PUC-Rio – Software Básico – INF1018
Prova 2 – 28/11/13

1. (2,0 pontos) Considere o programa formado pelos seguintes arquivos:

   - arquivo `mod1.h`:

   ```
   extern int limite;
   int vmaior(int *v, int n);
   ```

   - arquivo `mod1.c`:

   ```
   #include "mod1.h"
   int limite = -1;
   int vmaior(int *v, int n) {
     int i, m = limite;
     for (i = 0; i < n; i++)
       if (v[i] > m) m = v[i];
     return m;
   }
   ```

   - arquivo `prog.c`:

   ```
   #include <stdio.h>
   #include "mod1.h"

   #define TAM 10
   int limite = -1;

   int main() {
     int i,v;
     int valores[TAM];
     for (i = 0; i < TAM; i++) {
       printf("proximo valor:");
       scanf("%d",&v);
       valores[i] = (v < 0) ? limite : v;
     }
     printf("\nmaior: %d\n", vmaior(valores, TAM));
     return 0;
   }
   ```

   (a) Supondo que os arquivos `mod1.c` e `prog.c` sejam compilados em separado, liste para cada um dos arquivos objeto gerados quais símbolos apareceriam como **D** (símbolo na área de dados, exportado), **T** (símbolo na área de código, exportado) e **U** (símbolo indefinido) na saída do programa `nm`.

   (b) Caso o *linkeditor* seja invocado com o comando abaixo para gerar o programa executável a partir dos arquivos objetos

   ```
   gcc -o prog mod1.o prog.o
   ```

   o arquivo executável `prog` seria gerado? Por que?

2. (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 {
  double d;
  float  f;
  short  s;
} x = {-4.5, 9.75, -11};

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

Considerando que `x` seja alocado na posição de memória 0x804a018, mostre o que esse programa irá imprimir quando executado, explicando como você chegou aos valores exibidos (ATENÇÃO: valores sem contas **NÃO valem ponto!**). Coloque PP nas posições correspondentes a *padding*. Suponha que a máquina de execução é *little-endian* e que as convenções de alinhamento são as do Linux no IA-32.

3. (3,5 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.

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

```
double boo (double a, float *b);

void bar (float *f, int num);

struct X {
  float f1;
  double d1;
};

float foo (struct X *x, int n) {
  double res = 0.0;
  float locais[2];
  struct X *p = x;
  int contador;
  bar (locais, 2);
  for (contador=0; contador<n; contador++) {
    res += boo((double)p->f1, locais);
    p++;
  }
  return res;
}
```

4. (1,0 ponto) Considerando programa abaixo, que utiliza a biblioteca de *co-rotinas assimétricas* que vimos no curso,

```
#include <stdio.h>
#include <values.h>
#include "corotinas.h"
#define MAX 5

int numeros[] = {1, 2, 3, 4, 5};

int ping(int i) {
  int contador;
  for (contador = 0; contador<MAX; contador++) {
    printf ("em ping: %d\n", i);
    i = coro_yield(numeros[contador]);
  }
  return -MAXINT;
}

int main() {
  CoroDescrP cping1 = coro_create("ping1", ping);
  CoroDescrP cping2 = coro_create("ping2", ping);
  printf ("%d\n", coro_resume(cping1, 1));
  printf ("%d\n", coro_resume(cping2, 2));
  printf ("%d\n", coro_resume(cping2, 2));
  printf ("%d\n", coro_resume(cping2, 2));
  printf ("%d\n", coro_resume(cping1, 1));
  coro_free(cping1);
  coro_free(cping2);
  return 0;
}
```

   mostre o que o programa irá imprimir quando executado.

5. (1,0 pontos) Escreva em C uma função `maiorque1` que, **sem usar operações de ponto flutuante** (isto é, operando diretamente sobre a representação binária) retorne 1 se o valor do seu parâmetro *float* é **maior que** 1.0 e 0 caso contrário:

```
int maiorque1(float f);
```

   Para acessar a representação binária de um valor *float* você pode usar a `union` que vimos em um dos laboratórios:

```
typedef union {
  float f;
  unsigned int i;
} U;
```