

Nome: \_\_\_\_\_

# PUC-Rio – Software Básico – INF1018

## Prova 2 – Turma 3WB – 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 {
    char c;
    float f;
    short s;
    double d;
} x = {'L', -1.25, -10, -127.75};

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 = 101;

int main(void) {
    printf("s = %08hx\n", s);
    f2();
    return 0;
}
```

c) 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).

d) 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:

```
00000000000001129 <f1>:
1129:      55                push    %rbp
112a:    48 89 e5          mov     %rsp,%rbp
112d:    48 83 ec 10       sub     $0x10,%rsp
1131:    89 7d fc          mov     %edi,-0x4(%rbp)
1134:    89 75 f8          mov     %esi,-0x8(%rbp)
1137:    8b 4d f8          mov     -0x8(%rbp),%ecx
113a:    8b 45 fc          mov     -0x4(%rbp),%eax
113d:    ba 03 00 00 00   mov     $0x3,%edx
1142:    89 ce            mov     %ecx,%esi
1144:    89 c7            mov     %eax,%edi
1146:    b8 00 00 00 00   mov     $0x0,%eax
114b:    e8 ?? ?? ?? ??   call    1181 <f2>
1150:    c9              leave   %eax
1151:    c3              ret

00000000000002152 <main>:
2152:      55                push    %rbp
2153:    48 89 e5          mov     %rsp,%rbp
2156:    48 83 ec 10       sub     $0x10,%rsp
215a:    c7 45 f4 01 00 00 00 movl    $0x1,-0xc(%rbp)
2161:    c7 45 f8 02 00 00 00 movl    $0x2,-0x8(%rbp)
2168:    8b 55 f8          mov     -0x8(%rbp),%edx
216b:    8b 45 f4          mov     -0xc(%rbp),%eax
216e:    89 d6            mov     %edx,%esi
2170:    89 c7            mov     %eax,%edi
2172:    e8 ?? ?? ?? ??   call    1129 <f1>
2177:    89 45 fc          mov     %eax,-0x4(%rbp)
217a:    b8 00 00 00 00   mov     $0x0,%eax
217f:    c9              leave   %eax
2180:    c3              ret

00000000000003181 <f2>:
3181:      55                push    %rbp
3182:    48 89 e5          mov     %rsp,%rbp
3185:    89 7d fc          mov     %edi,-0x4(%rbp)
3188:    89 75 f8          mov     %esi,-0x8(%rbp)
318b:    89 55 f4          mov     %edx,-0xc(%rbp)
318e:    8b 55 fc          mov     -0x4(%rbp),%edx
3191:    8b 45 f8          mov     -0x8(%rbp),%eax
3194:    01 c2            add     %eax,%edx
3196:    8b 45 f4          mov     -0xc(%rbp),%eax
3199:    01 d0            add     %edx,%eax
319b:    5d              pop     %rbp
319c:    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 **114C** a **114F** e **2173** a **2176**.

Resultados sem justificativas não serão considerados.