Pedro Paulo Moussa de Medeiros - 119181709

Marcus Vinícius Torres de Oliveira - 118142223

Lista 1

Questão 1)

Pedro:

SO: Windows 10

Processo: Comando getconf -a | grep CACHE em um terminal virtual Ubuntu

```
adduser@DESKTOP-U26BJ62:/mnt/e/CC/Comp Prog/2021.1$ getconf -a | grep CACHE
LEVEL1_ICACHE_SIZE                 32768
LEVEL1_ICACHE_ASSOC                8
LEVEL1_ICACHE_LINESIZE             64
LEVEL1_DCACHE_SIZE                 32768
LEVEL1_DCACHE_ASSOC                8
LEVEL1_DCACHE_LINESIZE             64
LEVEL2_CACHE_SIZE                  262144
LEVEL2_CACHE_ASSOC                 4
LEVEL2_CACHE_LINESIZE              64
LEVEL3_CACHE_SIZE                  6291456
LEVEL3_CACHE_ASSOC                 12
LEVEL3_CACHE_LINESIZE              64
LEVEL4_CACHE_SIZE                  0
LEVEL4_CACHE_ASSOC                 0
LEVEL4_CACHE_LINESIZE              0
adduser@DESKTOP-U26BJ62:/mnt/e/CC/Comp Prog/2021.1$
```

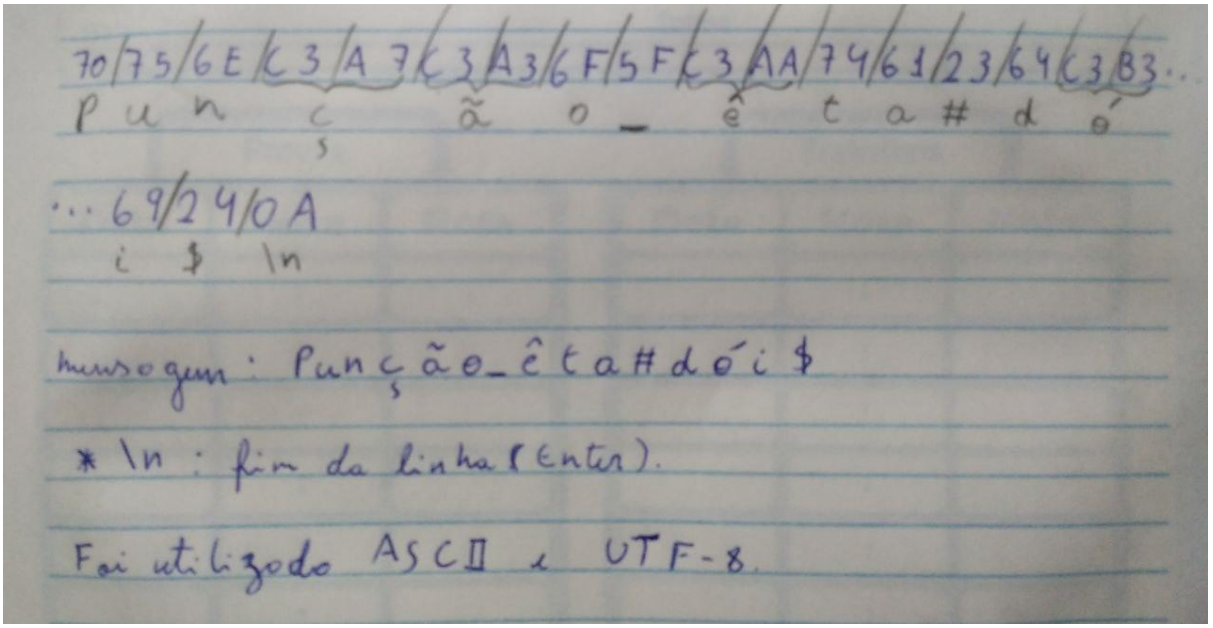
Marcus:

