Software Básico

Representação e manipulação da informação

Ponto Flutuante

Leard de Oliveira Fernandes
CET 088 hc1s1u0
(classroom.google.com)

Roteiro

- **Números Binários Fracionais**
- Padrão Ponto Flutuante IEEE: Definição
- Exemplos e propriedades
- Arredondamento, Adição, Multiplicação
- Ponto Flutuante no C

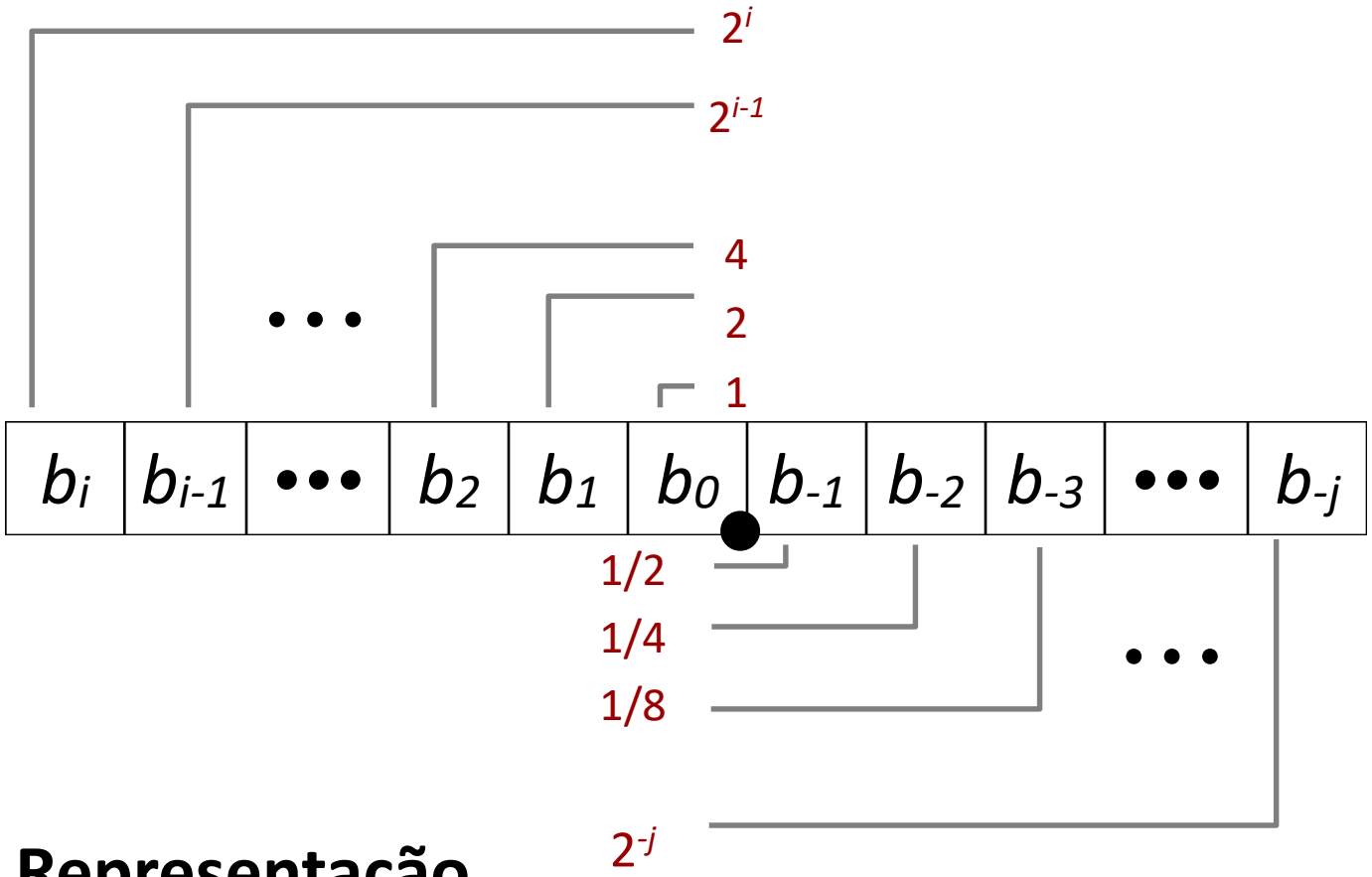
Números Binários Fracionais

- O que é 1011.101_2?

