Arquitetura de Computadores

PROF. DR. ISAAC

Sistemas de numeração.

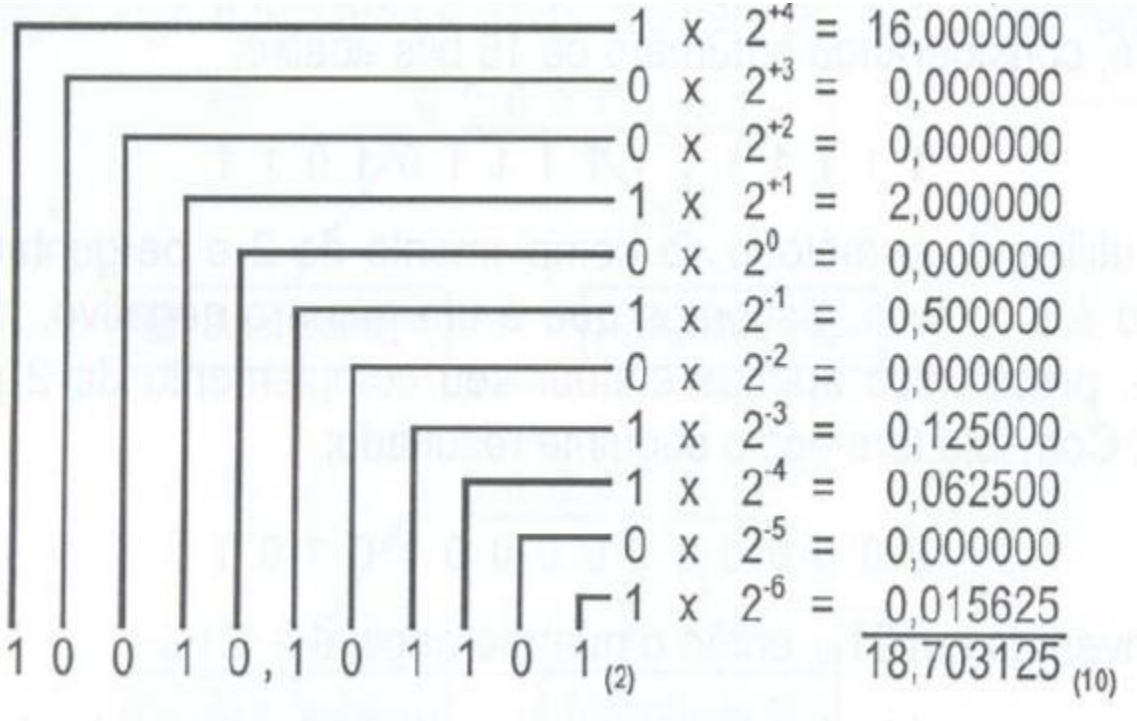
Números Reais

A representação de números reais em binário é uma extensão natural da técnica para inteiros. O peso do bit a direita da vírgula é 2^{-1}, o peso do próximo bit é 2^{-2}, e assim sucessivamente.

Números Reais

Exemplo:

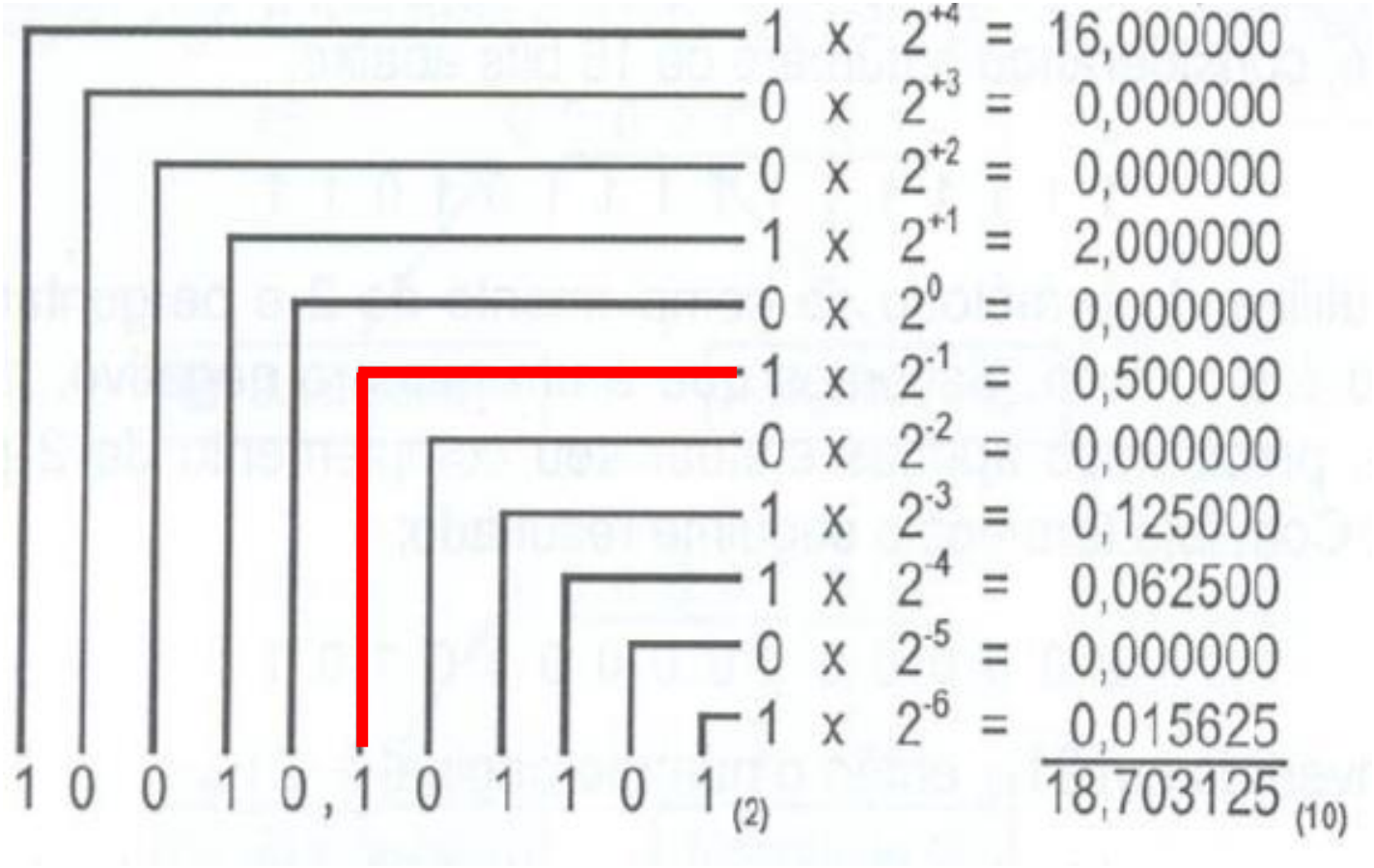
A conversão do número real binário **10010,101101** em decimal é mostrada abaixo:



