CS5214 – THE DESIGN OF OPTIMISING COMPILERS

Assignment 2

Due: 26 February 2016

Notes for the assignment:

1. This assignment is to be done on an individual basis. While you can discuss the assignment with others as much as is necessary, plagiarism will be doubly penalized – both the copier *and* the copied will be penalized.

2. The assignment is due 26 February 2016 at 11.59pm. This is a hard deadline.

3. This assignment involves møre pen-and-paper exercise than programming. You may choose to submit either a softcopy via IVLE or a hardcopy of your answer. You can pass it to me or slip it under my door if I am not in. However, if you are submitting a hardcopy, do drop a note in the IVLE submission folder so that I know as well as can grade it in IVLE itself.

4. Be sure to write down your name and matric number clearly on your submission.

Description of Assignment 2

Part 1 - The Lengauer-Tarjan algorithm

The Lengauer-Tarjan algorithm really needs some practice in order to (hopefully) get some insight on what is going on. In this assignment, please try to follow the algorithm closely in doing the exercise. A copy of the original paper has been uploaded into the "Lecture Notes" folder of the course IVLE. (Note: this is obtained from the ACM Digital Library. Since all of you have access to it via NUS Digital Library, I am not violating any copyrights here by providing it to you. However, please do not circulate it to your friends). You may need it in order to do the assignment.

1. Included in the last page of this document is a CFG.

2. Perform the depth-first numbering of the CFG a la Lengauer and Tarjan's algorithm.

3. Compute the semi-dominators for the CFG.

4. Using the semi-dominators computed above, compute the dominator tree. For which nodes do the semidominators and the dominators differ?

5. Compute the control dependence graph.

6. Perform loop detection by identifying the back edges, and computing the strongly connected components.

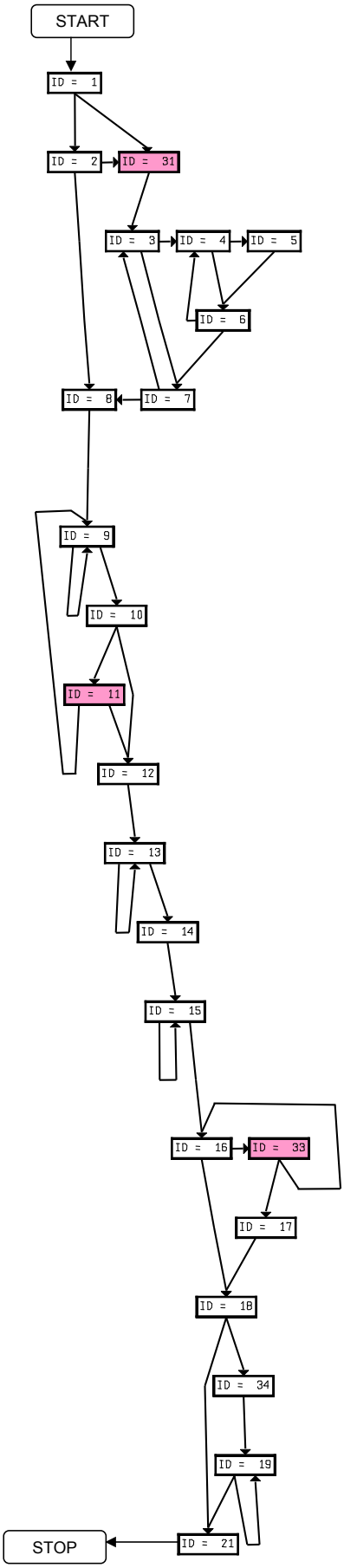
7. Suppose there is a variable that is written to in basic blocks 31, 11, and 33 (coloured pink in the CFG). Compute the dominance frontier for this variable, i.e. identify the blocks in which the ϕ–function for this variable is to be inserted.

Part 2 – Dataflow analysis

A variable is **used** if it occurs on the left hand side (let's ignore implicit input operands for the moment) of an assignment statement, or as an input of a statement. The **last use** of a variable is a node in the (statement-based) CFG such that for any path from this node to STOP, there is no further use of the variable. Last use analysis is a backward dataflow problem. Assume that a function `USE`(*n*) will return the set of variables used by the CFG node *n*.

1. Using what was discussed in the lectures, devise a dataflow analysis that will compute the last uses of all the variables in a CFG.
2. Using a simple CFG (of at least 5 nodes – including START and STOP) and at least one backedge, i.e., one loop), illustrate the working of your dataflow analysis as an example.

State clearly any assumptions that you may have made.



CFG for Part 1