- O que é 19.84_{10}?

$$\sum_{i=-n}^{i=m} d_i \times 10^i$$

Números Binários Fracionais



- **Representação**
  - Os bits à direita do "ponto binário" representam potências de dois fracionais
  - Representam um número racional:

$$\sum_{k=-j}^{i} b_k \times 2^k$$

Números Binários Fracionais: Exemplos

- **Valor** **Representação**
  - 5 3/4 **101.11_2**
  - 2 7/8 **10.111_2**
  - 1 7/16 **1.0111_2**
  - 63/64 **0.111111_2**
- **Observações**
  - Divide por 2 aplicando deslocamento à direita (unsigned)
  - Multiplica por 2 aplicando deslocamento à esquerda
  - Números $0.111111..._2$ estão apenas abaixo de 1
    - $1/2+1/4+1/8+...1/2^i+... \rightarrow 1.0$
    - Utilize a notação 1 - ε

Números Representáveis

- **Limitação #1**
  - Podemos representar apenas números da forma $\mathbf{x/2^k}$
    - Outros números racionais possuem representações com bits de repetição
  - Valor — Representação
    - 1/3 — $\mathbf{0.0101010101[01]\ldots_2}$
    - 1/5 — $\mathbf{0.00110011001[0011]\ldots_2}$
    - 1/10 — $\mathbf{0.000110011001[0011]\ldots_2}$
- **Limitação #2**
  - Apenas uma configuração de ponto binário com t bits
    - Limita o tamanho dos números

Roteiro

- Números Binários Fracionais
- **Padrão Ponto Flutuante IEEE: Definição**
- Exemplos e propriedades
- Arredondamento, Adição, Multiplicação
- Ponto Flutuante no C

Ponto Flutuante IEEE

- **Padrão IEEE 754**
  - Estabelecido em 1985 para uniformizar a aritmética ponto flutuante;
    - Antes, era ...
  - Suportado pela maioria dos processadores
- **Descrito por conceitos numéricos**
  - Bons padrões para arredondamento, overflow, underflow
  - Difícil de fazer rápido em hardware
    - Analistas numéricos predominaram sobre projetistas de hardware na definição do padrão

Representação Ponto Flutuante

- **Forma Numérica:**

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

  - **Bit de Sinal s** determina se o número é **+** ou **-**
  - **Significando M** normaliza um valor fracional entre [1.0, 2.0)
  - **Expoente E** multiplica o valor por uma potência de 2
- **Codificação**
  - MSB s é o bit de sinal **s**
  - Campo exp codifica **E** (não é igual a E)
  - Campo frac codifica **M** (não é igual a M)

Precisões

- Precisão única (Single Precision): 32 bits

- Precisão dupla (Double Precision): 64 bits

- Precisão extendida (Extended Precision): 80 bits

Valores Normalizados

$$v = (-1)^s M 2^E$$

- **Quando: exp ≠ 000...0 e exp ≠ 111...1**
- **Expoente é codificado como um valor enviesado (Biased): E = exp – Bias**
  - exp : valor unsigned do campo exp
  - Bias: $2^{k-1} - 1$, onde **k** é o número de bits do expoente;
  - Precisão Simples: 127 (exp: 1...254, E: -126...127)
  - Precisão Dupla: 1023 (exp:1...2046, E: -1022...1023)
- **Significando codificado com condução implícita de 1: M** = $1.xxx...x_2$
  - xxx...x: bits do campo frac
  - Mínimo quando frac = 000...0 (M=1.0)
  - Máximo quando frac = 111...1 (M=2.0-ε)
  - Temos um bit extra conduzido de "grátis"

