Gabarito dos Exercícios dos Capítulos I e II

1) Converter para decimal os seguintes números binários:

a. 10011 b) 11100010 c) 1000001 d) 1,1 e) 1100,01 f) 1000,001

a. 10011 .... $2^4 + 2^1 + 2^0 =$ **19**
b. 11100010 .... $2^7 + 2^6 + 2^5 + 2^1 =$ **226**
c. 1000001 .... $2^6 + 2^0 =$ **65**
d. 1,1 .... $2^0 + 2^{-1} =$ **1,5**
e. 1100,01 .... $2^3 + 2^2 + 2^{-2} =$ **12,25**
f. 1000,001 .... $2^3 + 2^{-3} =$ **8,125**

2) Converter para binário os seguintes números decimais:

a) 23; b) 2615; c) 2,5; d) 0,1; e) 3,8; f) 10,05

a. 23 .... **10111**
b. 2615 .... **101000110111**
c. 2,5 .... **10,1**
d. 0,1 .... **0,000110011001100**...
e. 3,8 .... **11,110011001100**...
f. 10,05 .... **1010,0000110011001100**...

3) Um computador armazena números reais utilizando 1 bit para o sinal do número, 7 bits para o expoente e 8 bits para a mantissa. Admitindo que haja arredondamento, como ficariam armazenados os seguintes números decimais?

a) 265; b) 12,5; c) -445,25; d) -0,1; e) -12,8; f) 2500,05

Os sete bits do expoente variarão de 00000001 até 11111110, isto é, de 1 até 126. Como precisamos representar expoentes negativos, vamos considerar que 63 representa 0 (zero), fazendo um deslocamento no conjunto dos números a serem representados. Assim, o expoente a ser representado deverá ser somado a 63 , para obter-se o valor a ser escrito nos sete bits reservados para o expoente. Dessa forma, quando se escrever 0000001, estaremos representando – 62 , pois –62 + 63 vale 1 (0000001).

Para representar o expoente 0 (zero), deve-se escrever 63 (0111111) , pois 0+63 = 63 . Para representar –1 escreve-se 62 (-1+63 = 62); para ter-se o expoente +1 , representa-se 64 (1+63 = 64). O maior expoente será, portanto, 126 – 63 = 63 . Dessa forma os expoentes, na forma normalizada, variarão de

–62 a + 63.

Lembramos que o expoente 0000000 será utilizado para o "underflow" gradual, forma não normalizada, permitindo obter valores mais próximos a zero que os da forma normalizada. Nesse caso, o expoente passa a valer –62 e a mantissa deixa de estar normalizada, passando a ser: 0, _ _ _ _ _ _ _ _.

Os números, na forma normalizada, terão 8 casas após a vírgula.

a) 265 .... 100001001 .... 1,00001001 x $2^{(8)}$

8 + 63 = 71 ...(em sete bits) ... 1000111

0	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**

b) 12,5 .... 1100,1 .... 1,10010000 x $2^{(3)}$

3 + 63 = 66 ... 1000010

0	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**

c) –445,25 .... –110111101,01 ....-1,1011110101 x $2^{(8)}$

8 + 63 = 71 ... 1000111

1	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**1**

d) –0,1 .... –0,0001100110011100...(dízima periódica)...-1,10011010 x $2^{(-4)}$

-4 + 63 = 59 ... 0111011

1	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**

e) –12,8 .... –1100,11001100...(dízima periódica)... –1,10011001 x $2^{(3)}$

3+63 = 66 ... 1000010

1	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**

f) 2500,05 ... 100111000100,000011001100...(dízima periódica)...

1,00111001 x $2^{(11)}$

11+63 = 74 ... 1001010

0	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**

4) Qual o valor verdadeiramente representado em cada caso acima ?

a) 265 (exato)

b) 12,5 (exato)

c) –445,25 (-445)

d) –0,1 (-0,000110011010 = -($2^{(-4)} + 2^{(-5)} + 2^{(-8)} + 2^{(-9)} + 2^{(-11)}$) = -0,10009765625)

e) –12,8 (-1100,11010 = -12,8125)

f) 2500,05 (1,00111001 x $2^{(11)}$ = 100111001000 = 2504)

5) Qual o maior e o menor número positivo nele representável ?

Maior número positivo: 0111111011111111

M = 1,11111111 x $2^{(63)}$ = (2-$2^{(-8)}$) x $2^{(63)}$ = **65408**

Menor número positivo:

Forma normalizada: 0000000100000000

m = 1,00000000 x $2^{(-62)}$ = $2^{(-62)}$ = **0,00006103515625**

Forma não normalizada: 0000000000000001

m = 0,00000001 x $2^{(-62)}$ = $2^{(-8)}$ x $2^{(-62)}$ = $2^{(-70)}$ = **0,0000002384185791 02**

6) Qual o menor número maior que 100 , nele representável ?

100 = 1100100 = 1,10010000 x $2^{(6)}$

O próximo número será: 1,10010001 x $2^{(6)}$ = 1100100,01 = **100,25**

7) Qual o maior número menor que 20 , nele representável ?

