UCT 2006 CSC3005 Compilers

Practical Assignment 2: IR Trees

Write a program to convert Abstract Syntax Trees into IR trees.

The abstract tree structure is for a simple expression calculator and only contains statements for assignment and printing. The grammar is illustrated by the following trees:

statementlist (print(integer(1)),print(integer(2)))
print (integer (1))
assign(variable(a),variable(b))
assign(variable(a),plus(integer(1),integer(2)))
assign(variable(a),minus(integer(5),integer(2)))
statementlist(assign(variable(a),times(integer(1),integer(2))),print(variable(a)))

Your program must read in the abstract syntax tree from a file and convert it to an IR representation, following the IR language provided in the textbook, with the following changes:

- Assume all variables are stack-based, so each variable X should be converted to MEM[BINOP['+',CONST[M_X],TEMP[FP]]].
- Use BINOP instead of the shorthand notation.
- Assume that there is a CALL_EXT node with "print" and an expression as parameters, in order to produce output.

Your main program must accept a single command-line parameter that is the name of a data file containing a single line of source text, with no spaces.

Your output must be a flattened tree structure, for example:

CALL_EXT(print,BINOP(*,BINOP(+,CONST(1),CONST(2)),CONST(3)))

You may use Java or any other programming language that the TA agrees to. You may use any built-in data structures and any parser tools (though those are not necessary).

Test your code with and provide output for the 4 data files attached (testdata.tar.gz). In each case, use output redirection to capture the output and save it to appropriately-named files (e.g., test1.out).

Your assignment will be marked according to the following criteria:

- Correctness (40%) (test programs generate correct output)
- Documentation (20%) (comments within hand-written source – no report necessary)
- Efficiency (10%)
- Stress (30%) (hidden test programs generate correct output)