Números Reais

Exemplo:

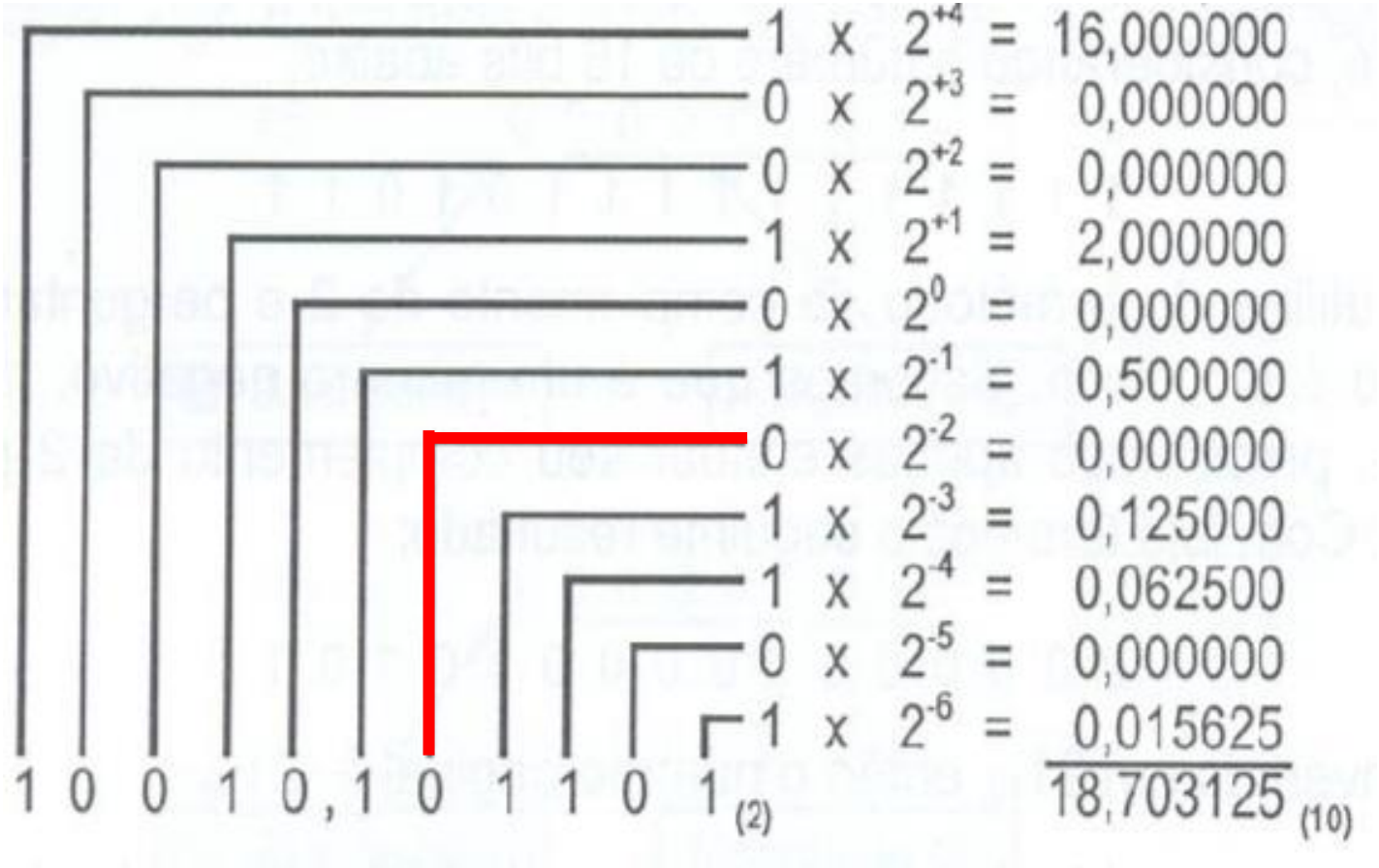
A conversão do número real binário **10010,101101** em decimal é mostrada abaixo:



