Vírgula Flutuante

Trabalho para Casa: **TPC2**

Resolução dos exercícios

Representação de valores em vírgula flutuante precisão simples - IEEE 754

1. Represente os seguintes valores em vírgula flutuante precisão simples (IEEE 754). Apresente o resultado final em hexadecimal.

Decimal	IEEE 754 precisão simples
16,375	0x41830000
-1024	0xC4800000
$515{,}625*10^{-3}$	0x3F040000
$-2.25 * 2^{-128}$	0x80480000

2. Converta para decimal os seguintes valores representados em vírgula flutuante precisão simples (IEEE 754).

IEEE 754 precisão simples	Decimal
0x436a0000	234
0xc4000000	-512
0x00700000	$0.875 * 2^{-126}$
0xff800000	-infinito

Representação de valores em vírgula flutuante: formatos `PEQUENO1` e `PEQUENO2`

Considere 2 novos formatos de vírgula flutuante, representados com 8-bits, baseados na norma IEEE:

- formato `PEQUENO1`:
  - → o bit mais significativo contém o bit do sinal
  - → os 4 bits seguintes formam o expoente (em excesso de 7)
  - → os últimos 3 bits representam a mantissa
- formato `PEQUENO2`:
  - → o bit mais significativo contém o bit do sinal
  - → os 3 bits seguintes formam o expoente (em excesso de 3)
  - → os últimos 4 bits representam a mantissa

Para todos os restantes casos, as regras são as mesmas que as da norma IEEE (valor normalizado, subnormal/desnormalizado, representação do 0, ± `infinito`, `NaN`).

3. Complete a expressão que, a partir dos campos em binário, permite calcular o valor em decimal para cada um dos formatos normalizados: $V= (-1)^S * 1.F * 2^{??}$

PEQUENO1 : $V= (-1)^S * 1.F * 2^{E-7}$ PEQUENO2 : $V= (-1)^S * 1.F * 2^{E-3}$

4. Para <u>ambos</u> os formatos, apresente os seguintes valores em decimal:

a) O maior número finito positivo

PEQUENO1 : **0 1110 111 = 1.111*2^7 = 1111.*2^4 = 15*16 = 240**

PEQUENO2 : **0 110 1111 = 1.1111*2^3 = 1111.1 = 15.5**

b) O número negativo normalizado mais próximo de zero

PEQUENO1 : **1 0001 000 = -1.0*2^{-6} = -0.015625**

PEQUENO2 : **1 001 0000 = -1.0*2^{-2} = -0.25**

c) O maior número positivo subnormal/desnormalizado

PEQUENO1 : **0 0000 111 = 0.111*2^{-6} = 0.875 * 0.015625 = 0.013671875**

PEQUENO2 : **0 000 1111 = 0.1111*2^{-2} = 0.9375 * 0.25 = 0.234375**

d) O número positivo subnormal/desnormalizado mais próximo de zero

PEQUENO1 : **0 0000 001 = 0.001*2^{-6} = 0.125 * 0.015625 = 0.001953125**

PEQUENO2 : **0 000 0001 = 0.0001*2^{-2} = 0.0625 * 0.25 = 0.015625**

e) O maior número inteiro positivo múltiplo de 4

PEQUENO1 : **0 1110 111 = 1.111*2^7 = 11110000_2 = 240**

PEQUENO2 : **0 110 1000 = 1.1000*2^3 = 1100.0 = 12**

5. Calcule os valores (número real, ± `infinito`, `NaN`) correspondentes aos seguintes padrões de bits no formato `PEQUENO1`:

a) 0xBB = 1 0111 011_2 = -1.011_2 * 2^0 = -1.375

b) 0x7C = 0 1111 100_2 = **NaN** **(nota: caso especial)**

c) 0x92 = 1 0010 010_2 = -1.01_2 * 2^{-5} = -1.25 * 2^{-5}

d) 0x05 = 0 0000 101_2 = 0.101_2 * 2^{-6} = 0.625 * 2^{-6} **(caso especial-subnormal)**

e) 0x41 = 0 1000 001_2 = 1.001_2 * 2^1 = 10.01_2 = 2.25

6. Codifique os seguintes valores como números de vírgula flutuante no formato `PEQUENO1`:

a) –110.01_3 = -12.11111_{10} = -1100_2 * 2^0 = -1.100_2 * 2^3 = 1 1010 100 = 0xD4

b) 1/16 Ki (por exemplo, para representar a dimensão de um ficheiro em *bytes*)

