PUC-Rio – Software Básico – INF1018
Prova 1 – 2/10/12 – Turma 3WA

1. (2,5 pontos) Considere o programa C a seguir:

```
#include <stdio.h>
#define MAX 3
void dump (void *p, int n) {
  unsigned char *p1 = (unsigned char *) p;
  while (n--) {
    printf("%p - %02x\n", p1, *p1);
    p1++;
  }
}
int main (void ) {
    int i;
    short s[MAX] = { -1, -2, -3};
    struct No no[MAX];
    for (i=0;i<MAX;i++) {
      no[i].a = -2052+i;
      no[i].c = 0xab>>i;
      no[i].s1 = &(s[i]);
      no[i].s2 = 42 | 520;
    }
    dump (&no[1], sizeof(struct No));
    return 0;
}
```

   Considerando que `no` será alocado na posição de memória 0xbffff7b8, e `s` na posição 0xbffff7a8, diga o que esse programa irá imprimir quando executado, explicando como você chegou aos valores exibidos (ATENÇÃO: valores sem contas e explicações NÃO valem ponto!!!). Suponha que a máquina de execução é *little-endian* e que as convenções de alinhamento são as do Linux no IA-32. Se houver posições de *padding*, indique seu conteúdo com `PP`.

2. Traduza as funções `foo` e `boo` abaixo para assembly IA-32 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros e resultados em C. Traduza o mais diretamente possível o código de C para assembly. Use registradores para abrigar as variáveis locais que aparecem nessas funções. Comente seu código.

   (Não se preocupe se você não souber o que as funções fazem, apenas traduza-as literalmente!)

   (a) (2,5 pontos)

```
int foo (int a[], int pos[]) {
  int soma = 0;
  int prox = pos[0];
  while (prox != 0) {
    soma += a[prox];
    prox = pos[prox];
  }
  return soma;
}
```

(b) (2,5 pontos)

```
struct X {
  char a;
  int status;
};
int f (int v);
int boo (struct X *x, int n) {
  int ac =  0;
  int i;
  for (i=0; i<n; i++,x++)
    if (x->status >0)
      ac += f(x->a);
    else
      ac += x->a;
  return ac;
}
```

3. (2,5 pontos) Considere um tipo `BigInt` com tamanho fixo de 20 bytes:

```
typedef unsigned char BigInt[20];
```

com as características do inteiro grande utilizado no trabalho (little-endian, etc). Escreva, em C, uma função com o protótipo:

```
BigInt bi_sumB (BigInt a, char num);
```

Essa função recebe um BigInt e um inteiro de um byte e retorna um BigInt (novo!) com a soma dos dois valores.