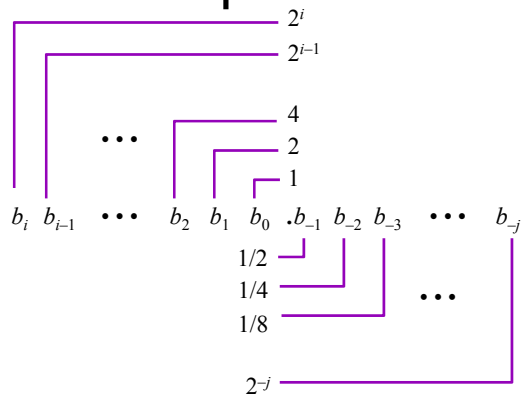
Representação em Ponto Flutuante

Números Fracionários em repr. binária



- Bits à direita do "ponto binário" representam potências de 2 fracionárias
- O número racional é representado como: $\sum_{k=-j}^{i} b_k \cdot 2^k$

Exemplos

- Número Representação

5 + 3/4	101.11_2
2 + 7/8	10.111_2
63/64	0.111111_2

- Observações
  - Divisão por 2: shift right
  - Multiplicação por 2: shift left
  - Números da forma $0.111111\ldots_2$ estão próximos de 1.0
    - 1/2 + 1/4 + 1/8 + … + $1/2^i$ + … $\rightarrow$ 1.0
  - Esta representação tem limitações

Limitação

- É possível representar exatamente apenas números racionais que tenham parte fracionária da forma: $\sum_{k=-j}^{0} b_k \cdot 2^k$
- Exemplos

Número	Representação
1/8 (= 0.125)	0.001_2
1/16 (= 0.0625)	0.0011_2
5.625	101.101_2

- Outros números possuem sequências de bits repetidas indefinidamente → a representação binária não é precisa
- Exemplos:

Número	Representação
1/3	$0.0101010101[01]\ldots_2$
1/5	$0.001100110011[0011]\ldots_2$
1/10	$0.0001100110011[0011]\ldots_2$

- Um problema desta representação: números muito grandes ou muito pequenos necessitariam de uma sequência muito longa de bits…

Representação IEEE

- As mais diversas representações de ponto flutuante já foram propostas, mas …
- O padrão IEEE 754 atualmente é o mais utilizado:
  - Criado em 1985 como padrão para representação e aritmética em ponto flutuante
  - Implementado na grande maioria das CPUs

- Define três precisões:
  - Single precision (`float`) 32 bits (precisão 24 bits)
  - Double precision (`double`) 64 bits (precisão 53 bits)
  - Double extended precision 80 bits (precisão 63 bits)

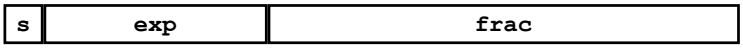
  Obs: esta última somente em arquiteturas Intel-like

Padrão IEEE 754

- Forma numérica

$$(-1)^s \; M \; 2^E$$

  - Bit de sinal *s* determina se número é negativo ou positivo
  - Mantissa *M* é um valor fracionário no intervalo [1.0,2.0), na representação normalizada.
  - Expoente *E*
- Codificação



  - bit mais significativo é s
  - Campo `exp` codifica *E*
  - Campo `frac` codifica *M*

IEEE 754: Precisões de Ponto Flutuante

- Tamanhos
  - **`float`**: `exp` = 8 bits, `frac` = 23 bits, `s` = 1 bit
    - Total: 32 bits
    - Faixa de valores: 2^{-126} até 2^{127}
  - **`double`**: `exp` =11 bits, `frac` = 52 bits, `s` = 1 bit
    - Total: 64 bits
    - Faixa de valores: 2^{-1022} até 2^{1023}
  - Precisão estendida: `exp` =15 bits, `frac` = 63 bits,`s` = 1 bit
    - Total: 80 bits
    - Faixa de valores: 2^{-16382} até 2^{16383}
    - 1 bit é desperdiçado

Padrão IEEE 754

Define-se duas representações:

- Normalizada (mais utilizada)

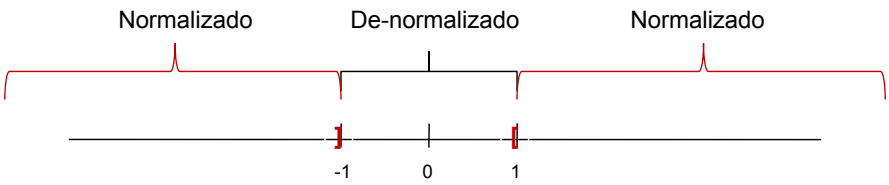
- De-normalizada: para representar o 0 e números cujo módulo é [0, 1)

