CSM2010 - Compiling Techniques
Course Assignment 2008-2009

Department of Computer Science. University of MALTA
Lecturer : Sandro Spina

The assignment is split in two parts. Part 1 (Parse Tree Generation) carries 50% of the overall mark of the assignment while Part 2 (Intermediate Code Generation) carries the rest of the overall marks awarded to the assignment.

Part 1 – Parse Tree Generation (50% of overall mark)

For the first part of the assignment you need to use JavaCC (or any combination of lexer+parser) in order to generate parse trees for a language of your choice. You will start off the assignment by specifying the grammar for a simple programming language of your choice , valid programs of which will then be parsed by your generated parser. Your grammar should contain production rules to parse variable declarations, expression (both arithmetic and boolean) and basic flow control statements for eg. assignment, if_then_else and while_do, as listed below.

if bexpr **then {** stmt **}**
id := expr
while (bexpr **) do { stmt }**

Note that in your grammar, multiplication and division should have higher operator precedence than addition and subtraction. Also brackets may be used to emphasize precedence. All boolean and arithmetic operators are left-associative.

You will need to develop the specification file (for JavaCC) and also integrate Java code into it. The output of this part should be a textual (or graphical if you prefer) representation of the parse tree.

Part 2 – Intermediate Code Generation (50% of the overall mark)

Given the parse tree of a valid program (from part 1), for the second part of your assignment you need to implement a traversal algorithm which outputs linear 3-address intermediate code. Lvalue and Rvalue functions can be used to generate the intermediate code. The output of this part of the assignment is a file with this intermediate code.