PCE

1) a) 725883_{10}

$11001\ 010\ 101\ 111\ 11010\ 011_2$
7 2 5 8 8 3

b) 6 3 333 8 2 9 99999 7

0110 1100 0011 1000 0010 1111 1001 0111
6 4× anteriores 3 8 2 6× anterior 9 7

2) a)
```
int funcA (int val, char op){
    if (op == '+')                    /* '+' ASCII 0x2b
        return val+2;
    else
        if (op == '*')
            return val*4;             /* '*' ASCII 0x2a
        return val;
}
```

funcA:

```
1  movb  12(%ebp) ; %dl
2  movl  8(%ebp), %eax
3  cmpb  $43, %dl
4  leal  2(%eax), %ecx
5  je    .L1
6  leal  0(, %eax, 4), %ecx
7  xorl  %eax, %eax
8  cmpb  $42, %dl
9  movl  %ecx, %eax
.L1
11 leave
12 ret
```

1 - Copia conteúdo de memória apontado por %ebp + 12 bytes (ou células) para o registo %dl (variável op)

2 - Copia o conteúdo de memória apontado por %ebp + 8 bytes para o registo %eax (var val)

3 - Compara a constante $43_{10} = 2b_{16}$ com o conteúdo do registo %dl, corresponde a "op == +"

4 - Soma 2 ao conteúdo ~~de~~ %eax e envia o resultado em %ecx

5 - faz salto se o resultado da instrução 3 for uma igualdade

6 - Soma 0 com 4 x conteúdo de memória apontado por %eax e coloca resultado no registo %ecx

?? 7 - disjunção exclusiva, põe o registo a 0

8 - Compara a constante $42_{10} = 0x2a_{16}$ com o conteúdo do registo %dl (var op)

9 - Copia conteúdo de %ecx para %eax

11 - prepara a stack para retorno

12 - faz o retorno da funcA

2 b)

```
int funcB(char *S, char d){
    int i=0;
    for( ; S[i] != '0' ; i++)      /* '0' ASCII 0x30
        if(S[i] == d)
            return i ;
    return 0;
}
```

```
funcB
 1 movl  8(%ebp), %esi
 2 movb  (%esi), %al
 3 xorl  %edx, %edx
 4 cmpb  $48, %al
 5 movb  12(%ebp), %bl
 6 je    .L9
.L7:
 8 cmpb  %bl, %al
 9 movl  %edx, %ecx
10 je    .L1
11 incl  %edx
12 movb  (%edx, %esi), %al
13 cmpb  $48, %al
14 jne   .L7
.L9:
16 xorl  %ecx, %ecx
.L1:
18 movl  %ecx, %eax
...
ret
```

1 - copia conteudo da memoria apontada por (%ebp) + 8 bytes para o registo %esi (corresponde a var S)

2 - copia conteudo da memoria apontada por %esi para o registo %al (var S)

3 - conteúdo de %edx a zero. (i=0)

4 - compara $48_{10} = 0x30_{16}$ ao conteúdo de %al

5 - copia conteudo memoria apontado por %ebp + 12 para %bl (var d)

6 - faz salto se instrução 4 é uma igualdade

8 - compara conteudo de %bl ao conteúdo de %al

9 - copia conteúdo de %edx para %ecx

10 - salto se resultado da instrução 8 é uma igualdade

11 - incremento do conteúdo de %edx (i++)

12 - copia soma do conteúdo da memoria apontado por %edx e %esi para o registo %al

13 - mesmo que 4.

14 - salto se inst 13 for uma diferença

16 - conteúdo de %ecx a zero

18 - copia conteúdo de %ecx p/ %eax

faz o retorno da função

PCE

3 IA16 (inteiros: 16 bits em complemento para 2)

a) $1k = 2^{10}$

Representar 24k

$24k = 16k + 8k = 2^{14} + 2^{13}$

$24k = 0110000000000000_2$

b) positivos: até $2^{16-1} - 1 = 32767$

negativos: até $-2^{16-1} = -2^{15} = -32768$

gama de valores $[-32768, 32767]$

4 6 bits expoente com excesso $2^{(n-1)} - 1$ $\Rightarrow$ excesso 31

9 bits para a fração

1 bit para o sinal

a) representar $-26375 * 10^{-2} = -263{,}75$

$\hookrightarrow 256 + 8 + 1 + 2^{-1} + 2^{-2}$

$= 2^8 + 2^3 + 2^0 + 2^{-1} + 2^{-2}$

$100001001{,}11$

arredondar

$V = (-1)^{1} \times 1{,}000.0100.1.1 \times 2^{8}$

$= E - 31$

$\hookrightarrow E = 39_{10} = 2^5 + 2^2 + 2^1 + 2^0$

$= 100111_2$

Representação no formato:

