Mini Pascal to MIPS Compiler User Manual

Intro:

Welcome to the wonderful world of converting your plain old mini pascal programs to the fun and exciting MIPS assembly language! This user manual will help you get started with running your compiler and will also serve as a guide to the possible errors you will encounter.

Getting Started:

1. Boot up your favorite terminal window
2. Navigate to the directory where BergstromCompiler.jar is stored
3. Ensure you know the class path of the pascal file you would like to have compiled
4. Enter into the terminal the following command:
   (This is where you will need to know the class path for the Pascal file)
   java -jar BergstromCompiler.jar *ENTER CLASS PATH OF PASCAL FILE HERE*
5. If the compile is a success, the terminal will display "Did it compile: YES"
   You will then see that three new files have been created:
   a. SymbolTable.txt: This file contains the symbol table created by the Compiler, which is where all variables are stored for easier access by the Parser.
   b. SyntaxTree.txt: This file contains the syntax tree created by the Parser, which is a tree made up of nodes that hold information about the pascal program.
   c. MIPS.asm: This file contains the pascal program converted into the MIPS assembly language.
6. Open up the simulation software QTSPIM and load the MIPS.asm file that was created by the compiler.
7. Once loaded you can either step through the program or just hit run.
8. Watch and enjoy your converted Pascal program run as a MIPS program in the simulator.
9.

Understanding Errors:

This section will serve as an index for the various errors you may encounter.

The compiler fails:

If you execute the compiler and the terminal prints out
"Did it compiler: NO" this means that somewhere the compiling process has stopped.

Show Stopping errors:

These will be thrown when the compiling process has had a major error occurred. Each error will provide some information such as the function and line number at which the error has occurred.

These errors will look like:

- File not Found: The Parser is unable to locate the file provided.
- Scan Error: The Parser has encountered an error while loading the file provided.
- MyScanner Exception: A fatal error has occurred within the Scanner.

- Simple Expression error: A fatal error has occurred within the simple expression function.
- Factor error: A fatal error has occurred within the factor function.
- Sign error: A fatal error has occurred within the sign function.
- Relop error: A fatal error has occurred within the relop function.

Non-Show Stopping Errors:

These will be thrown when something minor has occurred during the compiler process; however these errors will not halt the compiler process.

These errors will look like:

- Program name already exists: the program name already exists in the Symbol Table.
- Variable name already exists: the variable name already exists in the Symbol Table.
- Procedure name already exists: the procedure name already exists in the Symbol Table.
- Type mismatch has occurred: occurs when a variable's type does not stay consistent through the program
    - Ex: Declaring a variable of type integer and then later attempting to assign it to type real.