Valores Normalizados: Exemplo

$v = (-1)^s \, M \, 2^E$

$E = Exp - Bias$

- **Valor: float F = 15213.0**
  - 15213_{10} = 11101101101101_2
    = $1.1101101101101_2 \times 2^{13}$
- **Significando**
  M = $1.\underline{1101101101101}_2$
  frac = $\underline{1101101101101}0000000000_2$
- **Expoente**
  - E = 13
  - Bias = 127
  - Exp = 140 = 10001100_2
- **Resultado**

0	10001100	11011011011010000000000
s	**exp**	**frac**

Representar 31/3, em ponto flutuante de precisão simples

Valores Não Normalizados (Denormalizados)

$v = (-1)^s M 2^E$

$E = 1 - Bias$

- **Condição:** exp = 000...0

- **Valor do expoente:** ***E = 1 – Bias*** **(Ao invés de** ***E = 0 – Bias*****)**
- **Significando codificado com condução implícita de 0:** $M = 0.xxx...x_2$
  - xxx...x: bits de frac

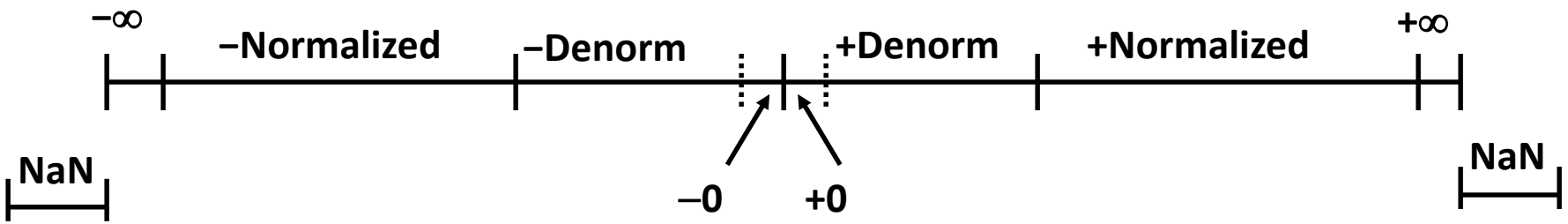
- **Casos:**
  - exp = 000...0, frac = 000...0
    - Representa o valor 0
    - Note distintos valores: +0 e -0 (Por quê?)
  - exp = 000...0, frac ≠ 000...0
    - Números próximos de 0
    - Equiespaçados

Valores Especiais

- **Condição:** exp = 111...1

- **Caso:** exp = 111...1, frac = 000...0
  - Representa valor ∞ (Infinito)
  - Operação com overflow
  - Seja positive ou negativo
  - e.g: 1.0/0.0 = -1.0/-0.0 = + ∞; 1.0/-0.0 = - ∞
- **Caso:** exp = 111...1, frac ≠ 000...0
  - Not-a-Number (NaN)
  - Representa casos em que valores não numéricos são determinados
    - E.g: sqrt(-1), ∞- ∞, ∞*0

Visualizando a codificação ponto flutuante



Roteiro

- Números Binários Fracionais
- Padrão Ponto Flutuante IEEE: definição
- **Exemplos e propriedades**
- Arredondamento, Adição, Multiplicação
- Ponto Flutuante no C

Exemplo de ponto flutuante pequeno

- **Representação ponto flutuante de 8-bits**
  - O sinal é o MSB
  - exp possui 4 bits e bias de 7
  - Os últimos três bits são frac

- **Mesmo formato geral do padrão IEEE**
  - Normalizado, denormalizado
  - Representação de 0, NaN, Infinito

Intervalo Dinâmico (Positivo)