Números Reais

Exemplo:

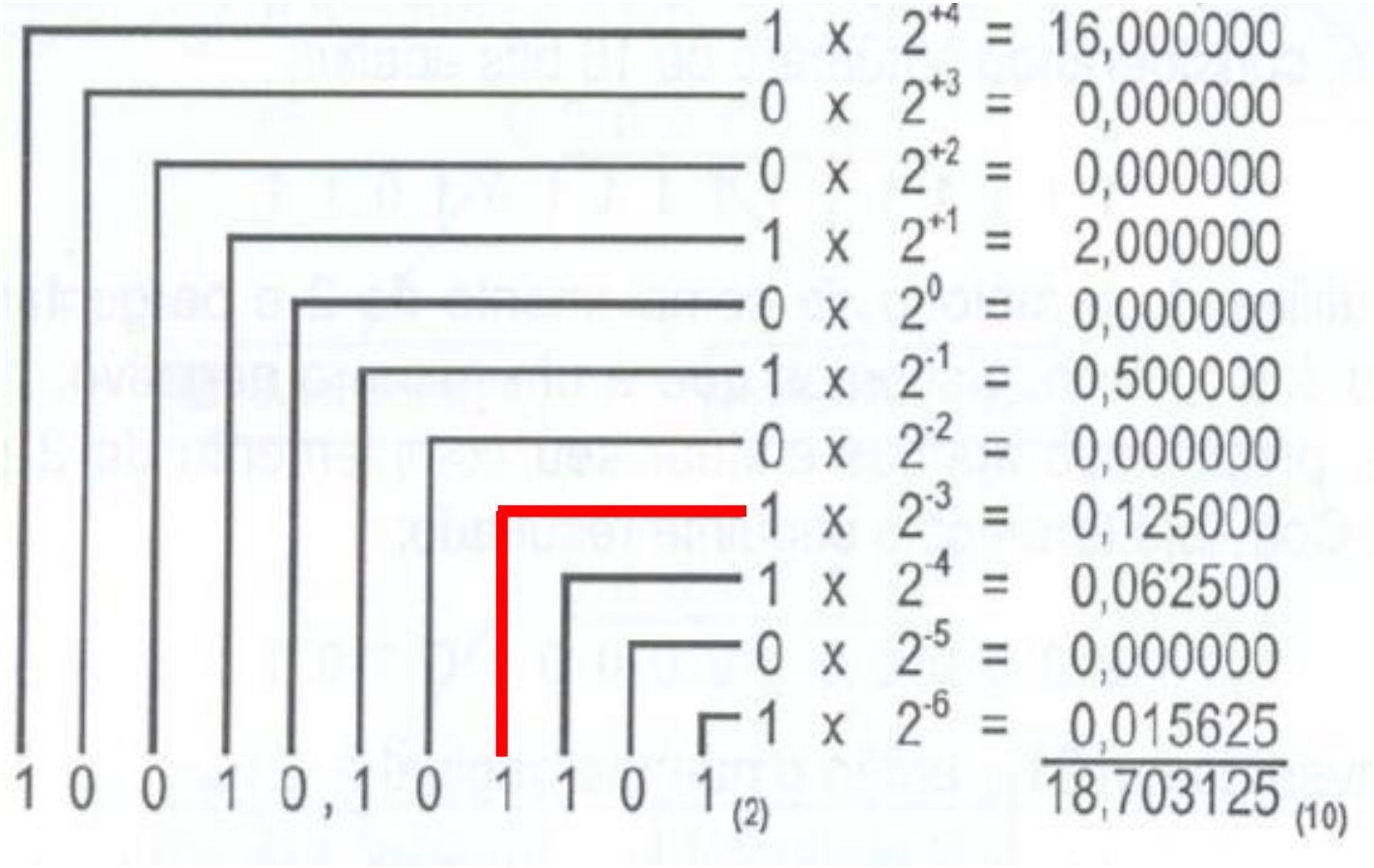
A conversão do número real binário **10010,101101** em decimal é mostrada abaixo:



Números Reais

Exemplo:

A conversão do número real binário **10010,101101** em decimal é mostrada abaixo:



Números Reais

Exemplo:

