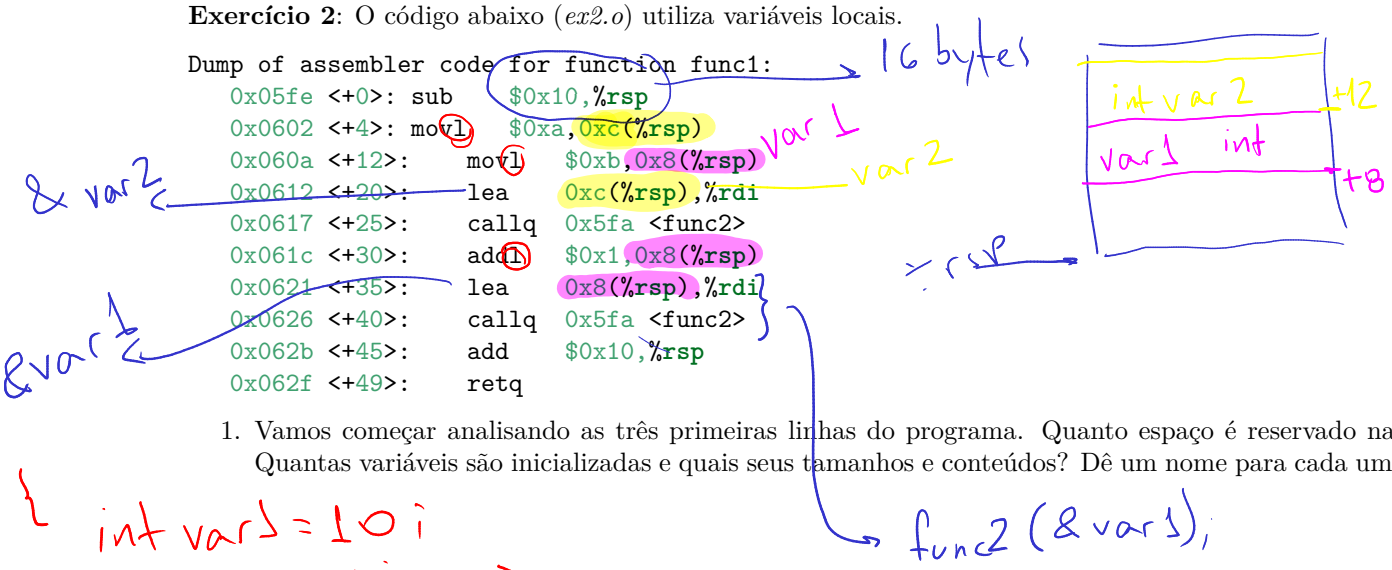
Exercício 1: É possível que o `lea` abaixo seja aritmético? Por que?

```
lea     0x8(%rsp),%rdx
```

Exercício 2: O código abaixo (*ex2.o*) utiliza variáveis locais.

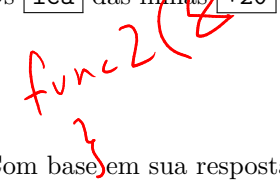
```
Dump of assembler code for function func1:
   0x05fe <+0>: sub    $0x10,%rsp
   0x0602 <+4>: movl   $0xa,0xc(%rsp)
   0x060a <+12>:    movl   $0xb,0x8(%rsp)
   0x0612 <+20>:    lea    0xc(%rsp),%rdi
   0x0617 <+25>:    callq  0x5fa <func2>
   0x061c <+30>:    addl   $0x1,0x8(%rsp)
   0x0621 <+35>:    lea    0x8(%rsp),%rdi
   0x0626 <+40>:    callq  0x5fa <func2>
   0x062b <+45>:    add    $0x10,%rsp
   0x062f <+49>:    retq
```



1. Vamos começar analisando as três primeiras linhas do programa. Quanto espaço é reservado na pilha? Quantas variáveis são inicializadas e quais seus tamanhos e conteúdos? Dê um nome para cada uma delas.

2. Identifique onde as variáveis locais encontradas são usadas.

3. Os `lea` das linhas `+20` e `+35` podem ser aritméticos? Que operação eles representam?



4. Com base em sua resposta acima, traduza as chamadas de função que ocorrem nas linhas `+25` e `+40`.

5. Traduza o programa acima para *C*