1	100111	000010011

b) Apresentar intervalo de valores possível representar

valor max para o expoente : $111110_2 = 62_{10}$

valor max para a mantissa: 1111111111_2

parte negativa : $v = (-1)^1 * 1,1111111111 \times 2^{(62-31)} = 31$

$= -4290772992_{10}$

parte positiva : $v = (-1)^0 \times 1,1111111111 \times 2^{31}$

$= 4290772992_{10}$

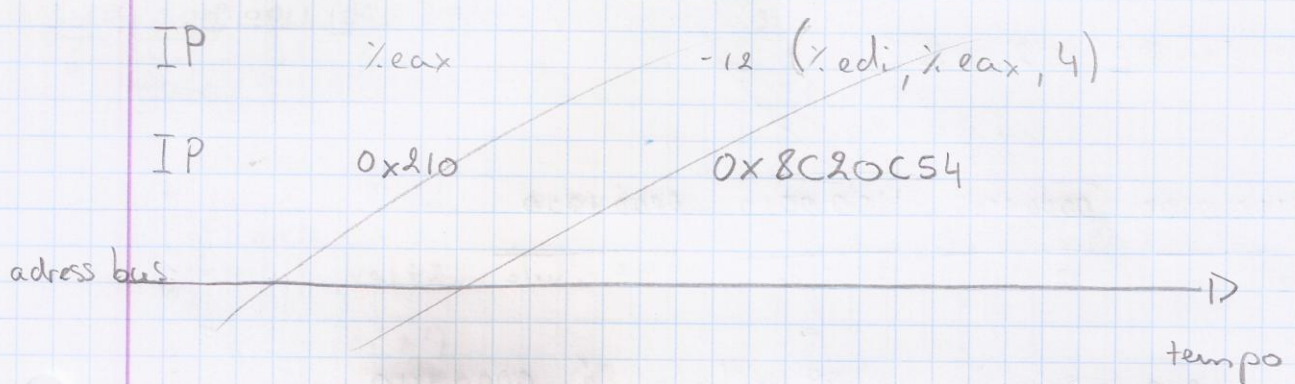
Intervalo de valores: $[-4290772992_{10}, 4290772992_{10}]$

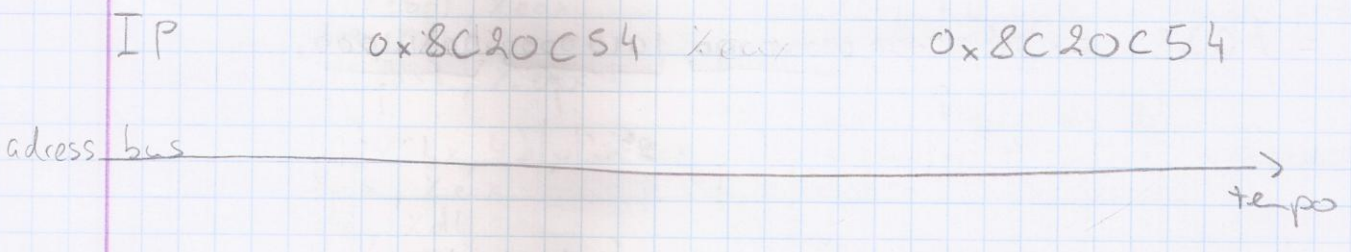
PCE

5 a) addl %eax, -12(%edi, %eax, 4)

A instrução em binário ocupa 4 células de memória

O 2º operando, em memória, é o valor 0x9e28





b) %eax = 0x210

-12(%edi, %eax, 4) = -12_{10} + 0x8c20420 + 4 * 0x210

= 8c20c54

8c20c54 + 210 = 8c20e64

ordenado em little Endian:

64 0E C2 08

ou seja

01100100	00001110	11000010	00001000
1ª célula	2ª célula	3ª célula	4ª célula

O valor armazenado na 2ª célula é 00001110_2

Acho que esta resolução está errada

b) 0x210 + 0x9e28

= A038 = 00000000 00000000 10100000 00111000$_2$

↑ ↑ ↑ ↑

ordenado em little Endian: 4ª célula 3ª célula 2ª célula 1ª célula

valor a armazenar na 2ª célula é 10100000_2

acho que a resolução correcta é esta