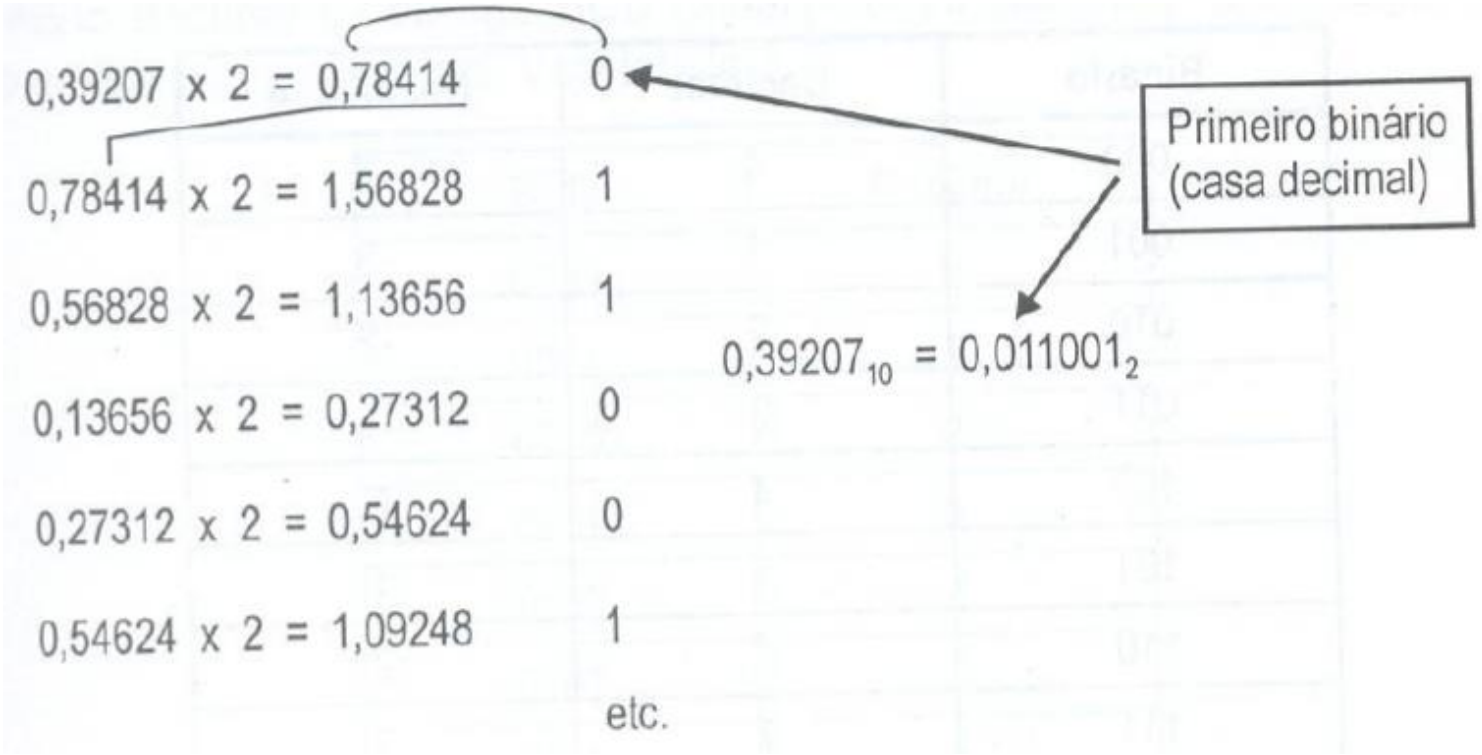
Para a conversão inversa, basta ficar multiplicando continuamente por 2 e extraindo a porção inteira do resultado.

O processo é repetido até que o resultado da multiplicação seja zero, ou até que se atinja a precisão desejada.

Números Reais

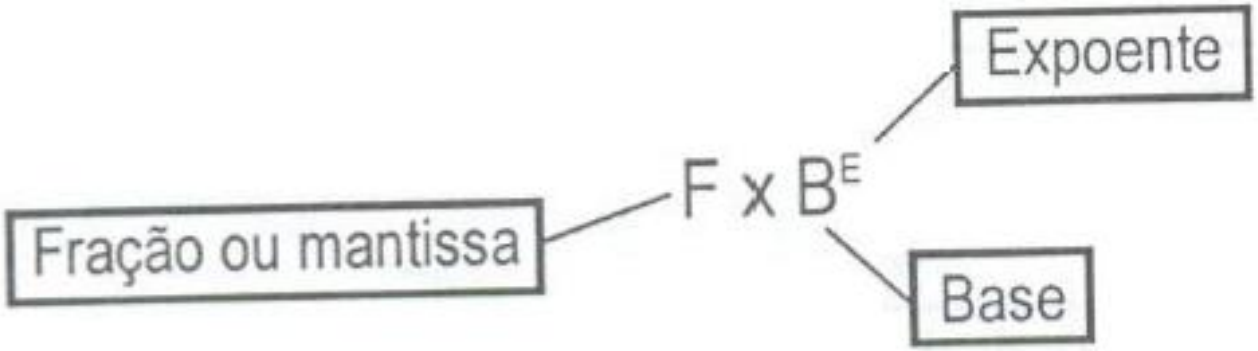
Exemplo:

Conversão de **0,39207** em binário com seis casas decimais de precisão é mostrado abaixo:



Números com Pontos Flutuantes

A representação em ponto flutuante é muito usada para descrever números reais. A notação ponto flutuante consiste em expressar um número utilizado um expoente, uma base e a fração (também chamada de mantissa).



