Consider the C function below. We show, next to it, the results produced by a register allocator.

```
01 int foo(void) {
02   int p0 = bar();    allocate p0 on r1
03   int p1 = baz();    allocate p1 on r2
04   if (p0 % 2) {
05     return p1;       read p1 from r2
06   } else {
07     return p0;       read p0 from r2
08   }
09 }
```

Figure 1: A simple program written in C, and the bindings produced by a register allocator.

1. The allocation is not correct. What is the problem with this assignment of variables to registers?

2. Could you write a static analysis to check if the results produced by a register allocator are correct or not?

3. Which information is manipulated by your analysis? In other words, what is the product of it? Given a program, what does it produce for that program?

4. Let *R* be the result of the static analysis (whatever *R* can be). How do you use these results, encoded in *R*, to find out if the allocation is correct or wrong?

5. If your analysis does not find an error in the allocation, does that mean that the allocation is correct?

6. Would it be possible to design an analysis with this guarantee mentioned in Question (5)? Or, put in other terms, what are the guarantees that your analysis can deliver about the correctness of the allocation?

7. In case your answer to (6) is positive, which theoretical tools would you use to demonstrate that your analysis is correct?

UFMG
DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
Compilers Laboratory

To know more: V. Krishna Nandivada, Fernando Magno Quintão Pereira, Jens Palsberg: *A Framework for End-to-End Verification and Evaluation of Register Allocators*. SAS 2007: 153-169