$v = (-1)^s M 2^E$

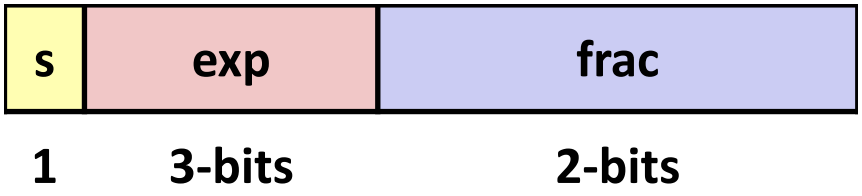
n: E = Exp – Bias

d: E = 1 – Bias

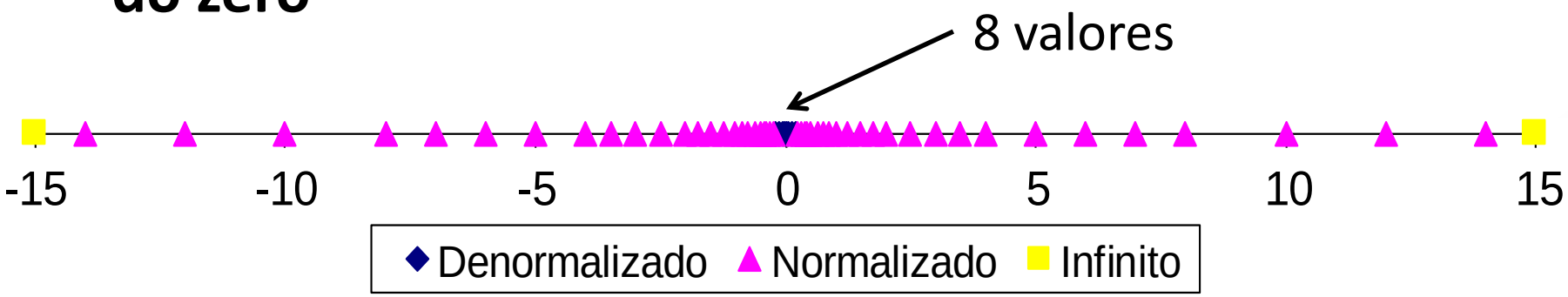
	s	exp	frac	E	Valor	
Números Denormalizados	0	0000	000	-6	0	
	0	0000	001	-6	1/8*1/64 = 1/512	Perto de Zero
	0	0000	010	-6	2/8*1/64 = 2/512	
	…					
	0	0000	110	-6	6/8*1/64 = 6/512	
	0	0000	111	-6	7/8*1/64 = 7/512	Maior Denormalizado
Números Normalizados	0	0001	000	-6	8/8*1/64 = 8/512	Menor Normalizado
	0	0001	001	-6	9/8*1/64 = 9/512	
	…					
	0	0110	110	-1	14/8*1/2 = 14/16	
	0	0110	111	-1	15/8*1/2 = 15/16	Próximo abaixo de 1
	0	0111	000	0	8/8*1 = 1	
	0	0111	001	0	9/8*1 = 9/8	Próximo acima de 1
	0	0111	010	0	10/8*1 = 10/8	
	…					
	0	1110	110	7	14/8*128 = 224	
	0	1110	111	7	15/8*128 = 240	Maior Normalizado
	0	1111	000	n/a	inf	

Distribuição de Valores

- **Formato com 6-bits**
  - exp= 3 bits
  - frac = 2 bits
  - Bias é $2^{3-1} - 1 = 3$