Números com Ponto Flutuante

Exemplo:

O número decimal **2,40625** é mostrado neste exemplo com 12 bits.

$$= + 0{,}01001101 \times 2^{011}$$
$$= + 0{,}01001101 \times 2^{3}$$
$$= + 0010{,}01101$$
$$= + 2 + 0{,}25 + 0{,}125 + 0{,}03125$$
$$= + 2{,}40625$$

Ponto Flutuante no padrão IEEE 754

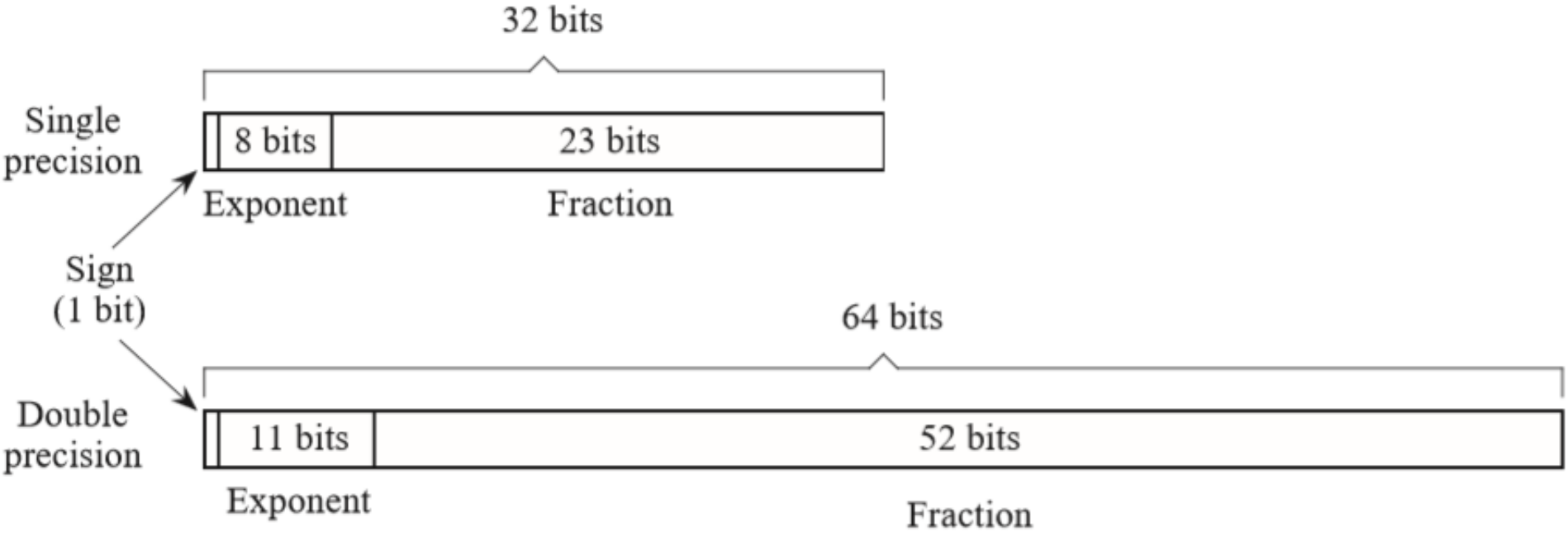
O padrão IEEE 754 foi adotado em 2000, esse padrão define regras de normalização para representações de números binários com ponto flutuante.

Antes disso, cada fabricante de computadores e outros dispositivos, possuía um formato de representação diferente.

Existem dois formatos:

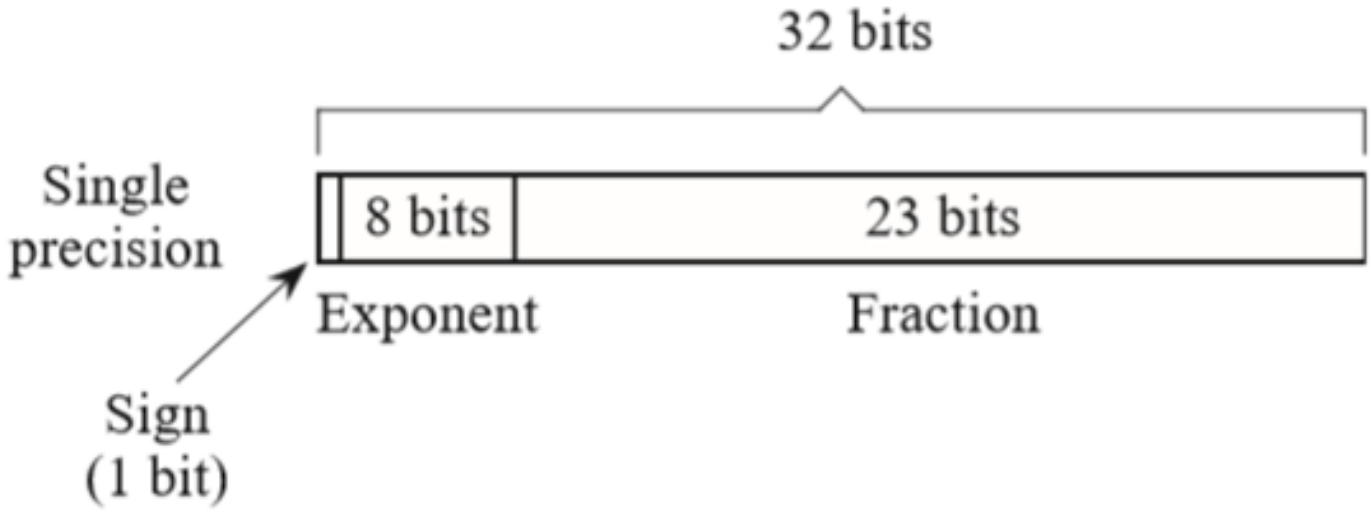
- Precisão Simples - **32 bits**
- Dupla precisão - **64 bits**

