ILP500 – Laboratório de Arquitetura e Organização de Computadores
Prof. Sérgio Luiz Banin

Registro da Aula

Números Reais em Binário – Números em ponto flutuante – Revisão e Conclusão

Exercícios propostos na aula passada

$(0,23)_{10}$ ~= (0,00111)$_2$ ← Resultado

0,23	x 2	0,46	0	0,5	0
0,46	x 2	0,92	0	0,25	0
0,92	x 2	1,84	1	0,125	0,125
0,84	x 2	1,68	1	0,0625	0,0625
0,68	x 2	1,36	1	0,03125	0,03125
					0,21875

4,9%

0,23			0	0,5	0
			1	0,25	0,25
			0	0,125	0
			0	0,0625	0
			0	0,03125	0
					0,25

8,7%

$(0,87)_{10}$ ~= (0,11100)$_2$ ← Resultado

0,87	x 2	1,74	1	0,5	0,5
0,74	x 2	1,48	1	0,25	0,25
0,48	x 2	0,96	0	0,125	0
0,96	x 2	1,92	1	0,0625	0,0625
0,92	x 2	1,84	1	0,03125	0,03125
					0,84375

Erro a menor

3,0%

0,87			1	0,5	0,5
			1	0,25	0,25
			1	0,125	0,125
			0	0,0625	0
			0	0,03125	0
					0,875

Erro a maior

0,6%

$(0,155)_{10}$ ~= (0,00101)$_2$ ← Resultado

0,155	x 2	0,31	0	0,5	0
0,31	x 2	0,62	0	0,25	0
0,62	x 2	1,24	1	0,125	0,125
0,24	x 2	0,48	0	0,0625	0
0,48	x 2	0,96	0	0,03125	0
					0,125

Erro a menor
19,4%

0,155			0	0,5	0
			0	0,25	0
			1	0,125	0,125
			0	0,0625	0
			1	0,03125	0,03125
					0,15625

Erro a maior
0,8%

$(0,306)_{10}$ ~= (0,01010)$_2$ ← Resultado

0,306	x 2	0,612	0	0,5	0
0,612	x 2	1,224	1	0,25	0,25
0,224	x 2	0,448	0	0,125	0
0,448	x 2	0,896	0	0,0625	0
0,896	x 2	1,792	1	0,03125	0,03125
					0,28125

Erro a menor
8,1%

0,306			0	0,5	0
			1	0,25	0,25
			0	0,125	0
			1	0,0625	0,0625
			0	0,03125	0
					0,3125

Erro a maior
2,1%

$(0,296)_{10}$ ~= (0,01010)$_2$ ← Resultado

0,296	x 2	0,592	0	0,5	0
0,592	x 2	1,184	1	0,25	0,25
0,184	x 2	0,368	0	0,125	0
0,368	x 2	0,736	0	0,0625	0
0,736	x 2	1,472	1	0,03125	0,03125
					0,28125

Erro a menor
5,0%

0,296			0	0,5	0
			1	0,25	0,25
			0	0,125	0
			1	0,0625	0,0625
			0	0,03125	0
					0,3125

Erro a maior
5,6%

$(0,18)_{10}$ ~= (0,00110)$_2$ ← Resultado

0,18	x 2	0,36	0	0,5	0
0,36	x 2	0,72	0	0,25	0
0,72	x 2	1,44	1	0,125	0,125
0,44	x 2	0,88	0	0,0625	0
0,88	x 2	1,76	1	0,03125	0,03125
					0,15625

Erro a menor
13,2%

0,18			0	0,5	0
			0	0,25	0
			1	0,125	0,125
			1	0,0625	0,0625
			0	0,03125	0
					0,1875

Erro a maior
4,2%

Trabalho em grupo
Escreva um programa em Python 3 que leia um número real no intervalo aberto (0, 1) e gere suas representações em binário com diferentes precisões (quantidade de bits), começando com 5 bits e indo até 12 bits, de um em um.
A saída deve ser apresentada conforme o modelo abaixo.
Importante: Não é permitido usar nenhuma biblioteca de Python específica para essa conversão.

```
Dado de entrada = 0.23
Resultados
Com 5 bits
Aproximação a menor: 0.00111 -> 0.21875 com erro = 4.9%
Aproximação a maior: 0.01000 -> 0.25 com erro = 8.7%
Com 6 bits
Aproximação a menor: 0.001110 -> 0.21875 com erro = 4.9%
Aproximação a maior: 0.001111 -> 0.234375 com erro = 1.9%
...
Com 12 bits
Aproximação a menor: ...
Aproximação a maior: ...
```

(veja o enunciado completo no Teams)

Padrão IEEE-754

Números Reais em ponto flutuante e Números Reais Normalizados

$17{,}316 \times 10^{0} = 1{,}7316 \times 10^{1} = 0{,}17316 \times 10^{2} = 173{,}16 \times 10^{-1}$

Forma normalizada: $1{,}7316 \times 10^{1}$
O que caracteriza a forma normalizada é que à esquerda do ponto decimal nós temos apenas um dígito, obrigatoriamente, diferente de zero.

