PUC-Rio – Software Básico – INF1018
Prova 2 – Turma 3wb – 26/11/2015

1. (2,5 pontos) Considere o seguinte módulo C (`prog.c`):

```
#include <stdio.h>

#define VALI -3
float valf = 9.125;
double vald = -255.0;

double minha_funcao(int p1, float p2, double p3);

int main(void) {
  double res;
  res = minha_funcao(VALI, valf, vald);
  printf("resultado %f\n",res);
  return 0;
}
```

   Suponha que imediatamente após a CPU executar a instrução `call minha_funcao` (porém antes da função começar a executar), o valor de `%esp` seja `0xfff64700`.

   Mostre, **byte a byte**, em notação hexadecimal, apenas o conteúdo da porção da pilha de execução **onde os parâmetros estão armazenados**, começando no endereço mais baixo. Indique para cada byte o endereço de memória correspondente (também em hexadecimal).

   Considere que a máquina de execução é *little-endian* e que as convenções de alinhamento são as do Linux no IA-32 (vistas em sala).

   (ATENÇÃO: mostre como você chegou aos valores exibidos. Valores sem contas **NÃO** valem ponto!).

2. Considere novamente o módulo `prog.c` da questão anterior.

   (a) (1,0 ponto) Suponha que este módulo seja compilado com *gcc -Wall -m32 -c prog.c*, gerando um arquivo objeto `prog.o`. Se inspecionássemos a tabela de símbolos desse arquivo objeto com o programa `nm`, que símbolos apareceriam como **D** (símbolo na área de dados, exportado), **T** (símbolo na área de código, exportado) e **U** (símbolo indefinido)?

   (b) (0,5 ponto) Considere agora o seguinte módulo C (`aux.c`):

```
extern double valor;
double minha_funcao(int p1, float p2, double p3) {
  return valor * (p1+p2+p3);
}
```

   Suponha que quisessemos gerar um executável a partir desses dois módulos, executando o comando *gcc -Wall -m32 -o tudo prog.c aux.c*. Indique em que passo essa geração falharia (pré-processador, compilador, montador ou ligador) e qual seria o motivo dessa falha.

3. Traduza as funções `foo` e `boo` abaixo para assembly IA-32 (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.

   (Não se preocupe em entender o que as funções fazem, apenas traduza-as literalmente.)

   Comente seu código!

   (a) (2,5 pontos)

```
float outra1(float val);

float foo (double *vals, int n) {
  float junta = 0.0;
  int i;
  for (i = 0; i < n; i++) {
    junta += outra1((float) (*vals));
    vals++;
  }
  return junta;
}
```

   (b) (2,5 pontos)

```
struct X {
  char c;
  int i1;
  int i2;
};
int outra2(int x, int *p);

int boo (struct X *px, int n) {
  int continua = 1;
  int ultimo;
  while (n && continua) {
    ultimo = outra2(px->i1, &continua);
    px++;
    n--;
  }
  return ultimo;
}
```

Boa Prova!