Live variable analyze.

**Code:**

begin

i : = 1;

j: = 0;

i: = i + 1;

i: = 2;

while (i <= 4) do {

j: = j + i;

i: = i + 1;

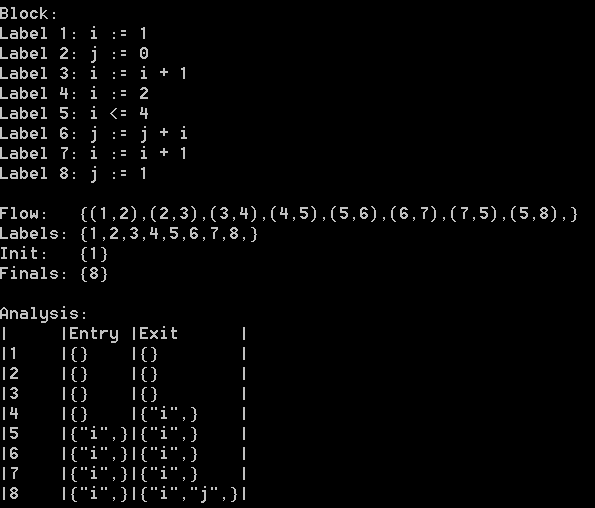
}

j: = 1;

end

**Generate information from the program:**

run “slv/slv2”



**Analyze.**

As you can see from the generated information from the program, the live variable analysis detects a lot faint variables at the beginning of the code example. This is due that at a later moment in the program the variables are re-assigned. The next part of this document will explain each step that the algorithm takes to get the same generated output as the above picture. It will be explained from bottom to top, because the algorithms works like this.

(8,5)  
At the first transformation from 8 to 5. We have as entry the variable “i”, because it needs to have a reference to the “i” at label 5. As exit it will kill the old reference to “i” and “j”.

(5,4)  
In this transformation from label 5 to 4. Needs information from the “i” and kill the old reference to “i”.

(5,7)  
In this transformation from 5 to 7. As you can see there is an “i” generated and exit. This is because we need information about the “i” in “(i + 1)”. Also the old reference to “i” is killed.

(6,5)  
This transformation is from 6 to 5. At this step it calculates ”j + i”. In this case the “j” is not important for this algorithm, because if you look at the code, you can see that it will assign “j” at the end of the program, so all the assigns before that to j are faint variables., are not important.

(7,6)  
This transformation is from 7 to 6. In this transformation we need a reference to “i” and kill the reference to the old “j”.

(5,7)  
In this transformation from 5 to 7. At this transformation we will generate a new “i”, because we need information about the “i”. The old reference to “i” needs to be killed.

(7,6)  
In this transformation from 7 to 6. All the variables will be re-assigned at a later point in the program. This means, that we need to kill the reference to this variables. So, the algorithm assumes that this is dead code or faint variables.

(6,5)  
In this transformation from 6 to 5. All the variables will be re-assigned at a later point in the program. This means, that we need to kill the reference to this variables. So, the algorithm assumes that this is dead code or faint variables.

(4,3)  
In this transformation from 4 to 3. We need information about an “i” and kill the reference to the old “i”.

(5,4)  
In this transformation from 5 to 4. In this step we need information about the “i” in the if. Also the old reference to the variable “i” is killed.

(4,3)  
In this transformation from 4 to 3. All the variables will be re-assigned at a later point in the program. This means, that we need to kill the reference to this variables. So, the algorithm assumes that this is dead code or faint variables.

(3,2).  
In this transformation from 3 to 2. All the variables will be re-assigned at a later point in the program. This means, that we need to kill the reference to this variables. So, the algorithm assumes that this is dead code or faint variables.

(2,1)  
In this transformation from 2 to 1. All the variables will be re-assigned at a later point in the program. This means, that we need to kill the reference to this variables. So, the algorithm assumes that this is dead code or faint variables.