UCT 2004 CSC305 Compilers

Practical Assignment 2 : The Full Monty (well, close anyway)

Due: 14 May 2004

Part 1

Create an Abstract Syntax Tree from the concrete syntax information parsed by the MiniJava parser. (Hint: Instead of hand-coding your nodes and data structures, try a tool such as SableCC).
Output this AST as a tree, for verification purposes, preferably to a file. Your output must make obvious what the structure and contents are, as this will form the primary basis for determining correctness. For simplicity, you may draw your trees in ASCII, similar to the example below:

Root
----|FirstChild
----|SecondChild
--------|GrandChild1
--------|GrandChild2
----|ThirdChild

Part 2

Generate activation record frames for each subprogram in your AST.
Output the contents of each frame after analysis, preferably to a file. Your output should indicate the class and method for each frame, and the names defined in that frame.

Part 3

Translate the AST into an IR using the IR language recommended by Appel.
Output your IR tree after translation, preferably to a file.

In each case, test your code with and provide output for the sample Factorial program provided in the text (and provided with the previous tutorial). Additional test programs will be provided later – you must provide output for each test program for each layer.

You will be provided with sample output for each layer so you can proceed with the next one if your code does not work – in these cases, you may simple read in the data files provided and proceed with translation. You will also be given the basic parser definitions in JavaCC and SableCC.

You may use any code available on the textbook's website, but are not required to do so.

Make the following assumptions and simplifications:

- General-purpose registers will rarely contain anything useful – all data will be stored in activation record instances and moved into and out of registers immediately before/after instructions. The exception is return values from methods – choose an appropriate register for this purpose.
- Classes will contain only methods and no variables. There will be no inheritance.
- Only stack-based memory will be used. The new operator will therefore not allocate memory, but only make an association between a name and its class. (note: this effectively converts the OO nature of the language to a Package nature)
- All the test/sample programs will be error-free – you may do type-checking but it is not required.

Your assignment will be marked according to the following criteria:
Correctness (40%) (test programs generate correct output at each layer)
Documentation (15%) (comments within hand-written source and short report, 3 pages max)
Efficiency (15%)
Stress (20%) (hidden test programs generate correct output at each layer)
Creativity (10%) (code optimisations, etc.)

Good Luck !