= 2^{-4} * 2^{10} = 1. * 2^6 = 0 1101 000 = 0x68

c) –0x28C = -10 1000 1100_2 * 2^0 = -1.010001100 * 2^9 => 1 1111 000 = +infinito

d) 101.01_{10} ~ 11000101.001_2 * 2^0 = 1.1000101001_2 * 2^7 ~ 0 1110 100 = 0x74

e) 0.006_8 = 0.000000110_2 * 2^0 = 0.110_2 * 2^{-6} = 0 0000 110 = 0x06

7. Converta os seguintes números `PEQUENO1` em números `PEQUENO2`. *Overflow* deve ser representado por `± infinito`, *underflow* por ±0 e arredondamentos deverão ser para o valor par mais próximo.

a) 0xB5 **= 1 0110 101 = -1.101$_2$ * 2^{-1} => PEQUENO2: E=2, 1 010 1010 = 0xAA**

b) 0xEA **= 1 1101 010 = -1.010$_2$ * 2^{6} => PEQUENO2: E=9 (-∞), 1 111 0000 = 0xF0**

c) 0x14 **= 0 0010 100 = 1.100$_2$ * 2^{-5} => PEQUENO2: E=-2 (0), 0 000 0000 = 0x00**

d) 0xCF **= 1 1001 111 = -1.111$_2$ * 2^{2} => PEQUENO2: E=5, 1 101 1110 = 0xDE**

e) 0x02 **= 0 0000 010 = 0.010$_2$ * 2^{-6} => PEQUENO2: E=-3 (0), 0 000 0000 = 0x00**

8. Considere o desenvolvimento de código científico em C para execução num *notebook* atual, cuja especificação impõe que os números reais sejam representados com pelo menos 8 algarismos significativos. **Indique, justificando**, se consegue representar essas variáveis como `float` ou se tem de as representar como `double`.

9. Um valor do tipo real (*float*) vem representado na norma IEEE 754 por **V= (-1)^{S} * 1.F * 2$^{(Exp-127)}$**, se estiver normalizado. **Indique, explicitando** os cálculos, qual o maior inteiro ímpar que é possível representar exatamente, neste formato.

10. O formato RGBE é usado para representar de forma compacta pixéis com elevada gama dinâmica (em inglês High Dynamic Range - HDR). Cada pixel de uma imagem HDR é representado usando 3 valores reais positivos. São 3 valores porque são usadas 3 cores primárias: Red, Green and Blue (RGB). Os valores dos pixéis são sempre >= 0.

Se fossem usados valores em vírgula flutuante precisão simples seriam necessários 12 bytes para cada pixel; o formato RGBE permite usar 4 bytes para cada pixel. A ideia é que o expoente é partilhado pelos 3 canais (R,G e B) e representado no 4º byte. A parte fraccionária da mantissa de cada canal usa 8 bits; a parte inteira da mantissa não é representada e é igual a 0. O algoritmo para codificar um pixel é o seguinte:

- identificar o canal (R, G ou B) com valor máximo: chamemos-lhe $V_{max} = max(V_R, V_G, V_B)$;
- calcular uma constante de normalização que seja uma potência de 2, $N = 2^E$, tal que $\frac{V_{max}}{2^E} \in [0.5 \cdots 1[$;
- normalizar os valores dos 3 canais: $(V_{nR}, V_{nG}, V_{nB}) * 2^E = \left(\frac{V_R}{2^E}, \frac{V_G}{2^E}, \frac{V_B}{2^E}\right) * 2^E = (V_R, V_G, V_B)$;
- codificar a parte fraccionária de V_{nR}, V_{nG} e V_{nB} em 8 bits cada e codificar o expoente E em 8 bits usando excesso de 128 (nota: o sinal não é codificado explicitamente porque os valores são sempre >= 0).

Codifique o pixel com o valor (24, 20, 6) em RGBE apresentando a respectiva sequência de bits em hexadecimal.

$V_{max} = max(24,20,6) = 24$

$N = 2^5 = 32$, tal que $\frac{24}{32}$=0.75 $\in [0.5 \cdots 1[$

$$(0.75, 0.625, 0.1875) * 32 = \left(\frac{24}{32}, \frac{20}{32}, \frac{6}{32}\right) * 32$$

$(0.75, 0.625, 0.1875, 5 + 128)$ =(11000000, 10100000, 00110000, 10000101)

`Representando os 32 bits em hexa: 0xC0A03085`