PUC-Rio – Software Básico – INF1018
Prova 2 – Turma 3wb – 22/06/2017

1. Considere o programa C a seguir (as reticências na inicialização de s são propositais):

```
#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;
};
int main (void) {
 struct X x = {...,...,...};
 dump (&x, 16);
return 0;
}
```

   Ao ser executado em uma máquina Linux IA-64, *little-endian*, este programa imprimiu:

```
  0x7ffd5aadf9a0 - f2
  0x7ffd5aadf9a1 - fe
  0x7ffd5aadf9a2 - 24
  0x7ffd5aadf9a3 - 97
  0x7ffd5aadf9a4 - 00
  0x7ffd5aadf9a5 - 00
  0x7ffd5aadf9a6 - 78
  0x7ffd5aadf9a7 - c1
  0x7ffd5aadf9a8 - 00
  0x7ffd5aadf9a9 - 85
  0x7ffd5aadf9aa - 24
  0x7ffd5aadf9ab - 97
  0x7ffd5aadf9ac - 1c
  0x7ffd5aadf9ad - 56
  0x7ffd5aadf9ae - 00
  0x7ffd5aadf9af - 00
```

   (a) (2,0 pontos) Analisando a saída do programa e a declaração da estrutura, diga quais foram os valores atribuídos aos campos **s**, **f** e **c** na inicialização da estrutura **x**. **Expresse esses valores em decimal!** (Note que a função dump exibe 16 bytes. Esse tamanho é suficiente para mostrar todos os campos da estrutura, mas podem ser impressos bytes a mais.)

   (b) (0,5 pontos) Qual é o tamanho (sizeof) da estrutura **x**?

2. (1,0 ponto) Escreva o código *assembly* equivalente ao código de máquina que a função **compila** de seu trabalho geraria para a função *Simples* a seguir:

```
  v3 = p1 + $1
  v1 = v3 + p2
  ret
```

3. Traduza as funções `foo` e `boo` a seguir para assembly IA-64, utilizando as regras usuais de alinhamento, passagem de parâmetros, salvamento de registradores e resultados em C/Linux. (Não se preocupe em entender o que as funções fazem, apenas traduza-as literalmente.)

   (a) (2,5 pontos)

```
int fun1(int x);
double fun2(double x, double y);

float foo(double *vd, int n) {
  double d = 0.0;
  while (n--) {
    d = fun2(d, vd[fun1(n)]);
  }
  return (float) (d + 1);
}
```

   (b) (2,5 pontos)

```
#define TAM 4
double fun3(double *p, int n);
double boo(float *pf, float v, int n) {
  double local[TAM];
  int i;
  double *pd = local;
  for (i = 0; i < TAM; i++) {
    *pd = *pf - v;
    pd++;
    pf++;
  }
  if (n)
    return fun3(local, TAM);
  else
    return local[0];
}
```

4. Considere os módulos **mod1.c** e **mod2.s** a seguir:

mod1.c:

```
#include <stdio.h>
extern int v1;
int v2 = 0;
int foo(int x);

int main() {
  int prox;
  scanf("%d %d", &v2, &prox);
  foo(prox);
  printf("%d\n", v1);
  return 0;
}
```

mod2.s:

```
.globl v1, foo
.data
v1: .int -1
.text
foo:
  cmpl %edi, v1
  jge no
  movl %edi, v1
no:
  movl v1, %edi
  movq v2, %rsi
  call bar
  ret
```

   (a) (1,0 ponto) Suponha que compilemos o módulo **mod1.c** com o comando ***gcc -c mod1.c***. Se inspecionarmos a tabela de símbolos do arquivo objeto **mod1.o** com o programa `nm`, que símbolos aparecerão na saída como **D** ou **B** (símbolo na área de dados, exportado), **T** (símbolo na área de código, exportado) e **U** (referência externa)?

   (b) (0,5 ponto) Suponha que tentemos gerar um executável composto por esses dois módulos, executando ***gcc -o prog mod1.c mod2.s***. O executável seria gerado? Se não, qual o motivo da falha, e em que passo ocorreria (compilação, montagem, linkedição)?