20 = 10100 = 1,01000000 x $2^{(4)}$

O número anterior será: 1,00111111 x $2^{(4)}$ = 10011,1111 = **19,9375**

8) Quais os erros absoluto e relativo ao se tentar nele representar os números: m = 25,5 ; n = 120,25 ; p = 2,5 ; a = 460,25 ; b = 24,005 ?

O erro relativo de qualquer número, ao ser representado, será inferior a $2^{(-8)}$ »

$4 \times 10^{(-3)}$, onde 8 é o número de bits da mantissa.

m = 25,5 = 11001,1 = $1,10011000 \times 2^{(4)}$ ... exato

n = 120,25 = 1111000,01 = $1,11100001 \times 2^{(6)}$ ... exato

p = 2,5 = 10,1 = $1,01000000 \times 2^{(1)}$ ... exato

a = 460,25 = 111001100,01 , representado por: $1,11001100 \times 2^{(8)} = 460$

erro absoluto igual a 0,01 = 0,25

erro relativo igual a $0,25 / 460 \approx 5,5 \times 10^{(-4)} < 2^{(-8)} \approx 4 \times 10^{(-3)}$

b = 24,005 = 11000,00000001010001111 0... ,

representada por $1,10000000 \times 2^{(4)}$

erro absoluto igual a 0,005

erro relativo igual a $0,005/24 \approx 2,1 \times 10^{(-4)} < 2^{(-8)} \approx 4 \times 10^{(-3)}$

9) Usando os valores acima, trabalhando em binário, qual o resultado das operações abaixo, bem como os erros absoluto e relativo ?

m + n , m . p , n . p , a + b , a – b , a / n

Obs: nas operações matemáticas, além da propagação dos erros que os operadores trazem, ao final de cada operação, pode ocorrer arredondamento, trazendo mais erros para o resultado. Isso precisa ser previsto.

Calcularemos os resultados das operações e os erros relativos, podendo os erros absolutos serem estimados pela multiplicação dos erros relativos pelos resultados das operações.

m + n

$m = 1,10011000 \times 2^{(4)}$

$n = 1,11100001 \times 2^{(6)}$

para fazer a soma vamos desnormalizar o número com menor expoente, para que assuma o expoente do maior.

$m = 0,0110011000 \times 2^{(6)}$

$n = 1,1110000100 \times 2^{(6)}$

somando-se obtem-se:

$m + n = 10{,}01000111 \times 2^{(6)}$

normalizando-se o resultado, haverá a perda de um bit, com o surgimento de mais uma causa de erro.

$m + n = 1{,}00100100 \times 2^{(7)} = 10010010{,}0 =$ **146**

erro absoluto de 0,25 originado pelo bit abandonado

erro relativo de $0{,}25 / 146 \approx 1{,}8 \times 10^{(-3)} < 2^{(-8)} \approx 4 \times 10^{(-3)}$

Observe-se que as parcelas m e n , neste caso, não traziam erro; se trouxessem, haveria propagação desses erros, além do arredondamento já referido.

m . p

$m = 1{,}10011000 \times 2^{(4)}$

$p = 1{,}01000000 \times 2^{(1)}$

$m . p = 1{,}11111110 \times 2^{(5)} =$ **63,75** (exato)

neste caso nem há propagação de erros, inexistentes em m e p, nem há arredondamento do resultado.

n . p

$n = 1{,}11100001 \times 2^{(6)}$

$p = 1{,}01000000 \times 2^{(1)}$

$n . p = 10{,}0101100101 \times 2^{(7)}$, o resultado não está normalizado; é preciso normalizá-lo.

$n . p = 1{,}00101101 \times 2^{(8)} =$ **301**, com arredondamento do resultado.

O resultado exato é: 300,625 .

erro absoluto de 0,375

erro relativo de $0{,}375 / 301 \approx 1{,}25 \times 10^{(-3)} < 2^{(-8)} \approx 4 \times 10^{(-3)}$

a + b

$a = 1{,}11001100 \times 2^{(8)}$, com erro relativo de $5{,}5 \times 10^{(-4)}$

$b = 1{,}10000000 \times 2^{(4)}$, com erro relativo de $2{,}1 \times 10^{(-4)}$

para fazer a soma, vamos desnormalizar o número com menor expoente, para

que assuma o expoente do maior.

a = 1,11001100 x $2^{(8)}$

b = 0,00011000 x $2^{(8)}$

a + b = 1,11100100 x $2^{(8)}$ = **484** , sendo 484,255 o resultado exato. O erro relativo é, portanto, 0,255 / 484 » 5,3 x $10^{(-4)}$.

