Kompilatorteknikk TDT4205 - Problem set 6

Ola Kristoffer Hoff

21st April, 2023

1 Data flow analysis

1.1 Control flow graph

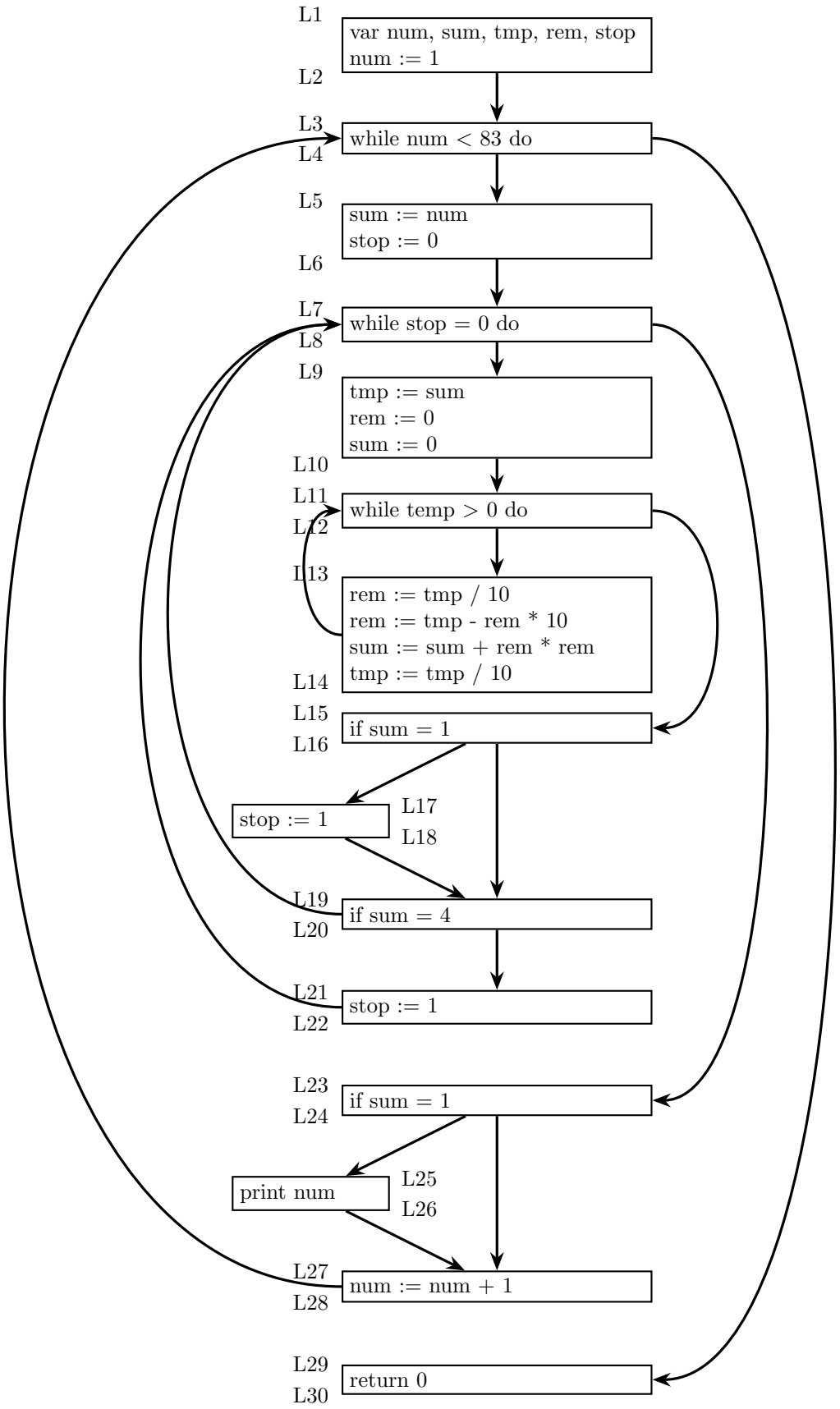

Figure 1: Control flow graph for the program.

1.2 Reaching definitions

First let's list the declarations in the CFG in figure 1:

- d1: num := 1
- d2: sum := num
- d3: stop:= 0
- d4: tmp := sum
- d5: rem := 0
- d6: sum := 0
- d7: rem := temp / 10
- d8: rem := temp - rem * 10
- d9: sum := sum + rem * rem
- d10: tmp := tmp / 10
- d11: stop := 1
- d12: stop := 1
- d13: num := num + 1

Now we list the data flow equations:

- $L1 = \{\}$
- $L2 = \{L1\} \cup \{d1\}$
- $L3 = L2 \cup L28$
- $L6 = \{L5 - \{d6, d9, d11, d12\}\} \cup \{d2, d3\}$
- $L7 = L6 \cup L20 \cup L22$
- $L10 = \{L9 - \{d2, d7, d8, d9, d10\}\} \cup \{d4, d5, d6\}$
- $L11 = L10 \cup L14$
- $L14 = \{L13 - \{d2, d4, d5, d6\}\} \cup \{d7, d8, d9, d10\}$
- $L18 = \{L17 - \{d3, d12\}\} \cup \{d11\}$
- $L19 = L18 \cup L16$
- $L22 = \{L21 - \{d3, d11\}\} \cup \{d12\}$
- $L27 = L26 \cup L24$
- $L28 = \{L27 - \{d1\}\} \cup \{d13\}$

Note that I have intentionally left out the obvious ones, from one block to another, e.g. $L5 = L4$.

