PUC-Rio – Software Básico – INF1018
Prova 1 – 24/09/2011

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. (2,0 pontos) Considere que uma chamada à função `gravacomp` do primeiro trabalho, executada em uma máquina de 32 bits, *little-endian*, gerou um arquivo binário com o seguinte conteúdo (em hexadecimal):

```
05 31 00 00 aa bb cc 00 dd 00 ff aa bb cc 01 dd
01 fe aa bb cc 02 dd 02 fd aa bb cc 03 dd 03 fc
aa bb cc 04 dd 04
```

   Esse arquivo binário tem o seguinte formato:

   - o primeiro byte do arquivo indica o número de structs armazenadas
   - a seguir aparecem os descritores de campos dos structs, codificados da seguinte forma:
     - 001 - char
     - 010 - short int
     - 100 - int
     - 101 - ponteiro
     - 000 - final da lista de campos
   - após os descritores aparecem os bytes com os dados do array de structs, armazenados no arquivo em *big-endian* e de forma compactada (sem *padding*).

   Conhecendo o formato do arquivo, e observando o seu conteúdo:

   (a) Forneça uma declaração para o array de estruturas armazenado no arquivo, preenchendo as lacunas da declaração abaixo:

```
struct s {
  /* declaração de campos */
  ...
};
struct s structs[...];
```

   (b) Mostre byte a byte a representação do array de structs na memória da máquina especificada (32 bits, *little-endian*). Represente com `'pp'` os bytes de *padding*.

2. Traduza as funções `foo` e `find` a seguir para assembly IA-32 do gcc/Linux (visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros, salvamento de registradores e retorno de resultado em C. **Não esqueça de comentar o seu código!**

   (Não se preocupe se você não entender o que as funções fazem, apenas traduza-as literalmente).

   (a) (3,0 pontos)

```
int f(int i);
int g(int i, int j);

int foo(int a[], int n) {
  int i, val;
  int acc = 0;
  for (i = 0; i < n; i++) {
    val = a[i];
    acc += g(val, f(val));
    a++;
  }
  return acc;
}
```

(b) (3,0 pontos)

```
struct Y {
  char c;
  int  v;
};

struct X {
  char k;
  struct Y *py;
  struct X *next;
};

struct Y *find(struct X *px, char key) {
  while (px != NULL) {
    if (px->k == key)
      return px->py;
    px = px->next;
  }
  return NULL;
}
```

3. (2,0 pontos) Para cada um dos itens a seguir indique a saída correspondente:

(a)

```
short s1 = -3;
unsigned short s2 = 0xffff;

unsigned int i1 = s1;
int i2 = s2;

printf("\n i1 = %08x i2 = %08x\n", i1, i2);
```

Observação: o formato `%08x` especifica a saída de pelo menos 8 'dígitos' hexadecimais, com preenchimento de zeros à esquerda.

(b)

```
int i1 = 1040;
int i2 = -i1;
int i3 = i1 | i2;
int i4 = (i3 << 3) & i1;

printf("\n i1 = %08x i2 = %08x i3 = %d i4 = %d\n", i1, i2, i3, i4);
```

ATENÇÃO: note que a saída dos dois primeiros valores é em **notação hexadecimal**, e a dos outros dois em **notação decimal**!