PUC-Rio – Software Básico – INF1018
Prova 2 – Turma 3WA – 27/06/2019

1. (2,5 pontos)

```
#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 c1;
   float f1;
   double d1;
   short s1;
} x = {-0xA, 0.0625, -7.875, 260};

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

   Supondo que `x` seja armazenado no endereço de memória 0x601040, diga o que o programa irá imprimir quando executado, deixando claro como você chegou a esses valores, mostrando as contas e outras informações usadas. Considere que a máquina de execução é *little-endian* e que as convenções de alinhamento são as do Linux no IA-64. Se houver posições de *padding*, indique seu conteúdo com **PP**. (ATENÇÃO: valores sem contas e explicações **NÃO** valem ponto!)

2. (1,5 ponto) Considere o seguinte código prog1.c:

```
#include <stdio.h>
extern char a;

static void reg () {
  a++;
  printf ("a = %c\n", a);
}

void pode () {
  reg ();
}
```

   e o código de prog2.c seguinte:

```
#include <stdio.h>
int reg ();

int a = 97;

int main (void) {
  printf("a = %08x\n", a);
```

```
  reg ();
  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 acontece ao compilarmos os dois módulos conjuntamente com o comando seguinte
`gcc -Wall -o prog prog1.c prog2.c` Justifique sua resposta.

3. Traduza as funções `fooba` 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.

(a) (2,5 pontos)

```
double vaip (double x1, double x2);

double fooba (double *v1, double *v2, int n) {
   int i;
   double acc = 0.0;
   for (i=0; i<n; i++) {
      acc += vaip (*v1, *v2);
      v1++; v2++;
   }

   return acc;
}
```

(b) (2,5 pontos)

```
struct exam { double taxa; int classe; };

float fica (double val, int bias);

float boo (double *dd, int pat) {
   int i;
   struct exam temp[10];

   for (i=0; i<10; i++) {
      temp[i].taxa = dd[i];
      temp[i].classe = i;
   }
   return fica(temp[0].taxa, pat);
}
```

4. (1,0 ponto) Escreva o código assembly equivalente ao código de máquina que a função *gera* do segundo trabalho de software básico geraria para o trecho de código *Simples* a seguir:

```
v1 < p2
v1 < v1 + $1
ret v1
```

Boa Prova!