Term Project | Cool Compiler (15 points)

Phase1: Cool lexical Analyzer

*Please note the project phases and lab materials are prepared from Stanford University-Compilers online course located at Coursera website.*

**Project Description**

In this project you will design and implement a compiler for a Cool language which designed specifically to ease the designing task of the compiler.

As compiler consists of many phases to be designed, this project will therefore divided and submitted through phases. The following is the time plan of phases’ submission:

|  |  |  |
| --- | --- | --- |
|  | **Submission** | **Discussion** |
| **Phase1: Cool lexical Analyzer(4points)** | 21March-2015 | 21March week labs |
| **Phase2: Cool Parser(4points)** | 11April-2015 | 11April week labs |
| **Phase3: Cool Semantic Analyzer(4points)** | 2May-2015 | 2May week labs |
| **Phase4: Cool Code Generator(4points)** | 16May-2015 | 16May week labs |

**Phase1 | Cool Lexical Analyzer (4points)**

*In this phase you are to write a lexical analyzer, also called a scanner, using a lexical analyzer generator. (The C++ tool is called ex; the Java tool is called jlex.) You will describe the set of tokens for Cool in an appropriate input format, and the analyzer generator will generate the actual code (C++ or Java) for recognizing tokens in Cool programs.*

*In this assignment, you are expected to write Flex rules that match on the appropriate regular expressions defining valid tokens in Cool as described in Section 10 and Figure 1 of the* [*Cool manual*](https://spark-university.s3.amazonaws.com/stanford-compilers/resources%2Fcool_manual.pdf) *and perform the appropriate actions, such as returning a token of the correct type, recording the value of a lexeme where appropriate, or reporting an error when an error is encountered. Before you start on this assignment, make sure to read Section 10 and Figure 1 of the* [*Cool manual*](https://spark-university.s3.amazonaws.com/stanford-compilers/resources%2Fcool_manual.pdf)*; then study the different tokens defined in cool-parse.h.*

*Your implementation needs to*

1. *define Flex/Jlex rules that match the regular expressions defining each token defined in cool-parse.h and perform the appropriate action for each matched token.*

* *For example, if you match on a token BOOL CONST, your lexer has to record whether its value is true or false; similarly if you match on a TYPEID token, you need to record the name of the type. Note that not every token requires storing additional information; for example, only returning the token type is sufficient for some tokens like keywords.*

1. *Your scanner should be robust, it should work for any conceivable input.*

* *For example, you must handle errors such as an EOF occurring in the middle of a string or comment, as well as string constants that are too long. These are just some of the errors that can occur; see the manual for the rest.*

**How to start?**

1. **First, check the following:**
   1. **The Lexical Structure of Cool Language (***Section 10 and Figure 1 of the* [*Cool manual*](https://spark-university.s3.amazonaws.com/stanford-compilers/resources%2Fcool_manual.pdf))
   2. **Tokens Defined in** *cool-parse.h (in the virtual image: /home/compilers/cool/include/PA2 )*
2. **Second, in the virtual image do the following:**
   1. **Open terminal, create new directory/floder**

 mkdir Assi1

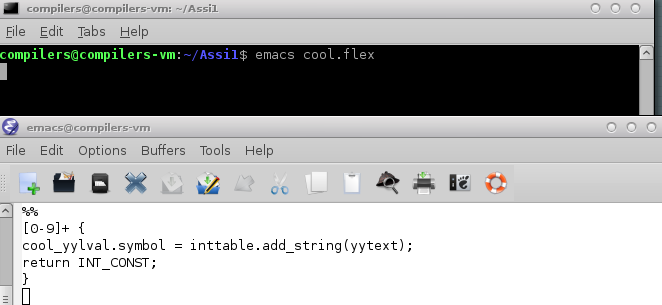
* 1. **jump to this directory:**

 cd Assi1

* 1. **type the following command:**

**** make -f /usr/class/cs143/assignments/PA2/Makefile

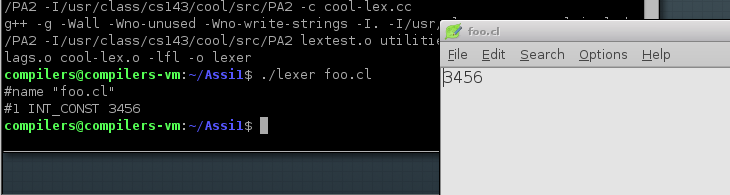
* 1. **Some files now are added to the directory you have just created**
  2. **You are required to fill the** cool.flex **with set of regular expressions that defined the lexical structure of Cool language and return the specified tokens that defined in** cool-parse.h (At minimum 5 Regular Expressions)
  3. **For example: to extract the digits as INT\_CONST token(defined in cool-parse.h) and store the value of token in a global object called cool\_yylval will write something similar to the following in cool.flex:**



Learn about inttable.addstring() by understanding the stirng table utility provided by the coursera course (check below references)

* 1. **Kindly read README file which generated during step (c) for a complete understand of what’s happening.**

1. **Third, to test your lexical analyzer:**
   1. **type the command** make lexer **to** **translate and run your cool.flex code**
   2. **[EITHER] to use** test.cl **as input program text to your lexical analyzer, type** make dotes **to run/make lexer on** test.cl **directly.**
   3. **[OR] to run lexer on another input file: lexer foo.cl (foo.cl is another cool input file)**
   4. **the output should be something similar to:**



**What to submit?**

* The lexical analyzer file (cool.flex)
* Samples of Cool programs and the output of lexical analyzer on these files (snapshot of outputs for each cool program)
* Submit a .zip file with name (COOLC-Phase1-<YourID/IDs>.zip) for example (COOLC-Phase-20110xxx.zip)

**How to Submit?**

* Team up to three members.
* Via Compilers Acadox class: [http://www.acadox.com/class/16939#tasks](http://www.acadox.com/class/16939" \l "tasks)

***Reference***

*For complete reference/information about the assignment and also a quick review of flex please check:*

* *Programming Assignment1-Stanford University - Coursera* [*https://class.coursera.org/compilers-2012-002/assignment/view?assignment\_id=2*](https://class.coursera.org/compilers-2012-002/assignment/view?assignment_id=2)
* Section 4.2 String Tables in [*https://class.coursera.org/compilers-2012-002/assignment/view?assignment\_id=2*](https://class.coursera.org/compilers-2012-002/assignment/view?assignment_id=2)