PUC-Rio – Software Básico – INF1018
Prova 2 – Turmas 3WA e 3WB – 24/11/2022

1. (3,0 pontos) No formato IEEE 754, números com um ponto flutuante são representados através de um sinal, um expoente e uma parte fracionária de 1, 8 e 23 bits ou 1, 11 e 52 bits para números do tipo float e double respectivamente. Considere um novo padrão de números reais de 16 bits que utiliza o mesmo algoritmo de codificação do que o formato IEEE 754, porém os campos sinal, expoente e fração possuem 1, 5, 10 bits respectivamente. Escreva em hexadecimal os seguintes números decimais neste novo padrão de 16 bits, justificando o resultado com as contas necessárias.

   (a) 513,0

   (b) 0,6

   ATENÇÃO: Resultados sem justificativas **NÃO** serão considerados.

2. (2,0 pontos) Considere o disassembly obtido abaixo através do programa objdump com opção –d como feito nos laboratórios e no trabalho:

```
0000000000001129 <main>:
    1129:       55                      push   %rbp
    112a:       48 89 e5                mov    %rsp,%rbp
    112d:       48 83 ec 10             sub    $0x10,%rsp
    1131:       c7 45 f4 08 00 00 00    movl   $0x8,-0xc(%rbp)
    1138:       c7 45 f8 07 00 00 00    movl   $0x7,-0x8(%rbp)
    113f:       8b 55 f8                mov    -0x8(%rbp),%edx
    1142:       8b 45 f4                mov    -0xc(%rbp),%eax
    1145:       89 d6                   mov    %edx,%esi
    1147:       89 c7                   mov    %eax,%edi
    1149:       e8 ?? ?? ?? ??          callq  1168 <transmogrifa>
    114e:       89 45 fc                mov    %eax,-0x4(%rbp)
    1151:       8b 55 fc                mov    -0x4(%rbp),%edx
    1154:       8b 45 f4                mov    -0xc(%rbp),%eax
    1157:       89 d6                   mov    %edx,%esi
    1159:       89 c7                   mov    %eax,%edi
    115b:       e8 ?? ?? ?? ??          callq  1168 <transmogrifa>
    1160:       89 45 fc                mov    %eax,-0x4(%rbp)
    1163:       8b 45 fc                mov    -0x4(%rbp),%eax
    1166:       c9                      leaveq
    1167:       c3                      retq

0000000000001168 <transmogrifa>:
    1168:       55                      push   %rbp
    1169:       48 89 e5                mov    %rsp,%rbp
    116c:       89 f0                   mov    %esi,%eax
    116e:       01 f8                   add    %edi,%eax
    1170:       5d                      pop    %rbp
    1171:       c3                      retq
```

   Considerando que a máquina de execução seja little-endian com as convenções de alinhamento e chamada do Linux no IA-64 vistas em sala, determine os valores correspondentes às lacunas (`??`) acima. Você deve determinar byte a byte em hexadecimal o valor das posições `114A` a `114D` e `115C` a `114F`.
   Resultados sem justificativas não serão considerados.

3. Traduza as funções `boba` e `foo` abaixo para assembly IA-64 (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.

   Comente seu código!

   (a) (1,5 pontos)

```
float boba (float v, float li) {
   return v + li;
}
```

   (b) (3,5 pontos)

```
#define TAM 5
struct S { short s; float f; };

double foo (struct S st[], float lim) {
   int i;
   double acc = 0.0;

   for (i=0; i<TAM; i++) {
      st[i].s = i;
      acc += boba(st[i].f, lim);
   }

   return acc;
}
```

Boa Prova!