Beteab Gebru

February 20 2019

MINI-pASCAL COMPILER

SOFTWARE DESIGN DOCUMENT

Table of contents

|  |
| --- |
| 1. Overview |
| 2. Design   * Scanner * Parser   Recognizer  Symbol Table  Syntax Tree   * Semantic Analysis * / |
| 4. –-  5. References |
|  |

|  |  |
| --- | --- |
| *Change log* | |
| *Date* | ***Comment*** |
| 01/20/19 | Created document |  | BG |
| 02/28/19 | Added Overview |
| 03/01/19 | Added scanner section |
| 03/03/19 | Added section on Recognizer design |
|  |  |  | BG |
|  |  |
|  |  |

1. Overview

This program when complete will be able to compile code written in pascal language into MIPS assembly code. It will be built up in section increments.

The 5 components listed below will make up the final program. Each component will be Junit Tested at every iteration.

• Scanner

• Parser

• Symbol Table + Syntax Tree

• Semantic Analyzer

• Code Generation

Design

Scanner

pre-defined pascal language IDs and symbols are listed in the grammar document. Below are the symbols and IDs set for detection by our pascal scanner. They will form the tokens.

KEYWORDS: *AND, DIV, MOD, NOT, NUMBER, ARRAY, BEGIN, DO, ELSE, END, FUNCTION, IF, INTEGER, OF, OR, PROCEDURE, PROGRAM, REAL, THEN, VAR, WHILE*

SYMBOLS: **.**, **,**, **:=**, **\***, **/**, **+**, **-**, **>**, **<**, **>=**, **<=**, **=**, **<>**, **|**, **(**, **)**, **{**, **}**, **[**, **]**,

The package scanner has the following files which will help us comb through a given pascal code and extract meaningful tokens in order of appearance in text.

Scanner (Scanner.Jflex): the DFA scanner is made by JFlex from the *myScannerfromJFLEX.java* which details the lexical rules and provides the predefined symbols and IDs. The scanner will identify the tokens.

Token: The class token defines what a token object will look like.

The class defines *lexeme* and *type* as properties of any given token. There is also a *toString()* function that will output detected Tokens in a format -> "Token: \"" + this.lexeme + "\" of type: " + this.type;.

TokenType: this class will define list of ENUMs as specifications for the *token.type* attribute.

LookupTable: this class will extend the HashMap<> from collections. We will store all our symbols and their lexemes for lookup during Token detection. We will have ease of access for compare decisions.

## Parser

Recognizer

Symbol Table

Syntax Tree