Strong live variable analysis.

Code:

```
begin
        i : = 1;
        j: = 0;
        i: = i + 1;
        i: = 2;
        while (i <= 4) do {
                j: = j + i;
                i: = i + 1;
        }
        j: = 1;
end
```

Generate information from the program:

run slv “slv/slv2”

```
Block:
Label 1: i := 1
Label 2: j := 0
Label 3: i := i + 1
Label 4: i := 2
Label 5: i <= 4
Label 6: j := j + i
Label 7: i := i + 1
Label 8: j := 1

Flow:   {(1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,5),(5,8),}
Labels: {1,2,3,4,5,6,7,8,}
Init:   {1}
Finals: {8}

Analysis:
|     |Entry |Exit       |
|1    |{}    |{}         |
|2    |{}    |{}         |
|3    |{}    |{}         |
|4    |{}    |{"i",}     |
|5    |{"i",}|{"i",}     |
|6    |{"i",}|{"i",}     |
|7    |{"i",}|{"i",}     |
|8    |{"i",}|{"i","j",}|
```

Analysis:

We'll explain starting from the bottom to top what variables are live, we kill and which we generate.

8: We start with the live variables i and j. This is because, these variables are used within the program and we assume that any such variable must be live at the

beginning of the analysis. The beginning is of course the exit of 8, because this is slv. The block j := 1; kills j and because we are not in a loop or anything, all it can do it kill j. Thus at the entry of 8 has only i in it's set of live variables.

7: Here we first kill i and then generate i. Although this happens in a while loop, no other statement or condition makes any other variable live. Thus the set of live variables remains just i.

6: This is the other statement in the while loop. Now we simply both kill and generate j. However, j was never live to begin with. Because of strong live variable analysis, we do not change the set of live variables.

5: The condition simply generates i and nothing more.

4: We kill i at this point, because of the assignment. The result is now that no variables are live.

3,2,1: No variables are live, which also means that from this point onwards we will not be generating/killing any variables.