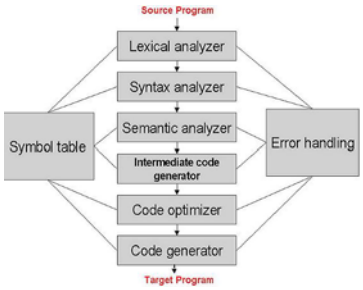
Project Part 5
Interpreter/ Code Generator
10 Points



This is the final and most exciting part of your project!

Write a code generator or interpreter for Javalet programs. Try to compile at least 1 "real" program as well as the usual examples.

Method 1 Write your interpreter as C or Java code. The interpreter should “execute” the program by walking the tree. Put in assignment statements since we don’t have input statements. You can decide whether to declare the variables or not (make everything an integer). Your interpreter can put the computed values in a table which is printed out at the end.

Method 2 Write an interpreter (actually a code generator) which outputs assembler of your choice or pseudo assembler. You must document the output code for your examples so we can try to tell if it works. There is no need to assemble, link and run (this course is about compiling), but if you want to (and have time!), add what you need (see Method 1) – 1 integer data type, please!

Input Your AST (for a program).
Output You want to show that your program works.

Code Generation/Interpretation Method The ast can be interpreted a number of ways. Section 11.3 shows code being generated by a simple tree walk. Section 11.7 shows a more elaborate algorithm (for expressions - you'll need to add other constructs).

Lex and Yacc In C Add a call in Main right after you have printed out the ast to run the code generator (that way you still have a pointer to the ast)

Javacc Users Try to turn your Java Code into an Applet

For all: Here are the options if you are short on time:

Output 1: (for up to 5 points) Program 1 works
Output 2: (for up to 7 points) Programs 1 and 2 work
Output 3: (for up to 8.5 points) Programs 1- 3 work
Output 4: (for up to 10+ points) Everything including a practical program works
.
Don't try to implement this all at once. Start small. Keep each stage's output so you have something to pass in if you don't get to the next stage.

Turn in everything in a nice package. Include the input programs and a screen dump (if you created screens); otherwise include the input and output. Submit the program separate from the documentation

Documentation This time you should submit a nice "package" which shows the whole process of compiling and what your compiler can do. (Take it to a job interview!)