

PUC-Rio – Software Básico – INF1612
Prova 2 – 27/11/04

1. (3,0 pontos) Considere o trecho de código C a seguir:

```
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;
}
```

Traduza a função `foo` para assembly IA-32 (o assembly visto no curso).

2. (3,0 pontos) Escreva uma função `mywrite` em assembly IA-32 que receba um descritor de arquivo e uma string (array de caracteres terminado em `'\0'`), e que escreva a string no arquivo:

```
int mywrite (int d, char *s);
```

Essa função não deve chamar nenhuma função da biblioteca C para fazer a escrita. Ela deve chamar diretamente o serviço `write` do Sistema Operacional. Segue uma descrição resumida e simplificada da página `man` da função `write` da biblioteca C:

```
int
write(int d, const void *buf, int nbytes);
DESCRIPTION
    Write() attempts to write nbytes of data to the object referenced by the
    descriptor d from the buffer pointed to by buf.
RETURN VALUES
    Upon successful completion the number of bytes which were written is
    returned.
```

Para construir a chamada ao sistema, lembre-se que o número da chamada é colocado em `%eax` e os argumentos, na ordem em que aparecem no protótipo, nos registradores `%ebx`, `%ecx`, `%edx`, `%esi` e `%edi`. O número da chamada `write` é 4. A interrupção que aciona o sistema operacional é a `0x80`.

3. (2,0 pontos) Liste todos os símbolos exportados (definidos nesse arquivo) e importados (esperados de outros arquivos) pelo módulo abaixo, ou seja, o que aparecerá como D (dados) ou T (texto, ou código), na primeira categoria, e, na segunda categoria, o que aparecerá como U (undefined), na listagem do `nm`.

```
extern float b;
static int c;
int a = 3;

int magica (char *x);

static void foo (int p) {
    float f = 0.0;
    b = 1.0 + p + f;
}

int bar (char *p1) {
    return magica (p1);
}
```

4. (2,0 pontos) Considere o seguinte código C e, em seguida, o código e endereços gerados pelo compilador:

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

1          fat:
2 0000 55          pushl   %ebp
3 0001 89E5        movl    %esp, %ebp
4 0003 8B4D08      movl    8(%ebp), %ecx
5 0006 83F900      cmpl    $0, %ecx
6 0009 7507        jne     .L6
7 000b B8010000    movl    $1, %eax
7          00
8 0010 EB12        jmp     ret
9          .L6:
10 0012 83E901     subl    $1, %ecx
11 0015 51         pushl   %ecx
12 0016 E8E5FFFF   call    fat
12          FF
13 001b 83C404     add     $4, %esp
14 001e 8B4D08      movl    8(%ebp), %ecx
15 0021 0FAFC1     imull   %ecx, %eax
16          fim:
17 0024 5D         pop     %ebp
18 0025 C3         ret
```

Suponha que a função `fat` foi chamada a partir de uma outra função qualquer e que no momento que o controle foi para `fat` pela primeira vez o endereço do topo da pilha era `0x10F0`.

Faça um esquema (desenho e valores) da pilha de execução no momento em que a recursão chegar ao caso base, isto é, na execução da instrução na linha 7 do assembler quando a função `fat` for chamada com argumento 0. Explícite na pilha os valores exatos de argumentos, `%ebp` e de endereços de retorno sempre que forem conhecidos. Quando os valores não puderem ser conhecidos coloque pontos de interrogação.