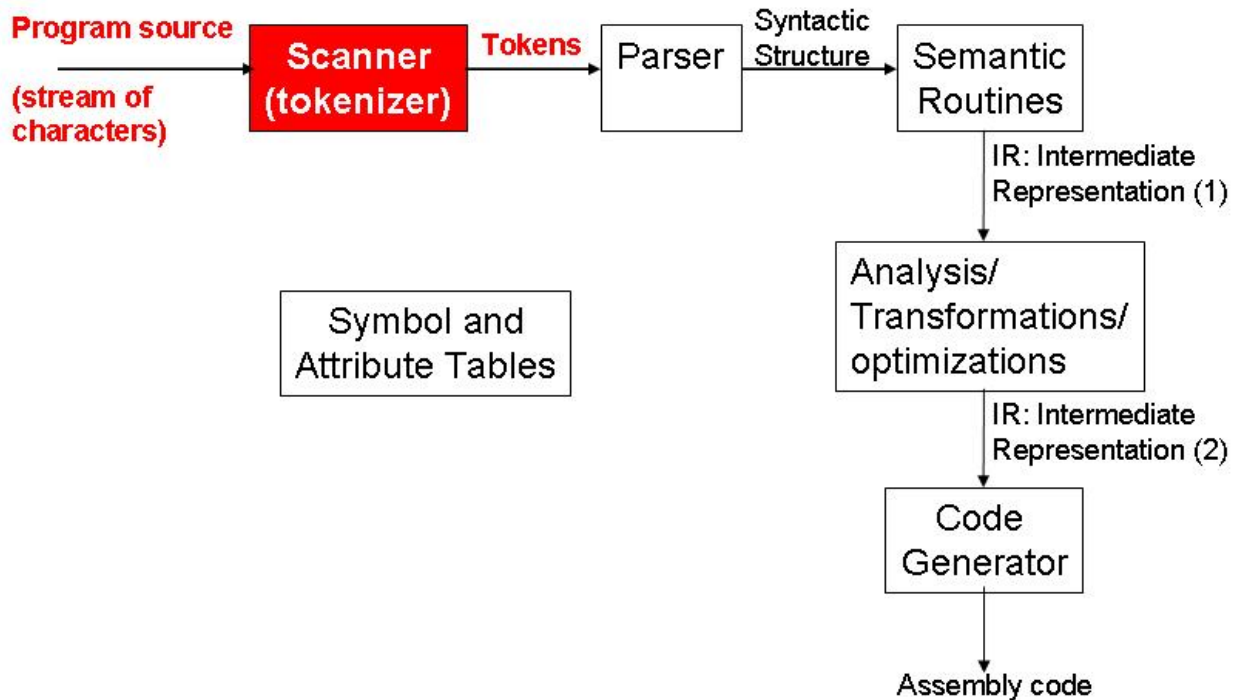# Project Step 1 - Scanner

You will develop a scanner (also known as a tokenizer) for the given language in this Step. A scanner will take a sequence of characters (i.e., the source files of the LITTLE language) as input and produce a sequence of tokens which will be the input to the next step (Parser) as shown in the figure.

Scanner Generation Tools

The scanner's source code is normally generated by a scanner generation tool such as ANTLR. These tools normally work by taking the token definitions expressed by regular expressions and generates the source code for the scanner automatically (the tokens are specified in the language's grammar).

The programmer has to add the code to handle the scanner's output. For example, at this step you will print all the tokens in the standard output. In the step 2 you will modify your scanner to feed the parser replacing the print routines by calls to the parser and passing the tokens as parameters. In order to learn how to merge the code of the scanner generator with the rest of your source code, please, read the user's manual of the tool you decide to use.

IMPORTANT: It is highly recommended that you use ANTLR. ANTLR supports Java by default. Therefore, it is recommended that you use Java for the project. If not using Java, please get approval from Instructor first.

The compiler's tokenizer (scanner)

At the end of this step, the program developed (the very first part of our compiler) should be able to open a program's source file written in the LITTLE language and recognize its tokens. At this step, the output of your program should be the prints of each token's type and value (see below). In the next step the same output will be used to feed the parser.

Your scanner program should be able to open and read a LITTLE source file and print all the valid tokens within the source file and their respective type in the standard output. You might want to redirect the output to a file and compare your results with the output files provided for the testcases.

Testcases

Test inputs and the expected outputs are given in the Step1_files.zip archive given below. See tokens.txt file for token definitions (Keywords, Operator, Literals).

Output Format

The program is expected to print each token in a predefined format. Here is an example

Token Type: INTLITERAL
Value: 20

Please read the output samples for more details.

Note: This step will be graded automatically, and the outputs generated by your code will be directly compared with the expected outputs using "**diff -B -b -s**" command. Please make sure your outputs are identical to the sample outputs provided.

What you need to submit to "Project Step 1"

- The ANTLR grammar dentition (.g4) you wrote for LITTLE. Make sure to name it **Little.g4**. Make sure to only include <u>lexer rules</u> (i.e., no parser rules).
- Your source program that is the driver for the lexing (i.e. one with the *main* method). Name it **Driver.java**. Section 3.3 of ANTLR book may be useful.
- Any other files that were not generated (meaning you wrote them)

- IMPORTANT: The grader will be using an executable called 'Micro' that takes care of running ANTLR, compiling source files, and running your code. On *osprey.unf.edu* server, the instructor will type './Micro.sh', followed by an input file name, on the terminal (as shown below), and have your code execute. Use `chmod` Unix command for setting the permission. **Submitting code that cannot be executed using the given Micro script may result in 50% penalty.**

```
$ ./Micro.sh sqrt.micro
```

What you will NOT submit

- The ANTLR jar file
- All files in 'Step1_files' archive
- All files that were auto generated (i.e., you did not write them)

Readings (Optional)

You may find it helpful to check some of the following references:

1. ANTLR documentation: https://github.com/antlr/antlr4/blob/master/doc/index.md
2. Setting Java paths:
   https://docs.oracle.com/javase/tutorial/essential/environment/paths.html

Q&A

- Q: What should I do if I encounter an illegal token?
- A: Stop the lexer at that point. You are not required to do error handling.