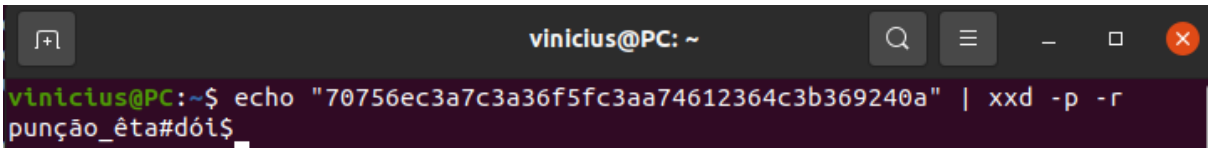
SO: ubuntu 20.04

Processo: Comando lscpu no terminal

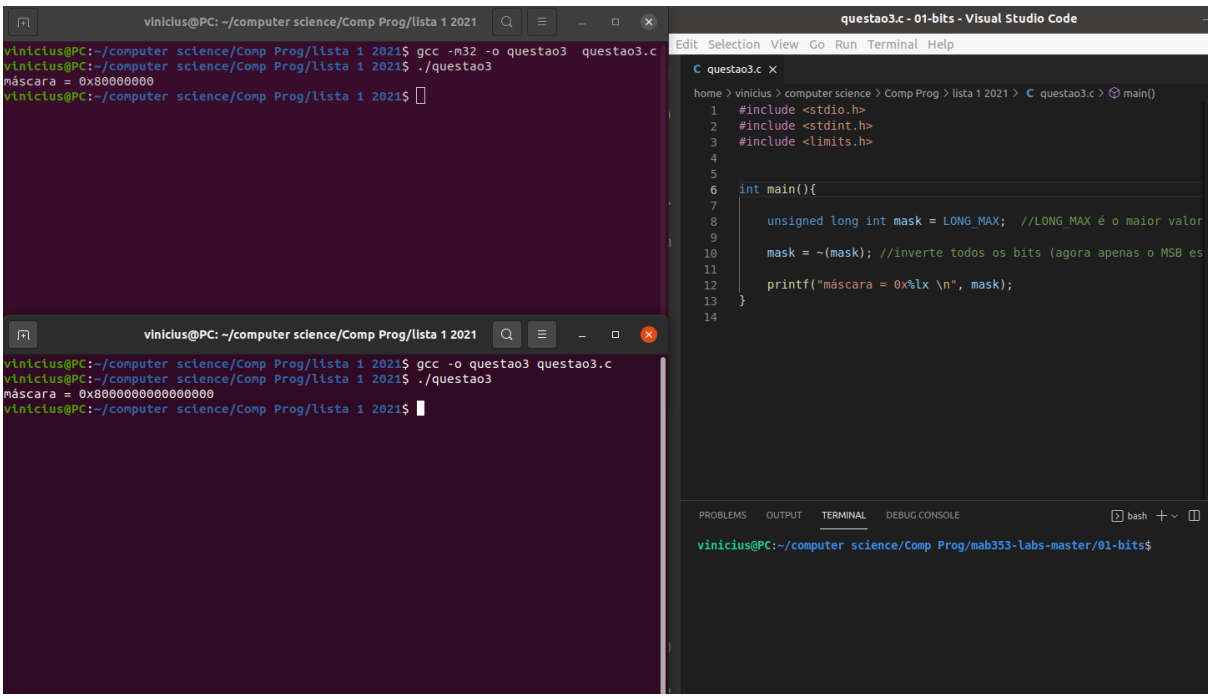
```
vinicius@PC: ~
vinicius@PC:~$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   36 bits physical, 48 bits virtual
CPU(s):                          4
On-line CPU(s) list:             0-3
Thread(s) per core:              2
Core(s) per socket:              2
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           58
Model name:                      Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz
Stepping:                        9
CPU MHz:                         1886.379
CPU max MHz:                     3400,0000
CPU min MHz:                     1600,0000
BogoMIPS:                        6800.55
L1d cache:                       64 KiB
L1i cache:                       64 KiB
L2 cache:                        512 KiB
L3 cache:                        3 MiB
```

