (-1)^0 \times 1.01_2 \times 2^1$$

$$s = 0$$
$$M = 1.01_2 = 1.25$$
$$E = 1$$

Exercício

Determine o bit de sinal, a mantissa e o expoente (de base 2) dos seguintes números reais representados em base 2:

101.11_2

-0.001_2

11.0_2

Exercício

Valor	Bit de sinal (s)	Mantissa (M)	Expoente (E)
101.11_2			
-0.001_2			
11.0_2			

Exercício

Valor	Bit de sinal (s)	Mantissa (M)	Expoente (E)
101.11_2 $= (-1)^0 \times 1.0111_2 \times 2^2$			
-0.001_2			
11.0_2			

Exercício

Valor	Bit de sinal (s)	Mantissa (M)	Expoente (E)
101.11_2 $= (-1)^0 \times 1.0111_2 \times 2^2$	0		
-0.001_2			
11.0_2			

Exercício

Valor	Bit de sinal (s)	Mantissa (M)	Expoente (E)
101.11_2 $= (-1)^0 \times 1.0111_2 \times 2^2$	0	1.0111_2 $= 1 + 0.25 + 0.125 + 0.0625$ $= 1.4375$	
-0.001_2			
11.0_2			

Exercício

Valor	Bit de sinal (s)	Mantissa (M)	Expoente (E)
101.11_2 $= (-1)^0 \times 1.0111_2 \times 2^2$	0	1.0111_2 $= 1 + 0.25 + 0.125 + 0.0625$ $= 1.4375$	2
-0.001_2			
11.0_2			

Exercício

Valor	Bit de sinal (s)	Mantissa (M)	Expoente (E)
101.11_2 $= (-1)^0 \times \mathbf{1.0111_2} \times 2^2$	0	1.0111_2 $= 1 + 0.25 + 0.125 + 0.0625$ $= 1.4375$	2
-0.001_2 $= (-1)^1 \times \mathbf{1.0_2} \times 2^{-3}$	1	$1.0_2 = 1$	3
11.0_2 $= (-1)^0 \times 1.1_2 \times 2^1$	0	$1.1_2 = 1.5$	1

Representando números reais no computador

- Vamos adotar essa ideia de representação em notação científica binária para armazenar números reais no computador!
    - 1 bit para guardar o sinal
    - Seguido de alguns bits para guardar a mantissa
    - E mais alguns bits para guardar o expoente

Padrão IEEE 754

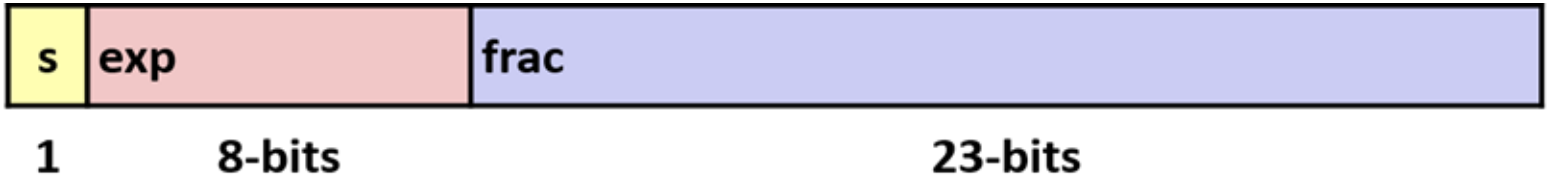
- No início era a baderna...
  - Cada fabricante de CPU definia seu próprio formato de número em ponto flutuante

- Padrão IEEE 754
  - Surgiu em 1985 para uniformizar operações em **ponto-flutuante**
  - Adotado por todas as CPUs mais importantes do mercado

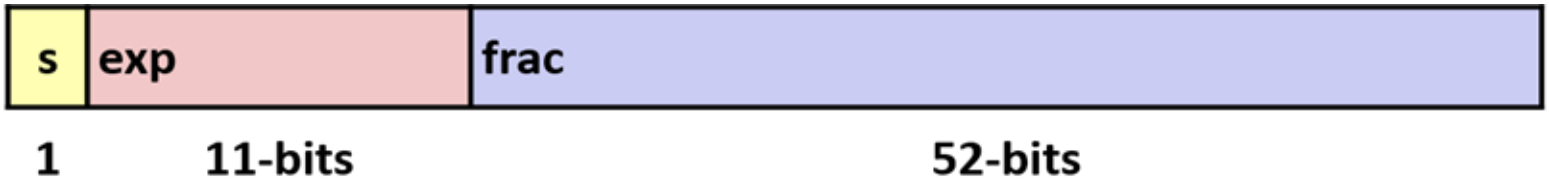
- Criado principalmente por razões matemáticas
  - Arredondamento, overflow, underflow tem propriedades bem bacanas
  - Mas é difícil de implementar eficientemente em hardware!
    - FPUs

