PUC-Rio – Software Básico – INF1018
Prova 1 – 10/4/14

1. (2,5 pontos) Considere o programa C a seguir (as interrogações na inicialização de **x** são propositais):

```
#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 {
  int i;
  unsigned short s;
}
  x[2] = {{?,?}, {?,?}};

int main(void) {
  dump(x, 20);
  return 0;
}
```

Imagine que, ao ser executado em uma máquina de arquitetura IA-32 (*little-endian*), obedecendo as convenções de alinhamento do Linux, esse programa imprimiu a saída a seguir:

```
0x804966c - ee
0x804966d - ff
0x804966e - ff
0x804966f - ff
0x8049670 - ff
0x8049671 - ff
0x8049672 - 00
0x8049673 - 00
0x8049674 - 04
0x8049675 - 00
0x8049676 - 01
0x8049677 - 00
0x8049678 - 80
0x8049679 - 80
0x804967a - 00
0x804967b - 00
0x804967c - 00
0x804967d - 00
0x804967e - 00
0x804967f - 00
```

Analisando a saída do programa e a declaração de `x` (um *array* de 2 estruturas), substitua as interrogações ("?") na inicialização de `x` com os valores dos campos respectivos. Expresse esses

valores em notação decimal. **Explique** como você chegou a esses valores (mostre suas contas e a posição dos valores na saída do dump). Valores sem explicação não serão considerados!

ATENÇÃO: O programa pede que o dump imprima 20 bytes. Esse número é suficiente para mostrar todo o conteúdo do *array*, mas **podem ser exibidos bytes a mais!**

2. Traduza as funções `foo` e `bar` abaixo para assembly IA-32 (o assembly visto em sala), utilizando as regras usuais de alinhamento IA-32/Linux, passagem de parâmetros, salvamento de registradores e resultados em C. Traduza o mais diretamente possível o código de C para assembly. *Comente seu código!*

   (a) (2,5 pontos)

```
void boo (int* a);

int foo (int n) {
  int x, i;
  boo(&x);
  for (i=0;i<x;i++)
    n = n*2;
  return n;
}
```

   (b) (3 pontos)

```
struct ss {
  int val;
  int key;
  struct ss* next;
};

int bar (struct ss *um_s) {
  int acc = 0;
  while (um_s != NULL) {
    if (um_s->key == 1)
      acc += um_s->val;
    um_s = um_s->next;
  }
  return acc;
}
```

3. (2 pontos) Escreva uma função `lton` que receba como argumentos um *unsigned int* e um array de 4 bytes, e preencha esse array com os 4 bytes do inteiro em *big-endian*:

```
void lton (unsigned int num, unsigned char* bytes);
```

   Como exemplo, se escrevermos a main a seguir:

```
int main (void) {
  int i;
  unsigned char b[4];
  lton (0xaabbccdd, b);
  for (i=0;i<4;i++)
    printf ("%2x ", b[i]);
  printf ("\n");
  return 0;
}
```

   a saída do programa deverá ser:

```
aa bb cc dd
```