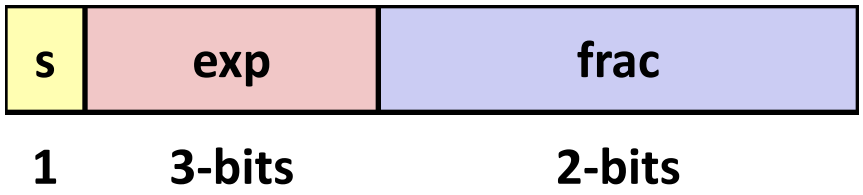


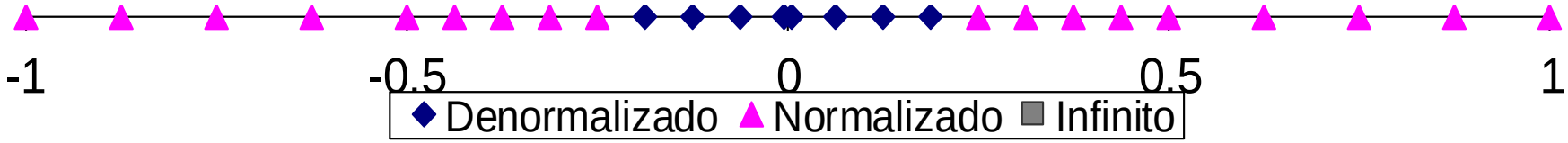
- **Observe como a distribuição fica densa em torno do zero**



Distribuição dos valores (Close-Up)

- **Formato com 6-bits**
  - exp= 3 bits
  - frac = 2 bits
  - Bias é 3





Propriedades Especiais da codificação da IEEE

- **Zero PF é o mesmo que um zero Int**
  - Todos os bits = 0
- **Pode (em alguns casos) utilizar a comparação de inteiro unsigned**
  - Deve ser comparado primeiro os bits de sinal
  - Deve ser considerado -0 = 0
  - NaNs é problemático
    - Será maior do que qualquer outro valor
    - O que a comparação deveria submeter?
  - Caso contrário, ok
    - Denormalizado vs Normalizado
    - Normalizado vs Infinito

Roteiro

- Números Binários Fracionais
- Padrão Ponto Flutuante IEEE: definição
- Exemplos e propriedades
- **Arredondamento, Adição, Multiplicação**
- Ponto Flutuante no C

Operações em Ponto Flutuante: Ideia Básica

- $x +^f y = Round(x + y)$
- $x \times^f y = Round(x \times y)$

- **Ideia Básica**
  - Primeiro Compute o resultado exato
  - Faça ele caber na precisão desejada
    - Possível overflow de um expoente grande
    - Possível arredondamento para caber em frac

Arredondando

- **Modos de arredondar (Ilustrando com R$)**

	R$1.40	R$1.60	R$1.50	R$2.50	–R$1.50
• Para zero	$1	$1	$1	$2	–$1
• Para baixo (−∞)	$1	$1	$1	$2	–$2
• Para cima (+∞)	$2	$2	$2	$3	–$1
• Par mais próximo (default)	$1	$2	$2	$2	–$2

Analisando o Arredondamento para o Par mais próximo

- **Modo de arredondamento padrão (Round-to-Even)**
  - Todos os outros são estatisticamente enviesados
    - Soma do conjunto de números positivos serão consistentemente sobre/subestimados
- **Aplicando em outras casas decimais / Posições dos Bits**
  - Quando temos caminhos intermediários entre dois possíveis valores
    - Arredonde tal que o bit menos significante seja PAR
  - E.g: arredondar para o centésimo mais próximo

7.8949999	7.89	(Menor que o valor intermediário)
7.8950001	7.90	(Maior que o intermediários)
7.8950000	7.90	(Intermediário—round up)
7.8850000	7.88	(Intermediário—round down)

Arredondando Números Binários

- **Números Binários Fracionais**
  - "Par", quando o número menos significante é zero
  - "Intermediário", quando bits à direita da posição de arredondamento = $100..._2$

- **Exemplos**
  - Arredondar para o 1/4 mais próximo (2 bits à direita do ponto binário)