Now to fill these with reaching definitions:

Iteration 1:

- $L1 = \{\}$
- $L2 = \{d1\}$
- $L3 = \{d1\}$
- $L4 = \{d1\}$
- $L5 = \{d1\}$
- $L6 = \{d1, d2, d3\}$
- $L7 = \{d1, d2, d3\}$
- $L8 = \{d1, d2, d3\}$
- $L9 = \{d1, d2, d3\}$
- $L10 = \{d1, d3, d4, d5, d6\}$
- $L11 = \{d1, d3, d4, d5, d6\}$
- $L12 = \{d1, d3, d4, d5, d6\}$
- $L13 = \{d1, d3, d4, d5, d6\}$
- $L14 = \{d1, d3, d7, d8, d9, d10\}$
- $L15 = \{d1, d3, d7, d8, d9, d10\}$
- $L16 = \{d1, d3, d7, d8, d9, d10\}$
- $L17 = \{d1, d3, d7, d8, d9, d10\}$
- $L18 = \{d1, d7, d8, d9, d10, d11\}$
- $L19 = \{d1, d3, d7, d8, d9, d10, d11\}$
- $L20 = \{d1, d3, d7, d8, d9, d10, d11\}$
- $L21 = \{d1, d3, d7, d8, d9, d10, d11\}$
- $L22 = \{d1, d7, d8, d9, d10, d12\}$
- $L23 = \{d1, d7, d8, d9, d10, d12\}$
- $L24 = \{d1, d7, d8, d9, d10, d12\}$
- $L25 = \{d1, d7, d8, d9, d10, d12\}$
- $L26 = \{d1, d7, d8, d9, d10, d12\}$
- $L27 = \{d1, d7, d8, d9, d10, d12\}$
- $L28 = \{d7, d8, d9, d10, d12, d13\}$

Iteration 2:

- $L1 = \{\}$
- $L2 = \{d1\}$
- $L3 = \{d1, d7, d8, d9, d10, d12, d13\}$
- $L4 = \{d1, d7, d8, d9, d10, d12, d13\}$
- $L5 = \{d1, d7, d8, d9, d10, d12, d13\}$
- $L6 = \{d1, d2, d3, d7, d8, d10, d13\}$
- $L7 = \{d1, d2, d3, d7, d8, d9, d10, d11, d12, d13\}$
- $L8 = \{d1, d2, d3, d7, d8, d9, d10, d11, d12, d13\}$
- $L9 = \{d1, d2, d3, d7, d8, d9, d10, d11, d12, d13\}$
- $L10 = \{d1, d3, d4, d5, d6, d11, d12, d13\}$
- $L11 = \{d1, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13\}$
- $L12 = \{d1, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13\}$
- $L13 = \{d1, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13\}$
- $L14 = \{d1, d3, d7, d8, d9, d10, d11, d12, d13\}$
- $L15 = \{d1, d3, d7, d8, d9, d10, d11, d12, d13\}$
- $L16 = \{d1, d3, d7, d8, d9, d10, d11, d12, d13\}$
- $L17 = \{d1, d3, d7, d8, d9, d10, d11, d12, d13\}$
- $L18 = \{d1, d7, d8, d9, d10, d11, d13\}$
- $L19 = \{d1, d3, d7, d8, d9, d10, d11, d12, d13\}$
- $L20 = \{d1, d3, d7, d8, d9, d10, d11, d12, d13\}$
- $L21 = \{d1, d3, d7, d8, d9, d10, d11, d12, d13\}$
- $L22 = \{d1, d7, d8, d9, d10, d12, d13\}$
- $L23 = \{d1, d7, d8, d9, d10, d12, d13\}$
- $L24 = \{d1, d7, d8, d9, d10, d12, d13\}$
- $L25 = \{d1, d7, d8, d9, d10, d12, d13\}$
- $L26 = \{d1, d7, d8, d9, d10, d12, d13\}$
- $L27 = \{d1, d7, d8, d9, d10, d12, d13\}$
- $L28 = \{d7, d8, d9, d10, d12, d13\}$

Iteration 3 brings no changes and hence we are done.

2 Code Generation II

The code section went well, not much to say. I decided to keep global counters for *while*- and *if*-statements. They function like this: every time a new statement is being generated the the global counter is stored in a local variable, then it is incremented. The local variable will represent this while/if-statement as a unique number at the end of the labels in the Assembly.

For the *break*-ing I implemented a stack to keep track of which *while*-statement I was in. Pushing the unique number to the stack when entering generate while, and popping it at the end. Then I had a *peek*-function that I used to get the value to print the jump in break.