KangCompiler User's Manual

April 21st, 2018

About This Manual

The *KangCompiler User's Manual* explains how to use the tool to compile ones pascal code into MIPS assembly language. The compiler currently supports a very primitive version of pascal, just the integers.

This user's guide discusses the usage of the command line. It assumes that you already know how to use the command line to navigate to the directory where the `.jar` and the `.pas` file resides (and to write a working pascal code, of course).

Dependencies

JRE – Java SE Runtime Environment can be downloaded at the following link:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

To ensure correct installation with the appropriate PATH system variable, try:

```
$ java –version
```

If this causes no error, you are good to go!

Executing Compiler

First, ensure that the executable `.jar` file and the code `.pas` file are in the same directory. On the command line, navigate to the path previously discussed. Given the filename of `test_pascal_code.pas` with `KangCompiler.jar` in the same folder:

```
$ java -jar KangCompiler.jar test_pascal_code.pas
```

The following command would be sufficient to run the code. Of course, the name of the target file can be renamed to better fit ones use.

Now What?

Assuming you have typed in the command above with the appropriate filename of your choice and no error messages have popped up on the command line, you should have three additional files with the identical prefix of the target file.

For example, `test.pas` would have produced:

```
test.symboltable
test.syntaxtree
test.asm
```

The **.symboltable** file contains the variable, function, procedure, and program name of the code in one place with the type of data they were associated with. The **.syntaxtree** file contains the structure of the code (pseudo-code, loosely speaking). The **.asm** file is the MIPS assembly language code of the target file. This code should be directly executable.

If an error was produced, it should be any of these followings:

1) `No file.`
   File name wasn't entered during execution.
   e.g. `$ java -jar KangCompiler.jar`

2) `Scan error.`
   There's nothing in the file! Did you save your code?

3) `Error: Match of …`
   There is a syntax error – the error message will further describe what the compiler was expecting and what was given. (Compare the error tokens with the TokenTypes section.)

4) `… There are error(s) in the code, thus code generation will not occur.`
   There are semantic errors that the compiler caught – the error message will further describe what variables and/or what types are causing the issues!

TokenTypes

`and`	AND	`,`	COMMA
`array`	ARRAY	`.`	PERIOD
`begin`	BEGIN	`:`	COLON
`div`	DIV	`[`	LSQBRACKET
`do`	DO	`]`	RSQBRACKET
`else`	ELSE	`(`	LPARENTHESES
`end`	END	`)`	RPARENTHESES
`function`	FUNCTION	`+`	PLUS
`if`	IF	`-`	MINUS
`integer`	INTEGER	`=`	EQUAL
`mod`	MOD	`<>`	NOTEQUAL
`not`	NOT	`<`	LESSTHAN
`of`	OF	`<=`	LESSTHANOREQUALTO
`or`	OR	`>`	GREATERTHAN
`procedure`	PROCEDURE	`>=`	GREATERTHANOREQUALTO
`program`	PROGRAM	`*`	ASTERISK
`real`	REAL	`/`	SLASH
`then`	THEN	`:=`	BECOMES
`var`	VAR	`write`	WRITE
`while`	WHILE	`read`	READ
`;`	SEMICOLON		

Contact Me

Please feel free to send me an email if you have any questions at `kang@augsburg.edu`