  Neste caso, `exp` = 0

  E = 1 – Bias

  M = `frac`

Faixa de Valores



- De-normalizado: f $\in$ (-1,1)
  `exp` = 0, M = `frac`
- Normalizado: f $\in$ $(-\infty,1] \cup [1, +\infty)$
  `exp` $\neq$ 0 (`00..01` – `11..10`), M= 1.0 + `frac`

Mantissa na rep. normalizada

- O padrão estabelece uma representação normalizada, com valores 1 <= *M* < 2. Isto é possível, escolhendo-se o *Exp* adequadamente.
- Como a mantissa é sempre da forma

  M = 1+ `frac`
- Armazena-se apenas a parte fracionária `frac`. Ou seja, a soma com "1" é sempre implícita.

- Exemplos:

  `frac` = $0...0_2 = 0_{10}$ M = 1.0

  `frac` = $0101...0_2 = 0.25 + 0.0625_{10}$ M = 1,03125

O Exponte na rep. normalizada

- O Expoente E, pode ser positivo ou negativo, mas **não** é armazenado em complemento a dois
- Usa-se a "representação excesso" (ou bias)

$$E = Exp - Bias$$

  - *Exp* : valor sem sinal representado por `exp`
  - *Bias* : valor de excesso
    - 127 em float (*Exp*: 1…254, *E*: -126…127)
    - 1023 em double (*Exp*: 1…2046, *E*: -1022…1023)
    - $2^{n-1} - 1$ (no caso geral), onde *n* é o número de bits em `exp`
- Exemplos:

```
exp = 1      E = (1-127) = -126
exp = 127    E = (127-127) = 0
exp = 225    E = (225-127) = 98
```

Exemplo (precisão simples)

- Valor

```
float F = 15213.0;
```

15213_{10} = 11101101101101_2 = 1.1101101101101_2 X 2^{13}

- Mantissa

M = **1.1101101101101_2**

`frac` = $11011011011010000000000_2$

- Expoente

E = 13

exp = E + Bias = 13+ 127 = 140 = 10001100_2

Sinal:	**0**						
Hex:	**4**	**6**	**6**	**D**	**B**	**4**	**0**	**0**
Binário	0100	0110	0110	1101	1011	0100	0000	0000
140:	100	0110	0					
15213:		*1*110	1101	1011	01			

Exemplo (precisão dupla)

- Valor

  `double D = 178.125 = 128+32+16+2 + 0.125;`

  178.125_{10} = 10110010.001_2

  1.78125_{10} = 1.0110010001_2 X 2^7

- Mantissa

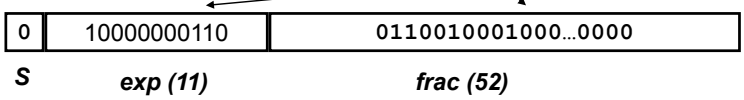
  M **=** **1. 0110010001_2**

  `frac` = $01100100010\ldots0_2$

- Expoente

  E = 7

  exp = E + Bias = 7+ 1023 = 1030 = 10000000110_2



S ***exp (11)*** ***frac (52)***

Valores Especiais

zero	S= 0	exp = 0	M = 0
+ ∞	S=0	exp=111...111	M = 0
- ∞	S=1	exp=111...111	M = 0
NaN(*)	S=0	exp=111...111	M ≠ 0

(*) NaN = Not a Number

A aritmética com esses números

Para qualquer ponto flutuante *a* vale:

- a + ∞ = ∞
- a + -∞ = -∞
- ∞ + -∞ = NaN
- NaN + a = NaN
- $\sqrt{}$ -1 = NaN
- a / ±∞ = 0
- ±∞ * ±∞ = ±∞
- a / 0 = ±∞, se a≠0
- ±0 / ±0 = NaN
- ±∞ / ±∞ = NaN
- ±∞ * 0 = NaN

Mais informações

http://steve.holasch.net/cgindex/Ieeefloat.html

IEEE Computer Society (1985) IEEE Standard for Binary Floating-Point Aritmetic, IEEE Std 754-1985.

Comparing floating point numbers, Bruce Dawson. http://www.cygnus-software.com/papers