1 Introduction

The objective of this assignment is to make you familiar with the Javali compiler framework used in the class. Future assignments require you to develop a compiler that uses Static Single Assignment (SSA) form as its intermediate representation. In future assignment you will be also asked to include optimizations into your compiler. You can use the Javali framework as a base for your compiler, however you can develop your own framework as well. The Javali framework is very much similar to the framework used in the Compiler Design class so if you took that class you might as well skip this assignment.

2 Building the Control Flow Graph

The first step of building the SSA-representation of a program is to record information about the program’s control flow. This assignment requires you to extend the Javali compiler so that it can build the Control Flow Graph (CFG) for programs given as input.

The construction of the CFG is placed into a separate phase of the compiler. This phase executes after the semantic analysis. You must build a CFG for each method using the class `cd.cfg.CFGBuilder`. The graph should be stored in a new data structure separate from the AST. The classes `cd.ir.ControlFlowGraph` and `cd.ir.BasicBlock` should be used to represent control-flow graphs. One `cd.ir.ControlFlowGraph` instance should be associated with each `cd.ir.Ast.MethodDecl` instance (a new field, `cfg`, has been added to `MethodDecl` for this purpose). BasicBlocks consist of a sequence of AST nodes. Note that AST nodes representing complex control-flow, such as If or While, should not be placed into a BasicBlock’s list of instructions. Their control flow should be reflected instead in the structure of the flow graph which you construct.