Precisão

- Precisão simples (single precision): 32 bits `float`

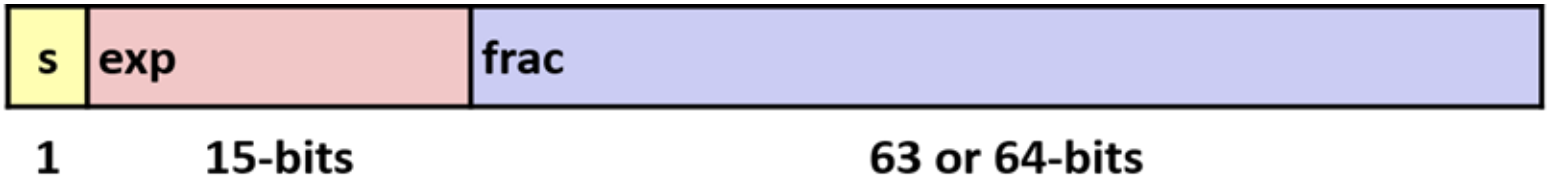


- Precisão dupla (double precision): 64 bits `double`



- Precisão estendida (extended precision): 80 bits (apenas Intel)

`long double`



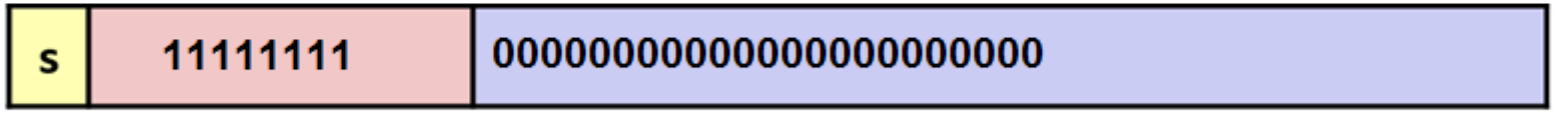
Insper

Representando valores float

Normalizado

Desnormalizado

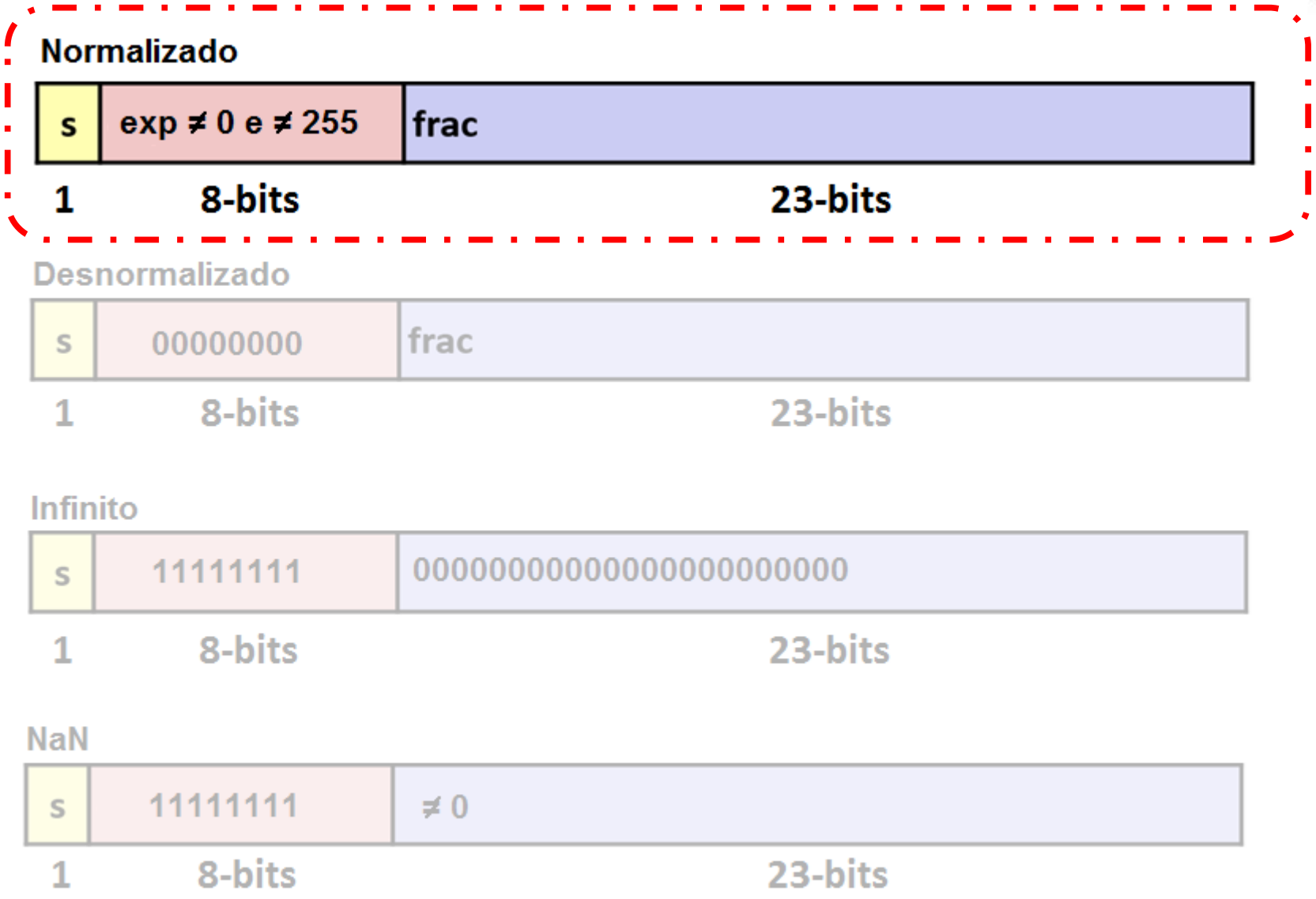
Infinito



NaN

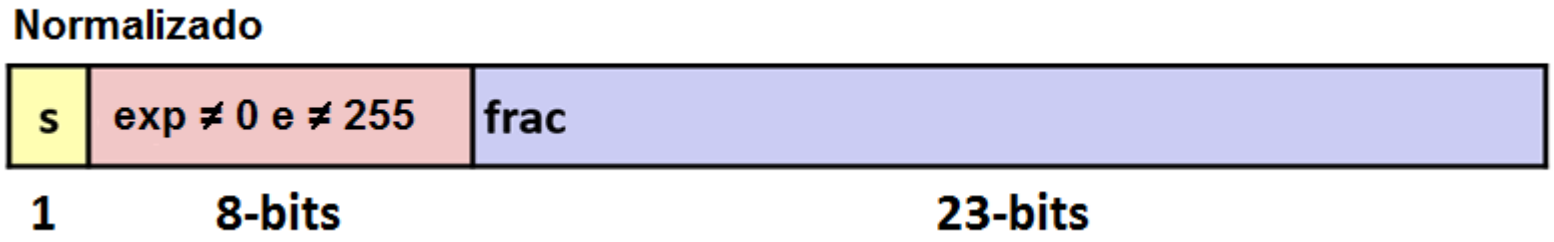
Insper

Valores normalizados



Insper

Valores normalizados



onde:

- s é o bit de sinal
- M = 1.frac
  - Assuma que já tem um bit 1 implícito lá e que frac representa os bits da parte fracionária
- E = exp – Bias
  - Tomando exp como inteiro sem sinal, subtraia o valor do bias. Tipo float: bias = 127

Aplicável quando 0 < exp < 255

- Ou seja, quando -126 ≤ E ≤ 127

Exemplo

Determine os bits de uma variável do tipo float que armazene o número matemático 101.11_2

Resposta:

Passo 1: Lembrando que $101.11_2 = (-1)^0 \times \mathbf{1.0111_2} \times 2^2$

temos $s = 0$, $M = 1.0111_2$ e $E = 2$

Passo 2: Como E >= -126, vamos usar float normalizado.

Passo 3:

23 bits

- De $M = 1.0111_2$ temos frac = 0111 0000 0000 0000 0000 000
- De E = 2 temos exp = E + Bias = 129 = 1000 0001