Questão 2)





Questão 3)



Questão 4)

a) Representação em precisão dupla: (-1)^s * 2^(e-1023) * 1.f onde s = 0, e = 11111110, f = 11111111111111111111111111111111111111111111111111

0 11111110 11111111111111111111111111111111111111111111111111 = 2^1023 * (1 + (1 - 2^-52))

b) s = 1, e = 00000001, f = 11111111111111111111111111111111111111111111111111 = -2^-1022 * (1 + (1 - 2^-52))

Questão 5 a)

Questão 5

a)

$0,3 \times 2^{-131}$

Passo 1: convertendo para binário

×2	
0,3	(0), 6
0,6	(1), 2
0,2	(0), 4
0,4	(0), 8
0,8	(1), 6

0,6	(1), 2
0,2	(0), 4
0,4	(0), 8
0,8	(1), 6
0,6	(1), 2
⋮	⋮

Assim, $0,3 = 0,0100\overline{1100}1$

Então $0,3 \times 2^{-131} = 0,0100\overline{1100}1 \ldots \times 2^{-131}$

$= 0,0100\overline{1100}1 \ldots \times 2^{-5} \times 2^{-126}$

* Como o expoente < -126, em precisão simples temos que usar a representação não normalizada ajustando a mantissa.

Passo 2: Obtendo a representação em precisão simples

$0,3 \times 2^{-131} = 0,0100\overline{1100}1 \ldots \times 2^{-5} \times 2^{-126}$

$\approx 0,00000\,0100\overline{1100}1 \times 2^{-126} = 0,0100\overline{1100}1 \times 2^{-131}$