Pontos Flutuantes padrão IEEE 754



Pontos Flutuantes padrão IEEE 754 de 32 bits

- O primeiro bit Indica se o numero é positivo ou negativo.
- O 8 bits do expoente é representado com excesso de 127.
- Os outros 23 bits representam a fração.



Pontos Flutuantes padrão IEEE 754 de 32 bits

Exemplo:

Converter o número **74** para padrão IEEE 754 de 32 bits.

Pontos Flutuantes padrão IEEE 754 de 32 bits

Exemplo:

Converter o número **74** para padrão IEEE 754 de 32 bits.

$(74)_{10} = (1001010)_2$ 1001010 = **1.001010×2^6**

$(6)_{10}$ em Excesso de 127 = 127 + 6 = 133

$(133)_{10} = (10000101)_2$

Portanto:

0 no bit de Sinal 10000101 no expoente e 001010 na fração.

0 10000101 00101000000000000000

Pontos Flutuantes padrão IEEE 754 de 32 bits

Exemplo:

Converter o número **74** para padrão IEEE 754 de 32 bits.

Portanto:

0 no bit de Sinal 10000101 no expoente e 001010 na fração.

0100 0010 1001 0100 0000 0000 0000 0000

```
1  #include <iostream>
2
3  int main() {
4    float valor = 74;
5
6    printf("Valor: 0x%x\n", *reinterpret_cast<unsigned
   int *>(&valor));
7
8  }
```

```
Valor: 0x42940000
```

Pontos Flutuantes padrão IEEE 754 de 32 bits

Exemplo: 2.40625

Converter o número **2.40625** para padrão IEEE 754 de 32 bits.

Pontos Flutuantes padrão IEEE 754 de 32 bits

Exemplo: 2.40625

Converter o número **2.40625** para padrão IEEE 754 de 32 bits.

$= + 0010,01101$

$= + 2 + 0,25 + 0,125 + 0,03125$

$= + 2,40625$

$10.01101 \quad = \quad \mathbf{1.001101 \times 2^1}$

Expoente $(1)_{10}$ em Excesso de 127 $\quad = \quad 127 + 1 \; = 128$

$(128)_{10} = \; (10000000)_2$

Pontos Flutuantes padrão IEEE 754 de 32 bits

Exemplo: 2.40625

Converter o número **2.40625** para padrão IEEE 754 de 32 bits.

10.01101 = **1.001101×2^1**

Expoente em Excesso de 127 = (10000000)

Portanto:

Bit de Sinal: **0**

Expoente: **10000000**

Mantissa: **001101**

0 10000000 00110100000000000000000

Pontos Flutuantes padrão IEEE 754 de 32 bits

Exemplo: 2.40625

Converter o número **2.40625** para padrão IEEE 754 de 32 bits.

Portanto:

Bit de Sinal: **0** **E**xpoente: **10000000** Mantissa: **001101**

0100 0000 0001 1010 0000 0000 0000 0000

```
1  #include <iostream>
2
3  int main() {
4    float valor = 2.40625;
5
6    printf("Valor: 0x%x\n", *reinterpret_cast<unsigned
   int *>(&valor));
7
8  }
```

```
Valor: 0x401a0000
```

Pontos Flutuantes padrão IEEE 754 de 32bits

	Value	Bit Pattern		
		Sign	Exponent	Fraction
(a)	$+1.101 \times 2^{5}$	0	1000 0100	101 0000 0000 0000 0000 0000
(b)	-1.01011×2^{-126}	1	0000 0001	010 1100 0000 0000 0000 0000
(c)	$+1.0 \times 2^{127}$	0	1111 1110	000 0000 0000 0000 0000 0000
(d)	+0	0	0000 0000	000 0000 0000 0000 0000 0000
(e)	–0	1	0000 0000	000 0000 0000 0000 0000 0000
(f)	$+\infty$	0	1111 1111	000 0000 0000 0000 0000 0000
(g)	$+2^{-128}$	0	0000 0000	010 0000 0000 0000 0000 0000
(h)	+NaN	0	1111 1111	011 0111 0000 0000 0000 0000

