PUC-Rio – Software Básico – INF1018
Prova 2 – 10/6/2014

1. (2,5 pontos) No trabalho 2, vocês tiveram que desenvolver um módulo com a interface a seguir.

```
typedef enum {INT_PAR, CHAR_PAR, DOUBLE_PAR, PTR_PAR} TipoParam;
typedef struct {
  TipoParam tipo;  /* indica o tipo do parâmetro */
  int amarrado;    /* indica se o parâmetro deve ter um valor amarrado */
  int posicao; /* indica em que posição esse parâmetro vai aparecer na chamada */
  union {
    int v_int;
    char v_char;
    double v_double;
    void* v_ptr;
  } valor;         /* define o valor do parâmetro se este for amarrado */
} Parametro;
void* gera_func (void* f, int n, Parametro params[]);
void libera_func (void* func);
```

   Suponha o trecho de código abaixo que usa *gera_func* para criar dinamicamente uma nova função. Escreva em *assembly* (**atenção! em assembly, não código de máquina!**) o código equivalente ao gerado por *gera_func* para esta nova função.

```
#include "gerafunc.h"
#include <stdio.h>
int foobar (int a, double b, int c);
typedef int (*func_ptr) (double x, int m);

int main (void) {
  Parametro params[3];
  func_ptr f_f = NULL;
  params[0].tipo = INT_PAR;
  params[0].amarrado = 0;
  params[0].posicao = 2;
  params[1].tipo = DOUBLE_PAR;
  params[1].amarrado = 0;
  params[1].posicao = 1;
  params[2].tipo = INT_PAR;
  params[2].amarrado = 1;
  params[2].valor.v_int = 1;

  f_f = (func_ptr) gera_func (foobar, 3, params);
  printf("%d\n", (*f_f)(2.0, -3));
  libera_func(f_f);

  return 0;
}
```

2. (2,0 pontos) Considere o programa formado pelos seguintes arquivos:

   - arquivo `mod1.c`:

```
int mesma = 0;
void bar (int a);
int foo (void) {
  bar(mesma);
  return 1;
}
```

- arquivo `prog.c`:

```
#include <stdio.h>
int mesma = 0;
int foo (void);
int main (void) {
  printf ("%d\n", foo ());
  return 0;
}
```

Suponha que os arquivos `mod1.c` e `prog.c` sejam compilados com *gcc -Wall -m32 -o meuprog mod1.c prog.c*. Em que passo essa geração falharia (pré-processador, compilador, montador, ligador)? Indique o(s) erro(s) sinalizado(s) por esse passo.

3. (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 {
  float f1;
  float f2;
  short s;
} x = {.25, -520.0, -17};
int main(void) {
  dump((void *)&x, sizeof(x));
  return 0;
}
```

Supondo que x seja armazenado no endereço de memória 0x80500e4, diga o que o programa irá imprimir quando executado, deixando claro como você chegou a esses valores. Considere que a máquina de execução é *little-endian*, e que as convenções de alinhamento são as do Linux no IA-32. Se houver posições de *padding*, indique seu conteúdo com **pp**. (ATENÇÃO: valores sem contas e explicações **NÃO** valem ponto!)

4. (3,0 pontos) Traduza a função `foo` abaixo para assembly IA-32 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros e resultados de C/Linux/IA-32. Traduza o mais diretamente possível o código de C para assembly. Comente seu código.

```
#include <stdio.h>
double f(double x, int *pi);
double foo (float a[], int n) {
  float *pfloat;
  double res;
  int status;
  double sum = 0.0;
  for (pfloat = a; n-- ; pfloat++) {
    res = f((double)*pfloat, &status);
    if (status) sum += res;
  }
  return sum;
}
```