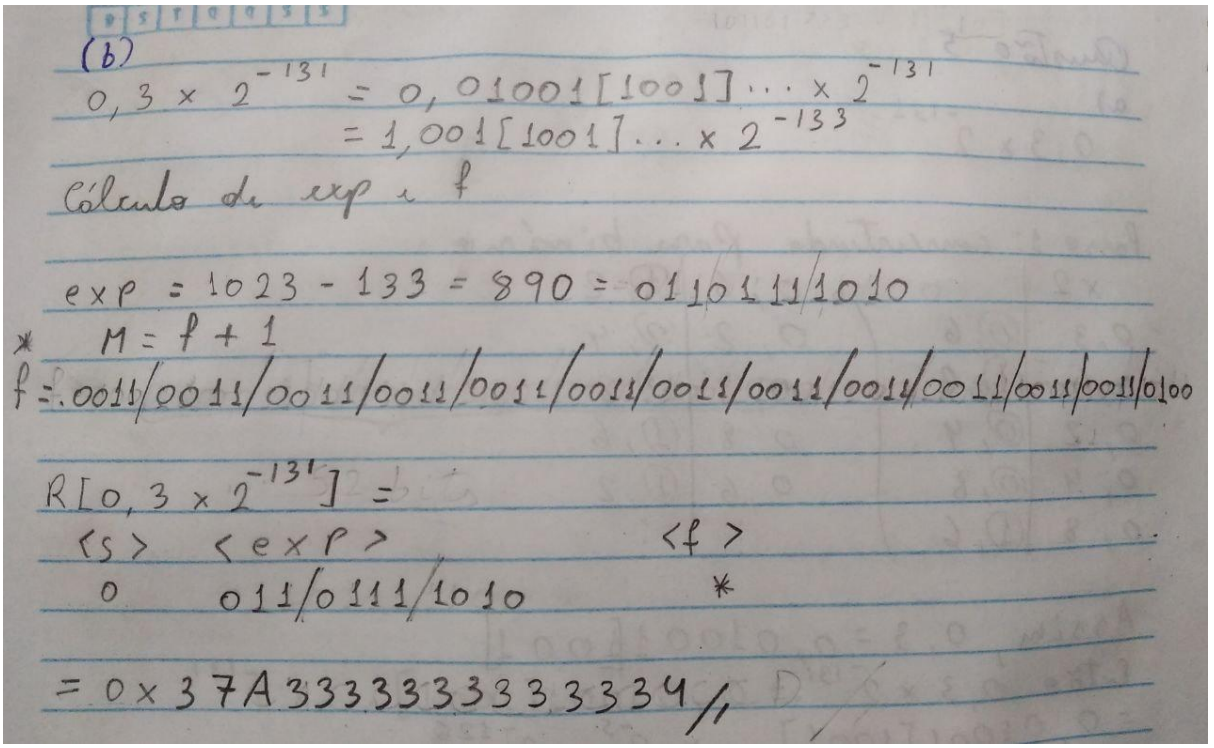
$R[0,3 \times 2^{-131}] =$

<s> <exp> <f>

0 00000000 00000000000001001

$= 0x00000009$

b)



(b)

$0,3 \times 2^{-131} = 0,01001[1001]\ldots \times 2^{-131}$

$= 1,001[1001]\ldots \times 2^{-133}$

Cálculo de exp e f

exp = 1023 - 133 = 890 = 0110 1111 010

* M = f + 1

f = .0011/0011/0011/0011/0011/0011/0011/0011/0011/0011/0011/0100

R[$0,3 \times 2^{-131}$] =

<s>	<exp>	<f>
0	011/0111/1010	*

= 0x37A3333333333334

Questão 6)

a)Pelo método das multiplicações sucessivas:

0,3 x 2 = 0,6
0,6 x 2 = 1,2
0,2 x 2 = 0,4
0,4 x 2 = 0,8
0,8 x 2 = 1,6
0,6 x 2 = 1,2 (começa a repetir)
0,2 x 2 = 0,4
0,4 x 2 = 0,8
0,8 x 2 = 1,6
0,6 x 2 = 1,2

(0.010011001100110011001100)

Em hexadecimal:

-Separando em grupos de 4:

0100 = 4
1100 = C (repete 5 vezes)

(0.4CCCCC)

b)Sabendo que 0,3 em binário é 0.01001100110011001100110011, então 4,3 é 100 + 0.01001100110011001100110011 = 100.01001100110011001100110011, escrevendo agora na forma 1.00010011001100110011001100 x 2^2, f = 00010011001100110011001|100, truncando em 23 bits: 00010011001100110011010

Para o expoente, somamos 2 ao 127 da representação de ponto flutuante em precisão simples e ficamos com 10000001. Como 4,3 é positivo, o bit de sinal é 0.

Por fim, a representação fica 0 10000001 00010011001100110011010

Em hexadecimal:

0100|0000|1000|1001|1001|1001|1001|1010 = 0x4089999A

c)Pelo método das multiplicações sucessivas:

0,2 x 2 = 0,4
0,4 x 2 = 0,8
0,8 x 2 = 1,6
0,6 x 2 = 1,2
0,2 x 2 = 0,4 (começa a repetir)

0.00110011001100110011001100110011001100110011001100110011

Em hexadecimal:

Separando em grupos de 4, temos 0011 repetido 13 vezes, que é igual a 3 em hexadecimal. Logo: (0.3333333333333)

d)Sabendo que 0,2 em binário é 0.00110011001100110011001100110011001100110011001100110011 e 3 é 0011, temos que 3,2 é igual a 11.00110011001100110011001100110011001100110011001100110011. Dividindo por 2, ficamos com 1.10011001100110011001100110011001100110011001100110011 x 2, e agora truncamos em 52 bits, então fica 1.1001100110011001100110011001100110011001100110011010 x 2. Para o expoente, somamos 1 ao 1023 e ficamos com 10000000000, e o bit de sinal é 0.