Pontos Flutuantes padrão IEEE 754 de 32bits

(f)	$+\infty$	0	1111 1111	000 0000 0000 0000 0000 0000

```cpp
#include <iostream>

int main() {
  float valor = 2e39; //Infinito, passou limite de float

  printf("Valor: 0x%x\n", *reinterpret_cast<unsigned int *>(&valor));
}
```

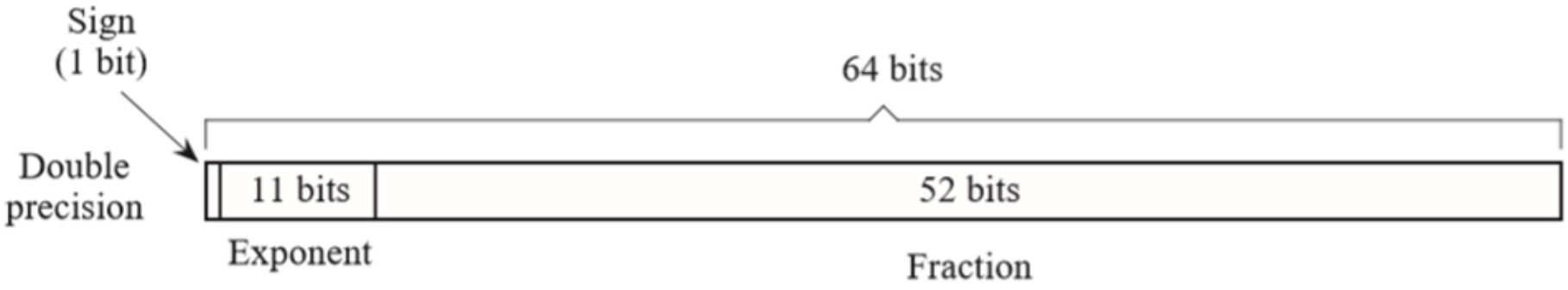
```
Valor: 0x7f800000
```

```cpp
#include <iostream>

int main() {
  float valor = 2e38; //Número abaixo do limite de float

  printf("Valor: 0x%x\n", *reinterpret_cast<unsigned int *>(&valor));
}
```

```
Valor: 0x7f167699
```

Pontos Flutuantes padrão IEEE 754 de 64 bits

- O primeiro bit Indica se o número é positivo ou negativo.
- O 11 bits do expoente é representado com excesso de 1023.
- Os outros 52 bits representam a fração.



Pontos Flutuantes padrão IEEE 754 de 64 bits

Exemplo: 2.40625

Converter o número **2.40625** para padrão IEEE 754 de 64 bits.

Pontos Flutuantes padrão IEEE 754 de 64 bits

Exemplo: 2.40625

Converter o número **2.40625** para padrão IEEE 754 de 64 bits.

$= + 0010,01101$

$= + 2 + 0,25 + 0,125 + 0,03125$

$= + 2,40625$

$10.01101 \quad = \quad \mathbf{1.001101 \times 2^1}$

Expoente $(1)_{10}$ em Excesso de 1023 $\quad = \quad 1023 + 1 \; = 1024$

$(1024)_{10} = \; (10000000000)_2$

Pontos Flutuantes padrão IEEE 754 de 64 bits

Exemplo: 2.40625

Converter o número **2.40625** para padrão IEEE 754 de 64 bits.

10.01101 = **1.001101×2^1**

Expoente em Excesso de 1023 = (10000000000)

Portanto:

Bit de Sinal: **0**

Expoente: **10000000000**

Mantissa: **001101**

0 10000000000
0011010000000000000000000000000000000000000000000000

Pontos Flutuantes padrão IEEE 754 de 64 bits

Exemplo: 2.40625

Converter o número **2.40625** para padrão IEEE 754 de 64 bits.

Portanto:

Bit de Sinal: **0** Expoente: **10000000000** Mantissa: **001101**

0100 0000 0000 0011 0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

```
1  #include <iostream>
2
3  int main() {
4    double valor = 2.40625;
5
6    printf("Valor: 0x%lx\n", *reinterpret_cast<unsigned
   long *>(&valor));
7
8  }
```

```
Valor: 0x4003400000000000
```

Exercícios

Exercícios.

Exercício 1:

a) Converter o número **2.6875** para padrão IEEE 754 de 32 bits.
b) Converter o número **127** para padrão IEEE 754 de 32 bits.
c) Converter o número **7.375** para padrão IEEE 754 de 32 bits.
d) Converter o número **2.6875** para padrão IEEE 754 de 64 bits.
e) Converter o número **127** para padrão IEEE 754 de 64 bits.
f) Converter o número **7.375** para padrão IEEE 754 de 64 bits.

Resposta do exercício 1.