$213{,}8 = 2{,}138 \times 10^{2}$

$3125{,}98 = 3{,}12598 \times 10^{3}$

$1{,}446 = 1{,}446 \times 10^{0}$

$0{,}388 = 3{,}88 \times 10^{-1}$

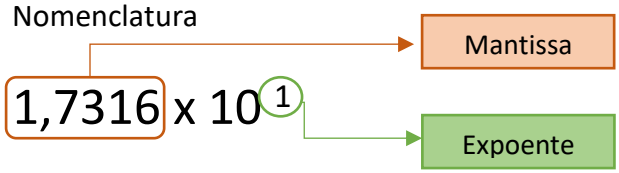
$0{,}00188 = 1{,}88 \times 10^{-3}$

$23{,}446 = 2{,}3466 \times 10^{1}$

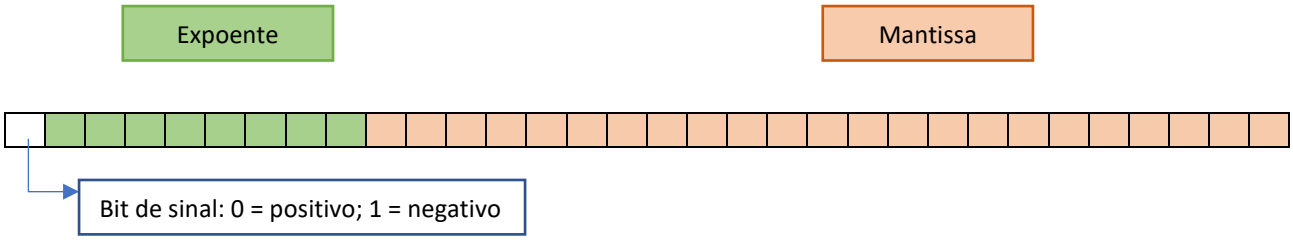
$17{,}0 = 1{,}7 \times 10^{1}$

A mesma ideia se aplica a números binários
$110101{,}0111 = 1{,}101010111 \times 2^{5}$
$1101{,}0011 = 1{,}1010011 \times 2^{3}$
$0{,}0001101 = 1{,}101 \times 2^{-4}$

Nomenclatura



Padrão IEEE-754 – 32 bits



Como ficaria a quantidade 17,316 representada no Padrão IEEE-754 ?

1. Converter a parte inteira para binário: $(17)_{10} = (10001)_2$
2. Converter a parte decimal para binário: $(0{,}316)_{10} = (0{,}????)_2$

0,316	x 2	0,632	0
0,632	x 2	1,264	1
0,264	x 2	0,528	0
0,528	x 2	1,056	1
0,056	x 2	0,112	0
0,112	x 2	0,224	0
0,224	x 2	0,448	0
0,448	x 2	0,896	0
0,896	x 2	1,792	1
0,792	x 2	1,584	1
0,584	x 2	1,168	1
0,168	x 2	0,336	0
0,336	x 2	0,672	0
0,672	x 2	1,344	1
0,344	x 2	0,688	0
0,688	x 2	1,376	1
0,376	x 2	0,752	0
0,752	x 2	1,504	1
0,504	x 2	1,008	1

3. Juntar as partes e normalizar:
   10001,0101000011100101011 = 1,00010101000011100101011 x 2^4

0	1	0	0	0	0	0	1	1	0	0	0	1	0	1	0	1	0	0	0	0	1	1	1	0	0	1	0	1	0	1	1

4. Conversão do expoente para binário – utiliza-se o método BIAS

Tomamos o valor do expoente = 4
A ele somamos o valor 127
4 + 127 = 131
O valor obtido é convertido para binário
$(131)_{10} = (10000011)_2$

O expoente pode ser um número positivo, negativo ou zero.
Para sua representação precisamos de um método de conversão de base 10 para base 2 que contemple esse requisito.
Poderia-se usar o Complemento de 2, mas não é o que acontece.
Usa-se um método alternativo conhecido como método BIAS

8 bits → 2^8 = 256 combinações → 256 / 2 -1 = 12 = 127 → BIAS

5. Finalização da representação escrevendo o Hexadecimal correspondente

0	1	0	0	0	0	0	1	1	0	0	0	1	0	1	0	1	0	0	0	0	1	1	1	0	0	1	0	1	0	1	1
4				1				8				A				8				7				2				B			

$(17{,}316)_{10} = (41\ 8A\ 87\ 2B)_{IEEE754}$

Exercícios: converta de decimal para o padrão IEEE-754, os seguintes números:

$(0,1)_{10}$ = (???)$_{IEEE754}$

$(105,15)_{10}$ = (???)$_{IEEE754}$

$(45319,316)_{10}$ = (???)$_{IEEE754}$

$(12,0)_{10}$ = (???)$_{IEEE754}$

$(453,075)_{10}$ = (???)$_{IEEE754}$

Refaça todos os valores acima para seu correspondente negativo