Consider the program below, and its control-flow graph.

```
int foo(
  int b0,
  int b1,
  int arg3,
  int arg4
) {
  int x = 0;
  if (b0) {
    x = arg3 + arg4;
  } else {
    if (b1) {
      return 0;
    }
  }
  int y = arg3 + arg4;
  return x + y;
}
```

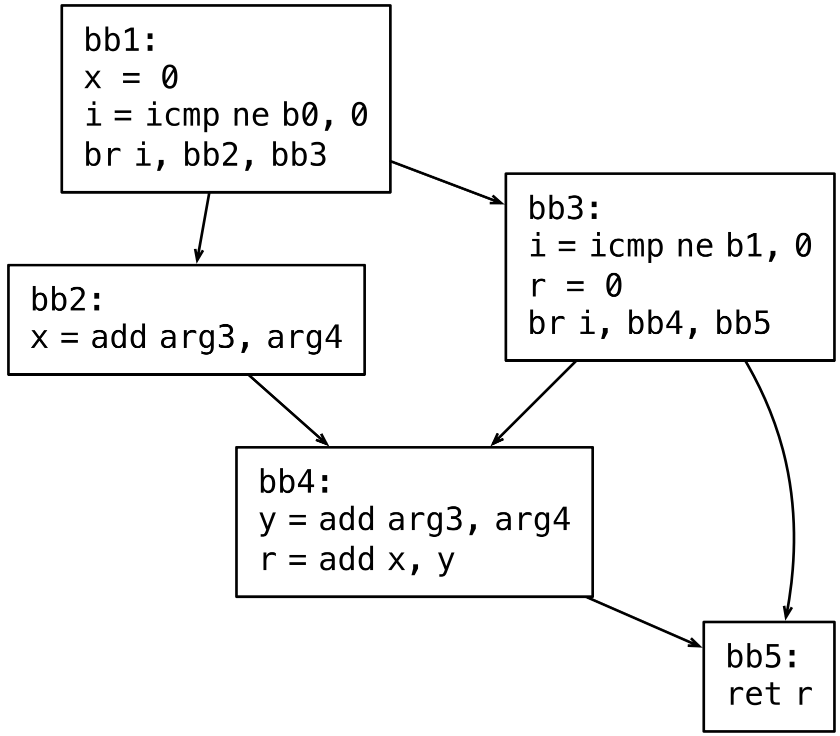

Figure 1: Example of a program that contains a partial redundancy, and its control-flow graph.

1. We say that the program above contains a “partial redundancy”. Can you spot the redundant computation in the code above?
2. Can you rewrite the program, so that it no longer contains this redundant code?
3. Is your optimization performance safe? I mean, is it possible that now, depending on the path that the program takes, it is actually running code that it should not run otherwise?
4. We say that an expression is “available” at a program point p if, regardless of the path the program takes to reach p, this expression will be computed before. What does available expressions have to do with redundant code?
5. We say that an expression is “very busy (VB)” at a program point p if, regardless of the path the program takes from p till the end of it, this expression will be calculated. What does VB expressions have to do with the performance safety of redundancy elimination?

UFMG
DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
Compilers Laboratory