Exercício 1:

a) 0 10000000 01011000000000000000000

b) 0 10000101 11111100000000000000000

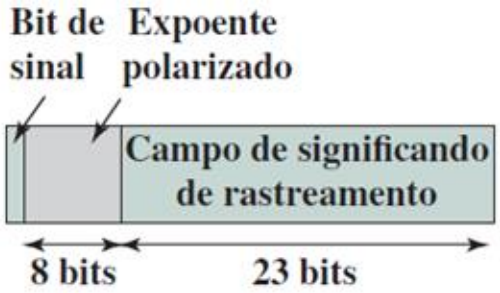
c) 0 10000001 11011000000000000000000

d) 0 10000000000 01011000000000000000000000000000000000000000000000000

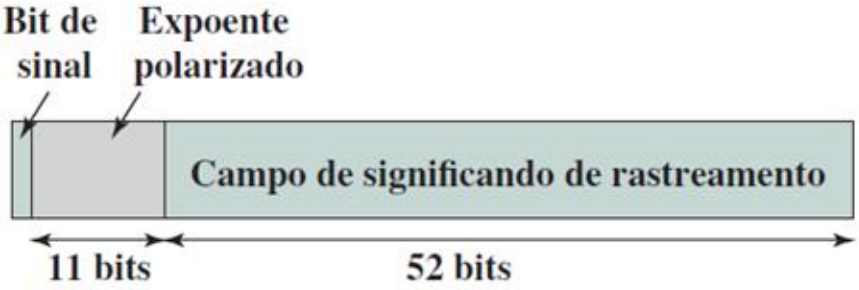
e) 0 10000000101 11111100000000000000000000000000000000000000000000000

f) 0 10000000001 11011000000000000000000000000000000000000000000000000

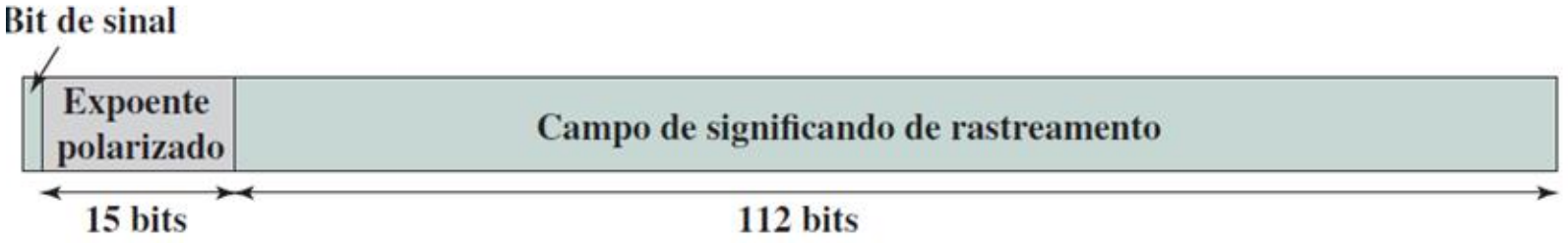
Pontos Flutuantes padrão IEEE 754



(a) Formato binário32



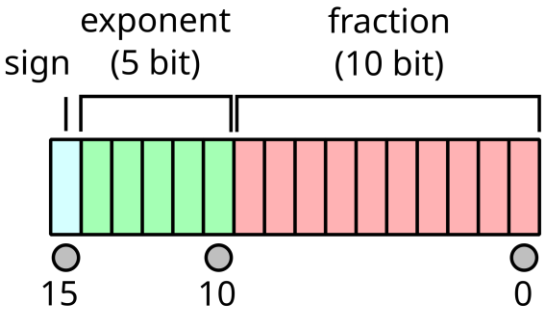
(b) Formato binário64



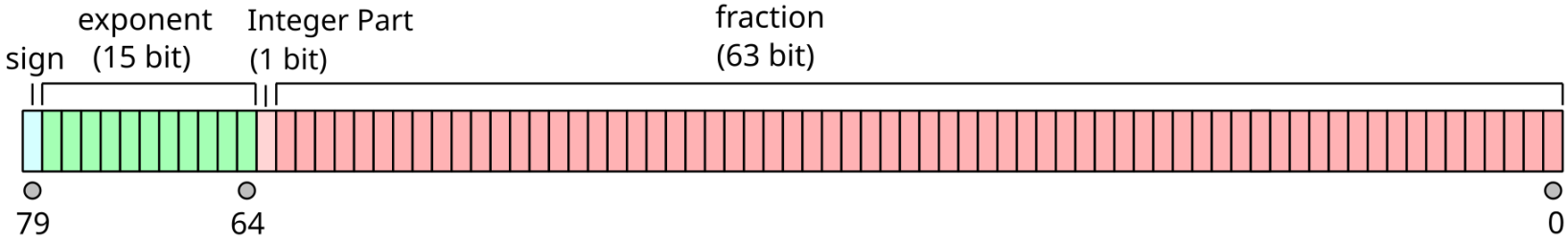
(c) Formato binário128

Pontos Flutuantes padrão IEEE 754

16bits - Meia precisão



80bits – Precisão estendida



Bibliografia

Murdocca, Miles, and Vincent Heuring. Computer Architecture and Organization. Vol. 271. New York, NY, USA: John Wiley & Sons, 2007.

ZELENOVSKY, R.; MENDONÇA, A. Microcontroladores Programação e Projeto com a Família 8051. MZ Editora, RJ, 2005.

Gimenez, Salvador P. Microcontroladores 8051 - Teoria e Prática, Editora Érica, 2010.