PUC-Rio – Software Básico – INF1018
Prova Final – 09/07/2019

1. (3,0 pontos) Considere o programa C a seguir:

```
#include <stdio.h>

void dump (void *p, int n) {
   unsigned char *p1 = (unsigned char *) p;
   while (n--) {
      printf("%p - %02x\n", p1, *p1);
      p1++;
   }
}

struct FP {
   char c;
   float f;
   short s;
   double d;
  } fp = {'e', 125.125, -0xF, 1.625};

int main (void) {
   dump (&fp, sizeof(struct FP));
   return 1;
}
```

   Supondo que `fp` seja armazenado no endereço de memória 0x601040, diga o que o programa irá imprimir quando executado, deixando claro como você chegou a esses valores, mostrando as contas e outras informações usadas. Considere que a máquina de execução é *little-endian*, que as convenções de alinhamento são as do Linux no IA-64 e que o valor do caractere 'a' na tabela ASCII é 97, em decimal. Se houver posições de *padding*, indique seu conteúdo com **PP**. (ATENÇÃO: valores sem contas e explicações **NÃO** valem ponto!)

2. (2,0 pontos) Escreva em C uma função que dado um inteiro qualquer retorne a posição de seu bit '1' mais significativo

```
int maisSig(int padrao);
```

   Caso a entrada *padrao* não possua nenhum bit igual a 1, o retorno da função deve ser -1. Por exemplo:

   - se chamarmos a função *maisSig* passando um inteiro com valor 3, o retorno da função deverá ser 1.
   - se chamarmos a função *maisSig* passando um inteiro com valor 0x7FFFFFFF, o retorno da função deverá ser 30.

3. Traduza as funções `verif` e `flicE` abaixo para assembly IA-64 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros, salvamento de registradores e resultados em C/linux. Traduza o mais diretamente possível o código de C para assembly. (Não se preocupe em entender o que cada função faz, apenas traduza-as literalmente.)

   (a) (2,5 pontos)

```
int verif(int *t, int lim, int n)
{
   int i;
   int acc = 0;

   for (i = 0; i < n; i++)
   {
      if (*t > lim) acc += *t;
      t++;
   }

   return acc;
}
```

   (b) (2,5 pontos)

```
double energ(char c);

double flicE(char *s, int term)
{
   double en = 0.0;
   while (*s != (char)term)
   {
      en += energ(*s);
      s++;
   }

   return en;
}
```