Resultado final: 0100 0000 1011 1000 0000 0000 0000 0000

0x 4 0 b 8 0 0 0 0

Exercício

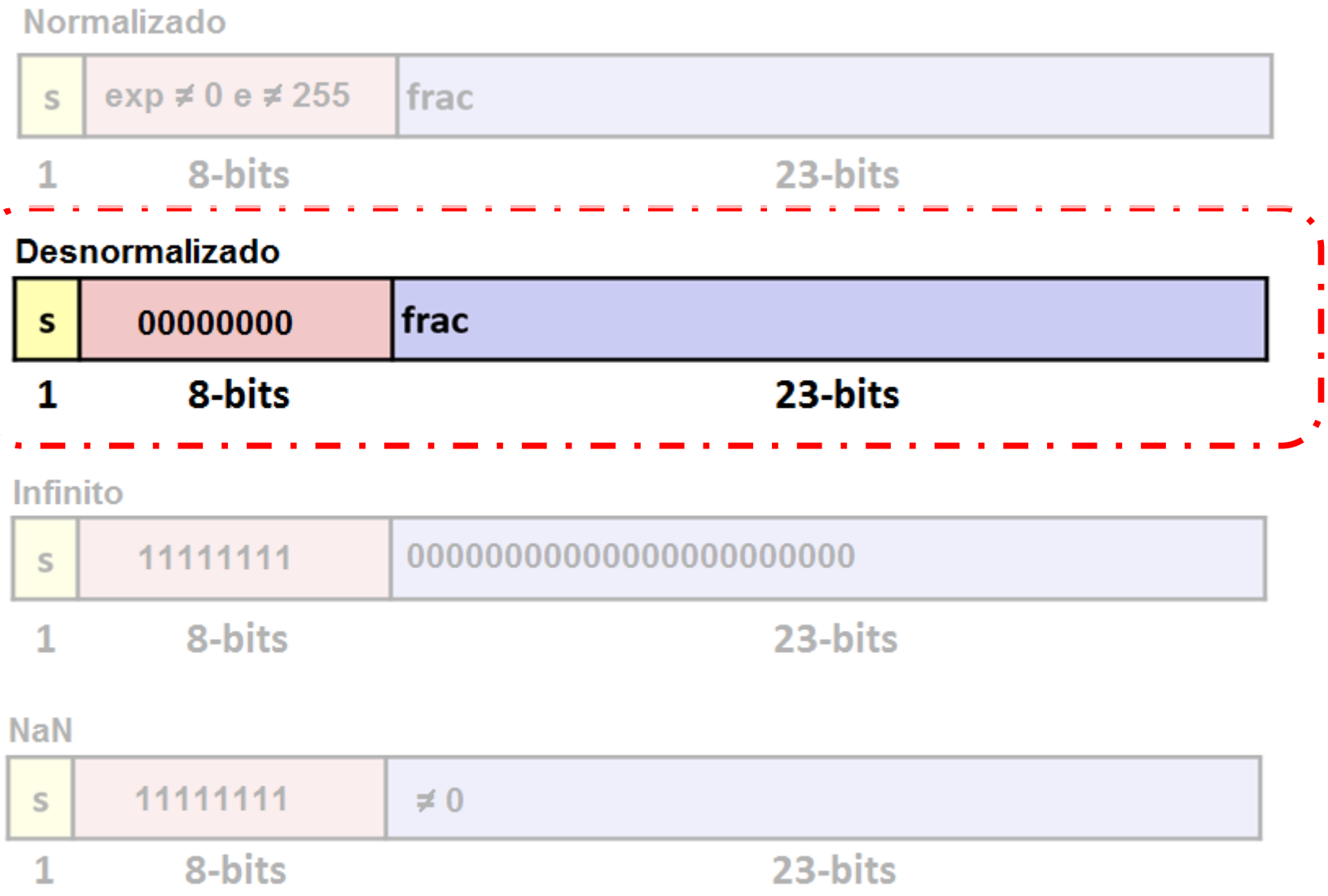
Determine o bit de sinal, o valor de frac e o expoente float (pode ser em decimal) dos seguintes números reais representados em base 2:

$$101.11_2$$

$$-0.001_2$$

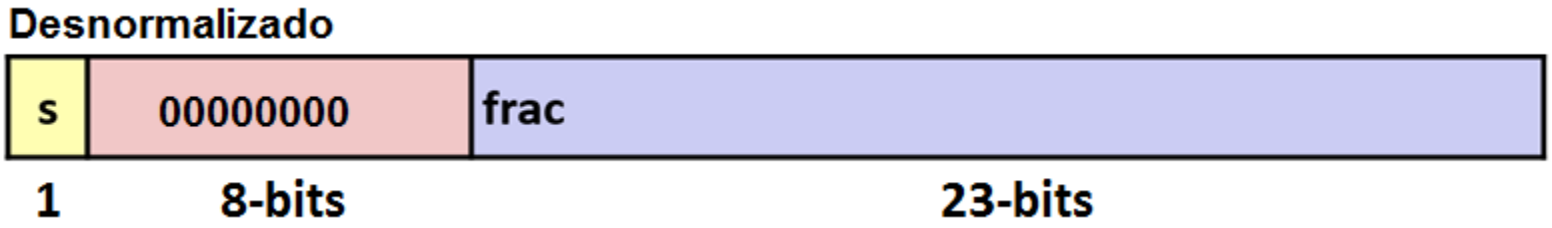
$$11.0_2$$

Valores desnormalizados



Insper

Valores desnormalizados



$$V = (-1)^s \times M \times 2^E$$

onde:

