**PUC-Rio – Software Básico – INF1018**
**Prova 2 – Turma 3WB – Exemplos de Soluções**

1. Os endereços de memória correspondem ao topo da pilha de execução (endereço de retorno e parâmetros da função)

```
ffffd2a0 e0  --> 4 bytes correspondentes ao endereço de retorno da função
ffffd2a1 83
ffffd2a2 04
ffffd2a3 08
ffffd2a4 ff  --> 4 bytes correspondentes ao parâmetro inteiro
ffffd2a5 fb        (-1025 -> complemento a 2)
ffffd2a6 ff
ffffd2a7 ff
ffffd2a8 00  --> 8 bytes correspondentes ao parâmetro double
ffffd2a9 00
ffffd2aa 00      2.0  ->  10.0  -> 1.0 * 2^1
ffffd2ab 00         s = 0
ffffd2ac 00         exp = 1 + 1023 = 1024 -> 10000000000
ffffd2ad 00         m = 0000...0
ffffd2ae 00
ffffd2af 40
ffffd2b0 00  --> 4 bytes correspondentes ao parâmetro float
ffffd2b1 00      -3.75 ->  11.11 -> 1.111 * 2^1
ffffd2b2 70         s = 1
ffffd2b3 c0         exp = 1 + 127 = 128 -> 10000000
                    m = 11100000...0
```

2. Observe o layout da estrutura na memória:

```
       vc (1 byte) + 3 bytes de padding
  +4   vi (4 bytes)
  +8   vd (8 bytes)
  +16  next (4 bytes)
```

   A função abaixo **não é um gabarito** pois há várias soluções corretas possíveis!

```
boo: push   %ebp
     mov    %esp, %ebp
     push   %ebx
     sub    $20, %esp     -> 8 bytes para mul + 12 bytes parâmetros de foo
     fld1
     fstpl -12(%ebp)      -> mul = 1.0
     mov    8(%ebp), %ebx
while:
     cmp    $0, %ebx
     je     fim
     fldl   8(%ebx)       -> px->vd
     fadds  12(%ebp)      -> soma f (float)
     fstpl  -20(%ebp)     -> "empilha" parâmetro b de foo
     movl   4(%ebx),%eax  -> "empilha" parâmetro a de foo
     movl   %eax, -24(%ebp)
     call   foo
     fmull  -12(%ebp)     -> mul * retorno de foo em %st(0)
     fstpl  -12(%ebp)
     mov    16(%ebx), %ebx   -> px = px->next
```

```
     jmp    while
fim:
     fldl   -12(%ebp)    -> coloca valor de retorno em %st(0)
     movl   -4(%ebp), %ebx
     movl   %ebp, %esp
     pop    %ebp
     ret
```

3. A função abaixo **não é um gabarito** pois há várias soluções corretas possíveis!

```
writeDoubles:
     push   %ebp
     mov    %esp, %ebp
     push   %ebx
     mov    16(%ebp), %edx  -> número de doubles
     imull  $8, %edx     ->  * 8 -> número de bytes
     mov    $4, %eax -> número da chamada
     mov    8(%ebp), %ebx
     mov    12(%ebp), %ecx
     int    $0x80
     pop    %ebx
     pop    %ebp
     ret
```

4. (a) Símbolos exportados e importados:

   `Exportados: buff e g`
   `Importados: i, f, strcat, strcpy, strlen, malloc`

   (b) Seria impresso o valor 0.
   Note que a variável `k`, é declarada como `static` no outro módulo e, portanto, não é visível a este módulo, que utiliza a "sua" variável `k` (em uma outra localização na memória).