Por fim, a representação em precisão dupla é:

0 10000000000 1001100110011001100110011001100110011001100110011010

Em hexadecimal:
0100 = 4
0000 = 0
0000 = 0
1001 = 9 (repete 12 vezes)
1010 = A

(0.400999999999999A)

e)
x = 0 10000001 00010011001100110011010

z = 0 10000000000 1001100110011001100110011001100110011001100110011010

Primeiro convertemos x de precisão simples para precisão dupla, então o expoente precisa ter 11 bits, então exp de x = 10000000001. Agora preenchemos a parte fracionária com 0s até atingir 52 bits, f = 0001001100110011001101000000000000000000000000000000.

Para igualar os expoentes, fazemos um shift na mantissa e ficamos com
0 10000000000 1000100110011001100110100000000000000000000000000000.

Agora que os dois possuem o mesmo expoente, só resta subtrair.

(10.0010011001100110011001[0]... x 2) -
(1.1001100110011001100110011001100110011001100110011001101 x 2) =
2 x (0.1000110011001100110011000110011001100110011001100110011) =
1.0001100110011001100110001100110011001100110011001100110011

Em precisão dupla:

0 10000000000 0001100110011001100110001100110011001100110011001100110011

Em hexadecimal:

0001 = 1
0000 = 0
0000 = 0
0001 = 1
1001 = 9
1001 = 9
1001 = 9
1001 = 9
1000 = 8
1100 = C (repete 7 vezes)

(1.00199998CCCCCCC)

f) 2^0 + 1/2^4 + 1/2^5 + 1/2^8 + 1/2^9 + 1/2^12 + 1/2^13 + 1/2^16 + 1/2^17 + 1/2^20 + 1/2^21 + 1/2^24 + 1/2^25 + 1/2^28 + 1/2^29 + 1/2^32 + 1/2^33 + 1/2^36 + 1/2^37 + 1/2^40 + 1/2^41 + 1/2^44 + 1/2^45 + 1/2^48 + 1/2^49

Igualando o denominador:

(2^49 + 2^45 + 2^44 + 2^41 + 2^40 + 2^37 + 2^36 + 2^33 + 2^32 + 2^29 + 2^28 + 2^25 + 2^24 + 2^21 + 2^20 + 2^17 + 2^16 + 2^13 + 2^12 + 2^9 + 2^8 + 2^5 + 2^4 + 2^1 + 1)/2^49 = 619.244.948.763.443 / 562.949.953.421.312 = 1,09999999999999964447286321199499

Com apenas 13 casas decimais: 1,0999999999999|9964447286321199499 = 1,1000000000000 = 1,1

Resultado obtido na execução do programa:

```
pedrowski@DESKTOP-U26BJ62:/mnt/e/CC/Comp Prog/2021.1$ ./l1_q6
y = 4.3 - 3.2 = 1.1000001907349
```

Questão 7)

Questão 7

O procedimento para obter o arredondamento de $x/2^k$ é:

$(x<0 \; ? \; x + (1<<k) - 1 \; : \; x) >> k$

Ou seja, se $x<0$, temos que o procedimento é:

$[x + (1<<k) - 1] >> k$

Veja o exemplo numérico:

Sabemos que $\frac{-12}{8} = \frac{-12}{2^3} = [-1]$ → Parte inteira da fração.

Vamos calcular esse exemplo usando o procedimento.

$\frac{-12}{8} = \frac{-12}{2^3} \Rightarrow x = -12 \text{ e } k = 3$

$[-12 + (1<<3) - 1] >> 3$

$[-12 + (0001<<3) - 1] >> 3$ // Representei 1 em binário

$[-12 + (1000) - 1] >> 3$ // 1000 = 8 em decimal

$[-12 + 8 - 1] >> 3$

$[-5] >> 3$ // -5 = 1011 em binário

$[1011] >> 3$

$[1111] = -8 + 7 = -1$

Basicamente, deve-se somar $2^k - 1$ à x, antes de "shiftar" (deslocar) para direita k vezes. Assim dá para obter o resultado arredondado corretamente.