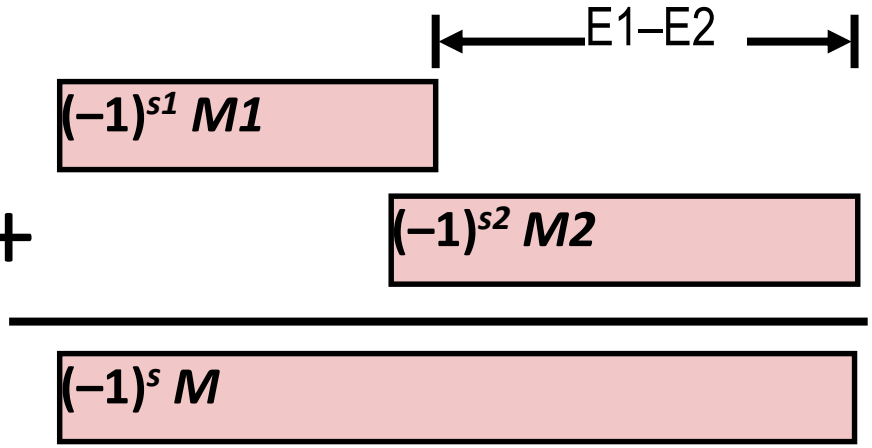
Valor	Binário	Arredondado	Ação	Valor Arredondado
2 3/32	10.00011_2	10.00_2	(<Inter—baixo)	2
2 3/16	10.00110_2	10.01_2	(>Inter—cima)	2 1/4
2 7/8	10.11100_2	11.00_2	(Inter—cima)	3
2 5/8	10.10100_2	10.10_2	(Inter—baixo)	2 1/2

Multiplicação PF

- $(-1)^{s1}\ M1\ \ 2^{E1}\ \ \text{x}\ \ (-1)^{s2}\ M2\ \ 2^{E2}$
- **Resultado Exato:** $(-1)^{s}\ M\ \ 2^{E}$
  - Sinal s: s1^s2
  - Significando M: M1 x M2
  - Expoente E: E1+E2
- **Fixando**
  - Se M ≥ 2, desloque M para direita, incremente E
  - Se E está fora do range, overflow
  - Arredonde M para caber na precisão de frac
- **Implementação**
  - A principal ocupação é apenas multiplicar os significandos

Adição Ponto Flutuante

- $(-1)^{s1}\ M1\ \ 2^{E1}\ \ +\ \ (-1)^{s2}\ M2\ \ 2^{E2}$
  - Assuma E1 > E2
- **Resultado Exato: $(-1)^{s}\ M\ 2^{E}$**
  - Sinal s, significando M:
    - Resultado do alinhamento do sinal e a soma
  - Expoente E: E1
- **Fixando**
  - Se M ≥ 2, Desloque M para direita, incremente E;
  - Se M < 1, desloque M para esquerda k posições, decremente E de k;
  - Overflow se E fora do intervalo;
  - Arredonde M para caber na precisão de frac



Propriedades Matemáticas da soma e Multiplicação PF

- **É cumulativa => A+B = B+A**
- **Não é associativa**
  - (A+B)+C != (A+C)+B
  - (A*B)*C != (A*C)*B
    - Devido o overflow
    - E a natureza inexata do arredondamento
- **Nem todo elemento possui aditivo inverso**
  - Infinito e NaNs
- **Não é distributiva**
  - A*(B+C) != A*B + A*C
    - Devido o overflow
    - E a natureza inexata do arredondamento

Demonstrar em código C, estas propriedades

Roteiro

- Números Binários Fracionais
- Padrão Ponto Flutuante IEEE: definição
- Exemplos e propriedades
- Arredondamento, Adição, Multiplicação
- **Ponto Flutuante no C**

Ponto Flutuante no C

- **C garante dois níveis**
  - Float (Precisão Simples)
  - Double (Precisão Dupla)
- **Conversões/Castings**
  - Casting entre int, float e double muda a representação dos bits
  - Double/float -> int
    - Trunca a parte fracional
    - Arredonda para zero
    - Não definida quando fora do intervalo ou NaN
      - Geralmente vai para CMIN
  - Int -> double
    - Conversão exata
  - Int -> float
    - Irá arredondar conforme o modo de arredondamento

Quebra-Cabeças Ponto Flutuante

- **Para cada uma das expressões em C, avalie:**
  - Verifique se é verdade para todos os valores de argumentos
  - Explique porque não é verdade

