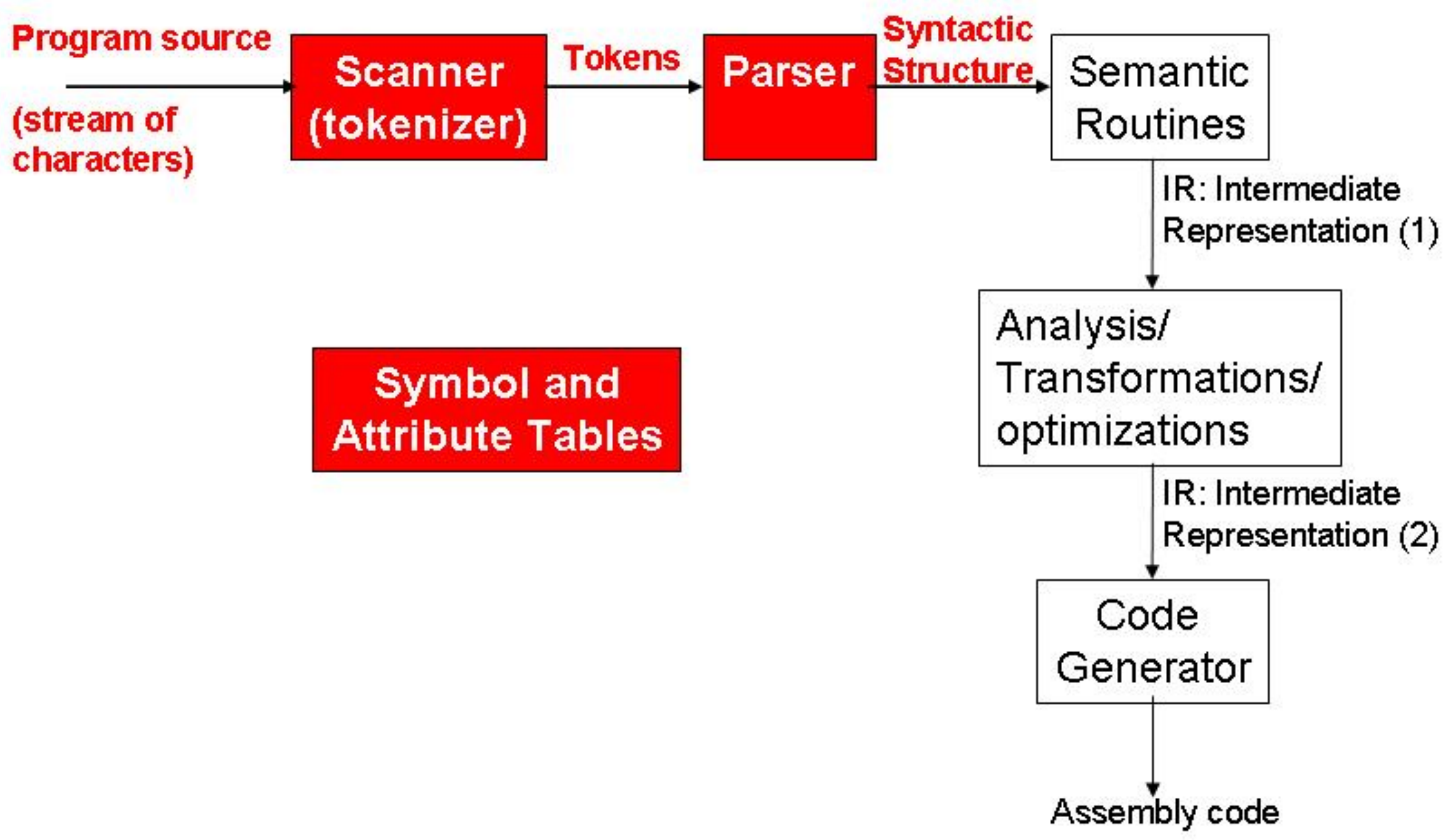
Step 3: Symbol Table Generation - Due: Friday, October 8th, 11:59pm

Introduction

Your goal in this step is to process varaiable declarations and create Symbol Tables. A symbol table is a data structure that keeps information about non-keyword symbols that appear in source programs. **Variables** and **String Variables** are examples of such symbols. Other example of symbols kept by the symbol table are the names of functions or procedures. The symbols added to the symbol table will be used in many of the further phases of the compilation. The diagram below shows the progress in our compiler construction at the end of this step:


Program source
(stream of characters)
Scanner (tokenizer)
Tokens
Parser
Syntactic Structure
Semantic Routines
IR: Intermediate Representation (1)
Analysis/ Transformations/ optimizations
IR: Intermediate Representation (2)
Code Generator
Assembly code
Symbol and Attribute Tables

The Task: The Compiler's Symbol Table

In Step 2 you didn't need token values since only token types are used by parser generator tools to guide parsing. But in this step your parser needs to get token values such as identifier names and string literals from your scanner. You also need to add semantic actions to create symbol table entries and add those to the symbol table.

In our Micro language there are two different scopes where variables can be declared. First, variables can be declared outside of any function. These variables are called "global variables" and can be accessed from any function. Next, variables can be declared inside a function. These variables are called local variables and can be accessed only from inside that function. You have to create a symbol table that stores global variables and also create one symbol table per function definition. Symbol tables will hold information about variable declarations and string declarations. Functions have names and function parameters have names so they are also considered symbols but in this step you don't have to handle function declarations and function parameters.

Symbol table entry

Once you process a declaration you can get various information about a variable. Defining a class/structure that can hold that information will simplify implementation. We call such class/structure a symbol table entry.

A symbol table entry will have:

- Name of a variable
- Type of a variable

For string variables you need an additional:

- String value (String literal)

Test cases and output

- Testcase 1 and tableoutput
- Testcase 2 and tableoutput

Submission

The same requirements (directory structure, bahavior of Makefile and compiler) as in step2 apply.

References

- **Alfred v. Aho, Ravi Sethi, Jeffrey D. Ullman, "Compilers: Principles, Techniques, and Tools", Addison-Wesley, 1986.**
  Symbol Tables - Chapter 7.6
- **Andrew W. Appel, "Modern Compiler Implementation in Java", Cambridge Univ. Press, 1998**.
  Symbol Table - Chapter 5.1
- **Charles Fischer, Richard LeBlanc, "Crafting a Compiler with C", The Benjamin/Commings Publishing Company, Inc, 1991.**
  Symbol Tables - Chapter 8