PUC-Rio – Software Básico – INF1018
Prova 2 – 29/11/2012 – Turma 3WB

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 trecho de código C a seguir:

```
int foo(int i, double d, float f);
int bar(...) {
  int i,j;
  i = foo(-1025, 2.0, -3.75);
  j = i * 2;
  ...
}
```

   Suponha que no momento em que o controle chegar ao label `foo` (ou seja, imediatamente após a execução de `call foo`), o valor de `%esp` seja `0xffffd2a0` e que o endereço da instrução seguinte à chamada de `foo` (correspondente ao código de `j = i * 2`) seja `0x080483e0`. Mostre o conteúdo da memória nesse momento, do endereço `0xffffd2a0` ao endereço `0xffffd2b3`. Esse conteúdo deve ser exibido, byte a byte, em hexadecimal. Para cada byte exibido indique seu endereço, também em hexadecimal.

2. (3,0 pontos) Traduza a função `boo` abaixo para assembly IA-32 do gcc/Linux (visto em sala), utilizando as regras básicas de alinhamento, passagem de parâmetros, retorno de resultado e uso de registradores em C. **Comente o seu código!**

   (Não se preocupe se você não entender o que a função faz, apenas traduza-a literalmente).

```
double foo(int a, double b);

struct X {
  char vc;
  int  vi;
  double vd;
  struct X *next;
}

double boo(struct X *px, float f) {
 double mul = 1.0;
 while (px != NULL) {
   mul *= foo(px->vi, px->vd + f);
   px = px->next;
 }
 return mul;
}
```

3. (2,5 pontos) Escreva em assembly IA-32 (convenções gcc/Linux) uma função `writeDoubles`, que recebe um descritor de arquivo, um array de `doubles` e o tamanho (número de elementos) deste array e escreve nesse arquivo o conteúdo do array. Essa função tem portanto o seguinte protótipo:

```
int writeDoubles(int d, double a[], int n);
```

   Sua função deve chamar diretamente o serviço `write` do Sistema Operacional para fazer a escrita. Esse serviço recebe 3 parâmetros:

- o descritor do arquivo (um inteiro)
- o endereço do buffer a ser escrito
- o número de bytes a escrever (um inteiro)

Para construir a chamada ao sistema, lembre-se que o código da chamada ao SO é passado em `%eax` e os demais parâmetros em `%ebx`, `%ecx`, `%edx`, etc... A interrupção que aciona o sistema operacional é a `0x80`. O código da chamada `write` é **4**.

4. (2,0 pontos) Considere o módulo abaixo:

```
#include <string.h>
#include <stdlib.h>

extern int i;
char *buff = NULL;
static int k = 0;
int f(char *p);

int g(char *s1, char *s2) {
  int a = strlen(s1);
  int b = strlen(s2);
  buff = (char *) malloc(a + b + 1);
  strcat(strcpy(buff,s1), s2);

  k += a + b + i;
  return f(buff);
}
```

(a) Liste os símbolos exportados (definidos) por este módulo e os símbolos importados por ele (referências externas).

(b) Considere agora o módulo abaixo:

```
#include <stdio.h>

int g(char *s1, char *s2);

int i = 0;
int k = 0;
char s1[] = "ola";
char s2[] = " mundo";

int f(char *p) {
  ...  /* algum código */
}

int main() {
  int j;
  ...
  j = g(s1, s2);
  printf("%d",k);
  ...
 }
```

Se esse módulo fosse "ligado" com o módulo anterior para compor um executável, qual seria o resultado da chamada a `printf`?