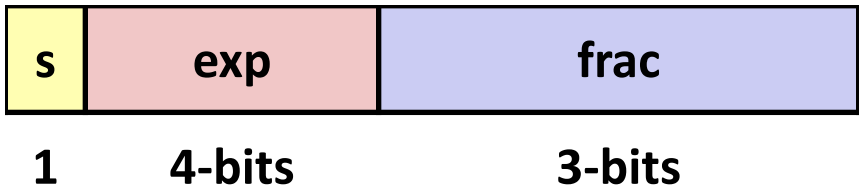
```
int x = …;
float f = …;
double d = …;
```

Nem `d` ou `f` é NaN

- `x == (int)(float) x`
- `x == (int)(double) x`
- `f == (float)(double) f`
- `d == (double)(float) d`
- `f == -(-f);`
- `2/3 == 2/3.0`
- `d < 0.0 ⇒ ((d*2) < 0.0)`
- `d > f ⇒ -f > -d`
- `d * d >= 0.0`
- `(d+f)-d == f`


IN ALTUM

Criando um número PF



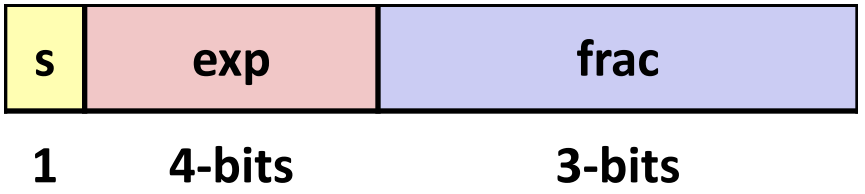
- **Passos**
  - Normalize para ter o dominante de 1
  - Arredonde para caber dentro de frac
  - Pós-normalize para lidar com os efeitos do arredondamento

- **Estudo de caso**
  - Converta números unsigned para o formato de ponto flutuantes pequenos

Números de Exemplos

128	10000000
15	00001101
33	00010001
35	00010011
138	10001010
63	00111111

Normalizando



- **Requisitos**
  - Organize o número binário tal que formem o padrão 1.xxxxx
  - Ajuste para que todos tenham um dominante
    - Decremente o expoente com um deslocamento à esquerda

Valor	*Binário*	*Fração*	*Expoente*
128	10000000	1.0000000	7
15	00001101	1.1010000	3
17	00010001	1.0001000	4
19	00010011	1.0011000	4
138	10001010	1.0001010	7
63	00111111	1.1111100	5

Arredondando

1.BBGRXXX

Bit de Guarda: LSB do resultado

Bit arredondamento: 1st bit removido

Sticky bit: OU dos bits restantes

- **Condições de Arredondamento**
  - R = 1, Sticky = 1 → > 0.5
  - G = 1, R = 1, Sticky = 0 → Round to even

Valor	*Fração*	*GRS*	*Incr?*	*Arredondado*
128	1.0000000	000	N	1.000
15	1.1010000	100	N	1.101
17	1.0001000	010	N	1.000
19	1.0011000	110	S	1.010
138	1.0001010	011	S	1.001
63	1.1111100	111	S	10.000

Números Interessantes

Descrição	*exp*	*frac*	*Valor Numérico*
• **Zero**	**00…00**	**00…00**	**0.0**
• **Menor Pos. Denorm.**	**00…00**	**00…01**	$2^{-\{23,52\}} \times 2^{-\{126,1022\}}$
• Single ≈ 1.4×10^{-45}			
• Double ≈ 4.9×10^{-324}			
• **Maior Denormalizado**	**00…00**	**11…11**	$(1.0 - \varepsilon) \times 2^{-\{126,1022\}}$
• Single ≈ 1.18×10^{-38}			
• Double ≈ 2.2×10^{-308}			
• **Menor Pos. Normalizado**	**00…01**	**00…00**	$1.0 \times 2^{-\{126,1022\}}$
• Apenas maior que o maior desnormalizado			
• **Um**	**01…11**	**00…00**	**1.0**
• **Maior Normalizado**	**11…10**	**11…11**	$(2.0 - \varepsilon) \times 2^{\{127,1023\}}$
• Single ≈ 3.4×10^{38}			
• Double ≈ 1.8×10^{308}