Nome: _____

PUC-Rio – Software Básico – INF1018
Prova 2 – Turma 3WC – 18/06/2024

1. (2,5 pontos) Supondo que `x` seja armazenado no endereço de memória **0x50A22010**, mostre 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-64 e que o formato de ponto flutuante é IEEE-754. A posição do caractere 'A' na tabela ASCII é **0x41**. Se houver posições de *padding*, indique seu conteúdo com **PP**. (ATENÇÃO: valores sem contas e explicações NÃO valem ponto!)

```
#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 {
      short s;
      float f;
      char c;
      double d;
} x = {-10, 'K', -1.75, -127.25};

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

2. (2,5 pontos) Traduza a função `boba` a seguir para assembly IA-64, utilizando as regras usuais de alinhamento, passagem de parâmetros, salvamento de registradores e resultados em C/Linux.

```
int calcula(float f);
double boba (float val) {
      int i;
      int pos;
      double d[5];
      for (i=0; i<5; i++) {
            pos = calcula(val);
            d[pos] += val;
      }
      return d[0];
}
```

3. (2,5 pontos) Considere o seguinte código **prog1.c**:

```
#include <stdio.h>
extern short s;
void f1(void) {
      s++;
      printf ("s = %hd\n", s);
}

void f2(void) {
      f1();
}
```

e o código de **prog2.c** seguinte:

```
#include <stdio.h>
void f2(void);
short s = 97;
int main(void) {
      printf("s = %08hx\n", s);
      f2();
      return 0;
}
```

a) Considere a compilação em separado dos dois códigos acima com os comandos

```
gcc -Wall -c prog1.c
```

e

```
gcc -Wall -c prog2.c
```

Ao examinarmos o módulo objeto gerado com o comando `nm`, liste para cada módulo quais símbolos aparecem como **T** (símbolo na área de código exportado), **t** (símbolo na área de código local), **D** (símbolo na área de dados exportado), **d** (símbolo na área de dados local), e **U** (símbolo indefinido).

b) Diga o que será impresso pelo programa ao compilarmos os dois módulos conjuntamente com o seguinte comando e depois o executarmos:

```
gcc -Wall -o prog prog1.c prog2.c
prog
```

Justifique sua resposta.

4. (2,5 pontos) Considere o disassembly obtido abaixo através do programa `objdump` com opção **-d** como feito nos laboratórios:

```
0000000000001129 <f1>:
    1129:       55                      push   %rbp
    112a:       48 89 e5                mov    %rsp,%rbp
    112d:       89 7d fc                mov    %edi,-0x4(%rbp)
    1130:       89 75 f8                mov    %esi,-0x8(%rbp)
    1133:       8b 55 fc                mov    -0x4(%rbp),%edx
    1136:       8b 45 f8                mov    -0x8(%rbp),%eax
    1139:       01 d0                   add    %edx,%eax
    113b:       5d                      pop    %rbp
    113c:       c3                      ret

000000000000213d <main>:
    213d:       55                      push   %rbp
    213e:       48 89 e5                mov    %rsp,%rbp
    2141:       48 83 ec 10             sub    $0x10,%rsp
    2145:       c7 45 f4 01 00 00 00    movl   $0x1,-0xc(%rbp)
    214c:       c7 45 f8 02 00 00 00    movl   $0x2,-0x8(%rbp)
    2153:       8b 55 f8                mov    -0x8(%rbp),%edx
    2156:       8b 45 f4                mov    -0xc(%rbp),%eax
    2159:       89 d6                   mov    %edx,%esi
    215b:       89 c7                   mov    %eax,%edi
    215d:       e8 ?? ?? ?? ??          call   1129 <f1>
    2162:       be 03 00 00 00          mov    $0x3,%esi
    2167:       89 c7                   mov    %eax,%edi
    2169:       b8 00 00 00 00          mov    $0x0,%eax
    216e:       e8 ?? ?? ?? ??          call   117d <f2>
    2173:       89 45 fc                mov    %eax,-0x4(%rbp)
    2176:       b8 00 00 00 00          mov    $0x0,%eax
    217b:       c9                      leave
    217c:       c3                      ret

000000000000317d <f2>:
    317d:       55                      push   %rbp
    317e:       48 89 e5                mov    %rsp,%rbp
    3181:       89 7d fc                mov    %edi,-0x4(%rbp)
    3184:       89 75 f8                mov    %esi,-0x8(%rbp)
    3187:       89 55 f4                mov    %edx,-0xc(%rbp)
    318a:       8b 55 fc                mov    -0x4(%rbp),%edx
    318d:       8b 45 f8                mov    -0x8(%rbp),%eax
    3190:       01 c2                   add    %eax,%edx
    3192:       8b 45 f4                mov    -0xc(%rbp),%eax
    3195:       01 d0                   add    %edx,%eax
    3197:       5d                      pop    %rbp
    3198:       c3                      ret
```

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 **215E** a **2161** e **216F** a **2172**.

Resultados sem justificativas não serão considerados.