PUC-Rio – Software Básico – INF1018
Prova 2 – Turma 3wa – 26/11/2015

1. (1,0 ponto) 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 {
  unsigned char c;
  double e;
} x = {255, -0.625};
int main(void) {
  dump(&x, sizeof(x));
  return 0;
}
```

   Supondo que `x` seja alocada na posição de memória 0x0828a244, diga o que esse programa irá imprimir quando executado. Coloque **PP** nas posições correspondentes a *padding*. Considere que a máquina de execução é *little-endian* e que as convenções de alinhamento são as do Linux no IA-32 (vistas em sala). *Mostre como você chegou aos valores exibidos. Valores sem contas* **NÃO** *valem ponto!.*

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, salvamento de registradores e resultados em C/linux. Traduza o mais diretamente possível o código de C para assembly. Não se preocupe em entender o que as funções fazem, apenas traduza-as literalmente.

   Comente seu código! Use variáveis locais apenas quando estritamente necessário!

   (a) (2,5 pontos)

```
double f (double a, double b, double c);

double foo (double *a, float val, int n) {
  int i;
  double res = 0.0;
  for (i=0;i<n;i++) {
    res += f (*a, val, n);
    a++;
  }
  return res;
}
```

   (b) (2,5 pontos)

```
int bar (int *f);

int horrivel (void) {
  int vals[5];
  int *v = vals;
  int i;
  do {
    i = bar(v);
    v++;
  }
  while (i!=0);
  return v-vals-1;
}
```

3. Considere os módulos C `mod1.c` e `mod2.c` a seguir, definidos cada um em um arquivo:

```
/**********/
/* mod1.c */
#include <stdio.h>

extern double a;
int chamadas = 0;

static void  boo () {
  a = -2050;
  chamadas++;
  printf ("chamadas = %d\n", chamadas);
}

void foo () {
  boo ();
}
```

```
/**********/
/* mod2.c */
#include <stdio.h>

int foo ();

unsigned int a[2] = {-1, -1};;

int main (void) {
  printf("a = %08x %08x\n", a[0], a[1]);
  foo();
  printf("a = %08x %08x\n", a[0], a[1]);
  foo();
  return 0;
}
```

   (a) (1,0 pontos) Suponha que estes módulos sejam compilados com `gcc -Wall -m32 -c mod1.c` e `gcc -Wall -m32 -c mod2.c`, gerando arquivos objeto `mod1.o` e `mod2.o`. Se inspecionássemos a tabela de símbolos de cada um desses arquivos objeto com o programa `nm`, que símbolos apareceriam como **D** (símbolo na área de dados, exportado), **T** (símbolo na área de código, exportado) e **U** (símbolo indefinido)?

   (b) (1,0 pontos) Suponha agora que criemos um arquivo a partir dos dois módulos, com

   `gcc -Wall -m32 -o meuprograma mod1.o mod2.o`.

   Descreva o que será impresso, explicando por que.

4. (1,0 ponto) Suponha que determinada função termina com o código a seguir:

```
movl $0xfafbfcf8, 4(%ebp)
movl $0x11223344, 8(%ebp)
movl $0x01020304, 12(%ebp)
movl %ebp, %esp
pop %ebp
ret
```

   Neste caso, para que endereço a função irá retornar depois do ret? (isto é, em que endereço estará a próxima intrução a ser executada depois do `ret`?

Boa Prova!