PUC-Rio – Software Básico – INF1612
Prova Final – 9/12/04

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 um {
  int i;
  char c;
  float f;
  short s;
};
struct um a =  {-32,'2',-33.75,1000};
int main (void) {
  dump (&a, sizeof(a));
  return 0;
}
```

   Considerando que `a` seja alocado na posição de memória `0xbffffcd0`, diga o que esse programa irá imprimir quando executado, explicando como você chegou aos valores exibidos (*mostre suas contas!!*). O valor ascii de '0' é 48 (`0x30`). Suponha que a máquina de execução é Pentium-Linux, ou seja, a representação de dados é a vista no curso.

2. Traduza as funções `fat` e `foo` abaixo para assembly IA-32 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros e resultados em C. Comente seu código.

   (a) (2,0 pontos)

```
int fat (int n) {
  if (n==0) return 1;
  else return n*fat(n-1);
}
```

   (b) (3,0 pontos)

```
struct X {
  double a;
  int b;
};
double foo (struct X *p, int n) {
  int i; double s = 0.0;
  for (i=0; i<n; i++)
    s = s + p[i].a + p[i].b;
  return s;
}
```

3. (2,5 pontos) Considere a representação usada no primeiro trabalho para inteiros muito grandes:

```
#define NUM_INTS 4
typedef unsigned int BigInt[NUM_INTS];
```

Como no trabalho, uma variável do tipo `BigInt` é representada por este array que deve ser interpretado como um único inteiro de 128 bits, em complemento a dois, seguindo a ordem little-endian. Desta forma, o inteiro 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE (igual a -2) é representado pelo array:

```
{0xFFFFFFFE, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF }
```

Usando essa representação, implemente em C *uma* (e apenas uma!) entre as operações de soma e shift para a direita, ou seja, uma entre as funções abaixo:

(a)
```
/* res = a + b */
void bi_sum (BigInt res, BigInt a, BigInt b);
```

(b)
```
/* desloca a de n bits para a direita e grava resultado em res */
void bi_shr (BigInt res, BigInt a, int n);
```