- s é o bit de sinal
- M = 0.frac
  - Assuma que o bit mais alto agora é zero. Novamente, frac representa os bits da parte fracionária
- E = 1 – Bias = -126

São os números de magnitude menor que 2^{-126}.

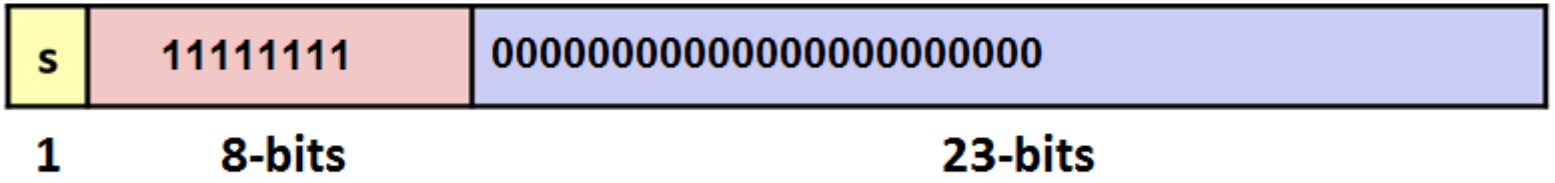
Inclui o valor matemático zero (e também o bizarro menos zero!)

Valores especiais

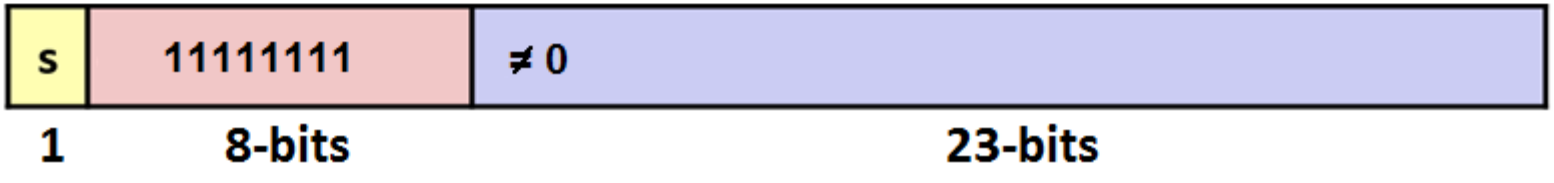
Normalizado

Desnormalizado

Infinito



NaN

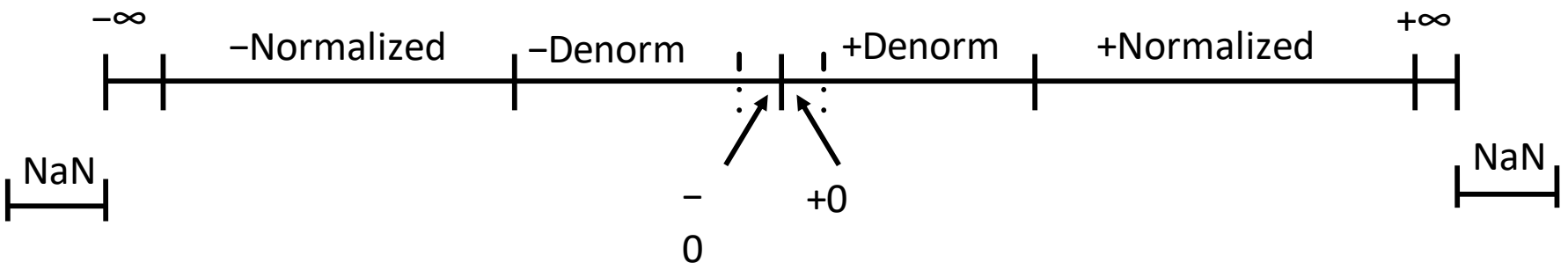


Insper

Valores especiais

- Infinito
    - Tanto positivo como negativo
    - Aparece em operações que resultam em overflow
- NaN
    - “Not-a-Number”
    - Aparece em operações que não resultam em número real

Visualizando a reta numérica



Insper

www.insper.edu.br