PUC-Rio – Software Básico – INF1018

Prova 2 – Turma 3WB

Q1	
Q2	
Q3	

Nome: _____

1) (3,0 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 S {
   short a;
   float b;
   char c;
   double d;
} s = {12, 12.125, -12, 12.125};

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

Mostre o que este programa irá imprimir quando executado, justificando os valores exibidos mostrando as contas e outras informações usadas para chegar ao resultado. Suponha que a máquina de execução seja *little-endian* com as convenções de alinhamento do Linux no IA-64 vistas em sala. Além disto, considere que s seja alocado na posição de memória **0x7ffee4bb4260**. Coloque **PP** nas posições correspondentes a *padding*.

2) (5,0 pontos) Traduza a função a seguir 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!

```
float passa_alta(short *v, int limite);
int foo(short *v, int n, long filtro) {
    int i;
    double soma = 0;
    for(i=0; i<n; i++) {
        if(v[i] > filtro) {
            soma += passa_alta(v[i], 3);
        }
    }
}
```

3) (2,0 pontos) Considere o *disassembly* obtido abaixo através do programa **objdump** com opção **–d** como feito nos laboratórios:

```
0000000000001139 <main>:
    1139:       55                      push   %rbp
    113a:       48 89 e5                mov    %rsp,%rbp
    113d:       48 83 ec 10             sub    $0x10,%rsp
    1141:       c7 45 f4 08 00 00 00    movl   $0x8,-0xc(%rbp)
    1148:       c7 45 f8 07 00 00 00    movl   $0x7,-0x8(%rbp)
    114f:       8b 55 f8                mov    -0x8(%rbp),%edx
    1152:       8b 45 f4                mov    -0xc(%rbp),%eax
    1155:       89 d6                   mov    %edx,%esi
    1157:       89 c7                   mov    %eax,%edi
    1159:       e8 ?? ?? ?? ??          call   <transmogrifa>
    115e:       89 45 fc                mov    %eax,-0x4(%rbp)
    1161:       8b 45 fc                mov    -0x4(%rbp),%eax
    1164:       c9                      leave
    1165:       c3                      ret

0000000000001166 <destransmogrifa>:
    1166:       55                      push   %rbp
    1167:       48 89 e5                mov    %rsp,%rbp
    116a:       48 8d 05 9f 2e 00 00    lea    0x2e9f(%rip),%rax
    1171:       48 89 c7                mov    %rax,%rdi
    1174:       e8 b7 fe ff ff          call   1030 <puts@plt>
    1179:       5d                      pop    %rbp
    117a:       c3                      ret

000000000000117b <transmogrifa>:
    117b:       55                      push   %rbp
    117c:       48 89 e5                mov    %rsp,%rbp
    117f:       89 7d fc                mov    %edi,-0x4(%rbp)
    1182:       89 75 f8                mov    %esi,-0x8(%rbp)
    1185:       e8 ?? ?? ?? ??          call   <destransmogrifa>
    118a:       8b 55 fc                mov    -0x4(%rbp),%edx
    118d:       8b 45 f8                mov    -0x8(%rbp),%eax
    1190:       01 d0                   add    %edx,%eax
    1192:       5d                      pop    %rbp
    1193:       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 ocupadas por **??** (listar 8 pares de endereço-valor).

Resultados sem justificativas não serão considerados.

Boa Prova!

PUC-Rio – Software Básico – INF1018

Prova 2 – Turma 3WC

Q1	
Q2	
Q3	

Nome: _____

1) (3,0 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 S {
   short a;
   double b;
   char c;
   float d;
} s = {-25, 25.25, 25, 25.25};

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

Mostre o que este programa irá imprimir quando executado, justificando os valores exibidos mostrando as contas e outras informações usadas para chegar ao resultado. Suponha que a máquina de execução seja *little-endian* com as convenções de alinhamento do Linux no IA-64 vistas em sala. Além disto, considere que s seja alocado na posição de memória **0x7ffee4bb4260**. Coloque **PP** nas posições correspondentes a *padding*.

2) (5,0 pontos) Traduza a função a seguir 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!

```
double passa_alta(short *v, int limite);
int foo(short *v, int n, long filtro) {
    int i;
    float soma = 0;
    for(i=0; i<n; i++) {
        if(v[i] > filtro) {
            soma += passa_alta(v[i], 3);
        }
    }
}
```

3) (2,0 pontos) Considere o *disassembly* obtido abaixo através do programa **objdump** com opção **–d** como feito nos laboratórios:

```
0000000000001139 <destransmogrifa>:
    1139:       55                      push   %rbp
    113a:       48 89 e5                mov    %rsp,%rbp
    113d:       48 8d 05 cc 2e 00 00    lea    0x2ecc(%rip),%rax
    1144:       48 89 c7                mov    %rax,%rdi
    1147:       e8 e4 fe ff ff          call   1030 <puts@plt>
    114c:       5d                      pop    %rbp
    114d:       c3                      ret

000000000000114e <main>:
    114e:       55                      push   %rbp
    114f:       48 89 e5                mov    %rsp,%rbp
    1152:       48 83 ec 10             sub    $0x10,%rsp
    1156:       c7 45 f4 08 00 00 00    movl   $0x8,-0xc(%rbp)
    115d:       c7 45 f8 07 00 00 00    movl   $0x7,-0x8(%rbp)
    1164:       8b 55 f8                mov    -0x8(%rbp),%edx
    1167:       8b 45 f4                mov    -0xc(%rbp),%eax
    116a:       89 d6                   mov    %edx,%esi
    116c:       89 c7                   mov    %eax,%edi
    116e:       e8 ?? ?? ?? ??          call   <transmogrifa>
    1173:       89 45 fc                mov    %eax,-0x4(%rbp)
    1176:       e8 ?? ?? ?? ??          call   <destransmogrifa>
    117b:       8b 45 fc                mov    -0x4(%rbp),%eax
    117e:       c9                      leave
    117f:       c3                      ret

0000000000001180 <transmogrifa>:
    1180:       55                      push   %rbp
    1181:       48 89 e5                mov    %rsp,%rbp
    1184:       89 7d fc                mov    %edi,-0x4(%rbp)
    1187:       89 75 f8                mov    %esi,-0x8(%rbp)
    118a:       8b 55 fc                mov    -0x4(%rbp),%edx
    118d:       8b 45 f8                mov    -0x8(%rbp),%eax
    1190:       01 d0                   add    %edx,%eax
    1192:       5d                      pop    %rbp
    1193:       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 ocupadas por **??** (listar 8 pares de endereço-valor).

Resultados sem justificativas não serão considerados.

Boa Prova!