

**PUC-Rio – Software Básico – INF1018**  
**Prova Final – 03/12/2010**

**Atenção:** Não esqueça de colocar o seu nome e número de matrícula na(s) folha(s) de respostas. Seja claro e objetivo. **Boa sorte!**

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++;
    }
}

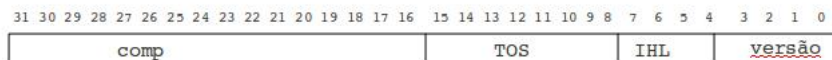
struct X {
    char c1;
    short s;
    char c2;
    float f;
    double d;
} x = {'g', -1024, (21 | 13) << 3, 0.825, -1.0};

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

Considerando que  $x$  seja armazenado no endereço de memória 0x2A000004, diga o que o programa irá imprimir quando executado, **explicando como você chegou a esses valores** (valores sem contas e explicações não serão considerados!).

Suponha que a máquina de execução é *little-endian* e que as convenções de alinhamento são as do Linux no IA-32. Escreva 'LL' nos bytes que contiverem lixo. O código ASCII de 'a' é 97.

2. (1,5 pontos) Considere um inteiro contendo os campos indicados abaixo:



Escreva uma função que receba um inteiro nesse formato e retorne a soma do valor do campo **comp** com o valor do campo **IHL**. Essa função deve ser escrita em assembly IA-32 utilizando as regras básicas de passagem de parâmetros e valor de retorno em C/Linux.

3. Traduza as funções `vals` e `foo` a seguir para assembly IA-32, utilizando as regras de alinhamento, passagem de parâmetros, valor de retorno e salvamento de registradores em C/Linux.

**Comente o seu código!**

- (a) (3,0 pontos)

```
struct T {
    char val;
    struct T *left;
    struct T *right;
};

int vals(struct T *t, char *v, int pos) {
    int newpos;

    if (t == NULL) return pos;

    newpos = vals(t->left, v, pos);
    v[newpos++] = t->val;
    return vals(t->right, v, newpos);
}
```

- (b) (3,0 pontos)

```
double fat (double d);

double foo (int n) {
    int i; double res = 0.0;
    for (i=0; i<=n; i++)
        res = res + 1.0 / fat ((double) i);
    return res;
}
```