

Pascal to MIPS Compiler

Software Design Document

Version <1.4>

HeeChan Kang
January 30th, 2018

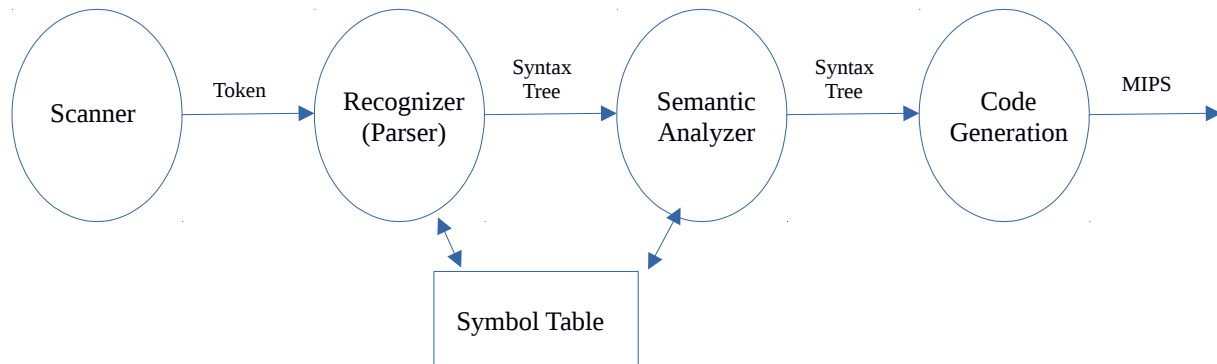
Revisions

Date	Description	Version
January 23 rd , 2018	Created the document.	1.0
January 30 th , 2018	Added cover page and separated sections.	1.1
January 30 th , 2018	Added documentation about Parser.java	1.2
February 1 st , 2018	Added graph containing the overview of the compiler.	1.3
February 3 rd , 2018	Added documentation about ParserTest.java	1.4

1. Introduction

This project is for Programming Languages and Compilers course, keystone for Computer Science majors, at Augsburg University. It contains a compiler in Java for Pascal, to finally, convert the code to MIPS. Its components are described below.

2. Overview



3. Implemented Components

/src/scanner/

- a) `Scanner.jflex` – we used a JFlex tool to create `Scanner.java` which currently contains the skeleton of being able to read-in a file and creating their respective Token.
- b) `Scanner.java` – created from the JFlex tool.
- c) `Token.java` – This class contains the simple Token object, which contains String lexeme and TokenType.
- d) `TokenType.java` – This class is an ENUM that contains all the relative token types in Pascal.
- e) `LookUpTable.java` – This class is a HashMap containing lexeme as a key and TokenType as value. The lexeme is matched to their respective TokenType using this LookUpTable.
- f) `ScannerTest.java` – This class currently contains JUnit testing for `yytext()` and `nextToken()`.

/src/parser/

- a) `Parser.java` – contains a parser for Pascal based on the grammar provided by professor. On top of abiding the rules of the grammar, there are six other methods, `isMulop()`, `mulop()`, `isAddop()`, `addop()`, `isRelop()`, and `relop()` for simplicity.

b) ParsertTest.java – contains JUnit testing cases for program(), declarations(), subprogram_declaration(), statement(), simple_expression(), and factor().