PUC-Rio – Software Básico – INF1018
Prova 2 – 29/11/12 – Turma 3WA

1. (2,5 pontos) Suponha que, em determinada execução da função `foo`, mostrada abaixo, cada instrução dela esteja carregada na memória a partir do endereço mostrado na coluna da esquerda. Considerando que, imediatamente depois da execução da instrução `call bar`, o valor de `%esp` seja 0x1f223344, mostre a *pilha de execução*, byte a byte, nesse momento, entre os endereços `0x1f223344` e `0x1f22334f`.

```
            foo:
0x012f3300    push %ebp
0x012f3301    movl %esp, %ebp

0x012f3303    push $3
0x012f3305    push $4
0x012f3307    fildl (%esp)
0x012f330a    addl $4, %esp
0x012f330d    fildl (%esp)
0x012f3310    addl $4, %esp
0x012f3313    fdivp
0x012f3315    subl $8, %esp
0x012f3318    fstl (%esp)
0x012f331b    subl $4, %esp
0x012f331e    fstps (%esp)
0x012f3321    call bar
0x012f3326    addl $12, %esp

0x012f3329    movl %ebp, %esp
0x012f332b    popl %ebp
0x012f332c    ret
```

2. (3 pontos) Traduza a função `ctst` abaixo para assembly IA-32 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros e resultados em C. Comente seu código. *ATENÇÃO:* Não se importe com o que a função ctst faz, apenas traduza-a literalmente.

```
struct B {
  int v;
  double d;
};
double bemboba (double a);
int maisboba (double *a, int num);

double ctst (struct B *l, int n) {
  int i;
  double vals[n];
  double *pv = vals;
  for (i=0; i<n; i++) {
    *pv = bemboba(l->d);
    pv++;
    l++;
  }
  return maisboba(vals, n);
}
```

3. (2,5 pontos) Escreva uma função `flwrite` em assembly IA-32 (convenções Linux) que receba um descritor de arquivo aberto, um array de valores float e o tamanho desse array, e escreva essa lista de valores no arquivo em formato big-endian. A função `flwrite` não retorna nada:

```
void flwrite (int fildes, float nums[], int tam);
```

Sua função deve escrever no arquivo sem chamadas à biblioteca C, utilizando diretamente a chamada `write` do sistema operacional.

Segue uma descrição resumida e simplificada da página `man` da função `write`:

```
     int write(int d, const void *buf, int nbytes);
DESCRIPTION
     write() attempts to write nbytes of data to the object referenced by the
     descriptor d from the buffer pointed to by buf.
RETURN VALUES
     Upon successful completion the number of bytes which were written is
     returned.
```

Para construir a chamada ao sistema, lembre-se que o número da chamada é colocado em `%eax` e os argumentos, na ordem em que aparecem no protótipo, nos registradores `%ebx`, `%ecx`, `%edx`, `%esi` e `%edi` (quantos forem necessários). O número da chamada `write` é 4. A interrupção que aciona o sistema operacional é a `0x80`.

Sugestão: Chame `write` para escritas de 1 byte. (Pense da seguinte forma: para cada float do array, a função deve escrever os 4 bytes na ordem correta.)

4. Considere os dois arquivos abaixo:

arquivo t1.c:

```
#include <stdio.h>
int max = 2049;
void boo ();
int main (void) {
  boo();
  printf ("%d\n", max);
  return 0;
}
```

arquivo t2.c:

```
XXX char max;
void boo () {
  max = 0;
}
```

Imagine que compilemos esses arquivos com `gcc -Wall -m32 -o t t1.c t2.c` e, caso tivermos, executemos o programa resultante. Indique qual vai ser o resultado da compilação e, em caso de sucesso, da execução do programa resultante, em cada um dos casos abaixo (justifique suas respostas):

(a) `XXX` = extern

(b) `XXX` = static