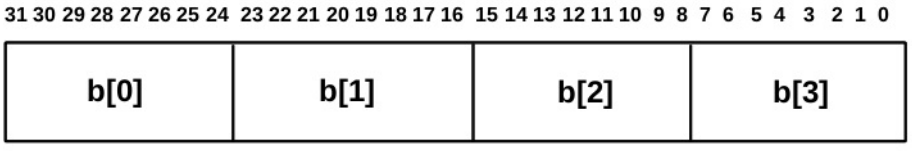
PUC-Rio – Software Básico – INF1018
Prova Final – 14/12/2017

1. (2,5 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 X {
   char c;
   short s;
   double d;
   int i;
  } x = {'d', 2050, -1030.75, -768};

int main (void) {
   dump (&x, sizeof(struct X));
   return 0;
}
```

   Supondo que x seja armazenado no endereço de memória 0x601040, diga o que o programa irá imprimir quando executado, deixando claro como você chegou a esses valores. 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 caracter '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) Suponha que armazenemos num inteiro *unsigned* de 32 bits quatro valores inteiros **com sinal** de um byte cada, numerados da esquerda para a direita, na forma indicada a seguir:



   Escreva uma função C que receba um inteiro *unsigned* neste formato e retorne a posição (de 0 a 3) do **menor valor** dentre os quatro bytes nele armazenados.

   O protótipo da função é

```
int pos_menor(unsigned int packed);
```

3. Traduza as funções `foo` e `boo` 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 g (int i);

void foo (int *a, int n) {
   int i = 0;
   for (i=0; i<n; i++) {
      if (g(i))
         a[i] = 2*i;
   }
}
```

   (b) (3,0 pontos)

```
struct S {
   char id;
   int coef;
   double d;
   struct S *prox;
};
double cut(double d);

double boo(struct S *cad, double val) {
   double sum = 0.0;
   while (cad) {
     sum += cad->d * cad->coef;
     cad = cad->prox;
   }
   return cut(sum)/val;
}
```