Podemos estimar este erro pelos erros das parcelas. O erro relativo da soma é igual à soma dos erros relativos das parcelas, ponderados pela participação de cada parcela na soma. Sendo e o erro relativo, podemos afirmar:

e (a+b) » e (a) .a/(a+b) + e (b).b/(a+b). No caso:

e (a+b) » 5,5 x $10^{(-4)}$. 460/484 + 2,1 x $10^{(-4)}$. 24/484 = 5,3 x $10^{(-4)}$

a – b

a = 1,11001100 x $2^{(8)}$, com erro relativo de 5,5 x $10^{(-4)}$

b = 1,10000000 x $2^{(4)}$, com erro relativo de 2,1 x $10^{(-4)}$

para fazer a subtração, vamos desnormalizar o número com menor expoente, para que assuma o expoente do maior.

a = 1,11001100 x $2^{(8)}$

b = 0,00011000 x $2^{(8)}$

a – b = 1,10110100 x $2^{(8)}$ = **436** , sendo 436,245 o resultado exato. O erro relativo é, portanto, 0,245 / 436 » 5.6 x $10^{(-4)}$.

Podemos estimar este erro pelos erros de a e b. O erro relativo da subtração é igual à soma dos erros relativos das partes, ponderados pela participação de cada parte na subtração. Sendo e o erro relativo, podemos afirmar:

e (a-b) » e (a) .a/(a-b) + e (b).b/(a-b). No caso:

e (a-b) » 5,5 x $10^{(-4)}$. 460/436 + 2,1 x $10^{(-4)}$. 24/436 = 5,9 x $10^{(-4)}$

Insisto que, tanto na subtração como na soma, não se subtrai erros, erros são sempre somados por seus valores absolutos, admitindo-se sempre a pior hipótese, por segurança. Na previsão dos erros, erra-se sempre para mais, nunca para menos.

a / n

a = 1,11001100 x $2^{(8)}$, com erro relativo de 5,5 x $10^{(-4)}$

n = 1,11100001 x $2^{(6)}$, valor exato

a / n » 0,111101001 x $2^{(2)}$ = 1,11101001 x $2^{(1)}$ = **3,82031** , sendo 3,82744 o resultado, com cinco casas decimais. O erro relativo é, portanto, 0,00713/3,82 » 0,0019 .

Neste exemplo, além da propagação do erro do componente a, há, ainda, o arredondamento da operação, com erro relativo de 4 x $10^{(-3)}$, conforma já citado.

O erro obtido, 1,9 x $10^{(-3)}$, é bem inferior ao erro máximo pelo arredondamento.

10) Seja um computador binário, cujo sistema de ponto flutuante tenha 1 bit para o sinal do número, 5 bits para o expoente e 6 bits para a mantissa, num total de 12 bits. Responda:

a. qual o menor número positivo e o maior número positivo nele representável ?
b. qual o maior $\varepsilon > 0$, tal que 4,25 + ε = 4,25
c. qual o menor número maior que 4,25, nele representável ?
d. qual o maior número menor que 80 , nele representável ?
e. efetue, nele, a multiplicação 0,8 x 5 e indique o resultado.

A gama de variação do expoente é de 00001 a 11110; isto é, de 1 a 30. Tomando 15 como representando o zero, 1 será -14 e 30 será mais 15. Nos cinco bits reservados para o expoente, representaremos o expoente desejado mais quinze. Assim, quando quisermos representar o expoente -14 escreveremos +1, quando desejarmos o expoente zero, representaremos +15, quando quisermos o expoente +15, representaremos +30.

a. menor número positivo

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

Estou assumindo que se trata do menor número não normalizado, para podermos chegar ainda mais próximo a zero (underflow gradual).

m = 0,000001 x $2^{(-14)}$ = $2^{(-20)}$

maior número positivo

0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

M = 1,111111 x $2^{(15)}$ = (2-$2^{(-6)}$) x $2^{(15)}$ = 65024

b- maior $\varepsilon > 0$, tal que 4,25 + ε = 4,25

$4,25 = 100,01 = 1,000100 \times 2^{(2)}$

$e = 0,00000001111111 \times 2^{(2)}$, para que, ao somar com

$4,25 = 1,00010001111111 \times 2^{(2)}$, o resultado, ao ser arredondado para mantissa com seis bits depois da vírgula, mantenha o valor original de 4,25 , pois só altera a partir do oitavo bit a partir da vírgula.

Logo $e = 0,00000001111111 \times 2^{(2)}$, que ao ser normalizado fica:

$e = 1,111111 \times 2^{(-6)}$. Logo $e = (2-2^{(-6)}) \times 2^{(-6)} = 127/64/64$

$e = 0,031005859375$

c) próximo número maior que 4,25 será: $1,000101 \times 2^{(2)} = 100,0101 =$ **4,3125**

d) maior número menor que 80

$80 = 1010000 = 1,010000 \times 2^{(6)}$

$1,001111 \times 2^{(6)} = 1001111 =$ **79**

79 é o maior número menor que 80, nele representável

e) calcular 0,8 x 5

$0,8 = 0,110011001100... = 1,100110 \times 2^{(-1)}$

$5 = 101 = 1,010000 \times 2^{(2)}$

$0,8 \times 5 = 1,1111111 \times 2^{(1)} \text{ » } 10,000000 =$ **4,0**