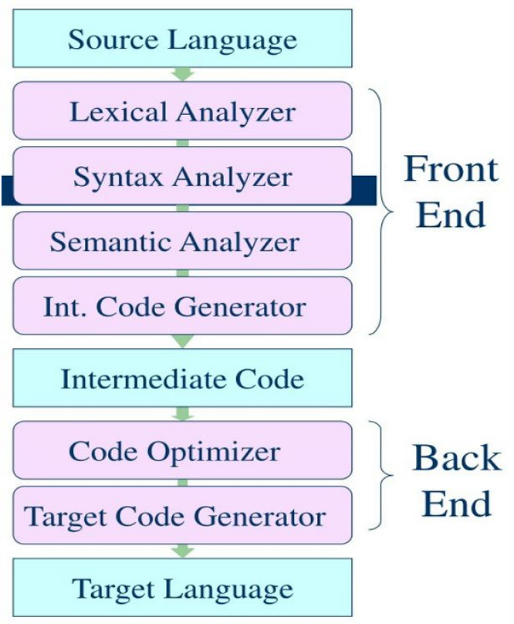
Name:Ali Khaled Fawzy

ID:200042772

1. **Introduction**

Lexical analysis is the first part of the compiler designing. Lexical analyzer is a program that breaks down the source code into a sequence of lexemes. A lexeme is a single consequence of characters like ‘{,.’ string, number, etc. Lexical analyzer reads the input and recognizes the lexemes and output a sequence of token describing the lexemes, You can say that is a pattern matcher for character strings . The main task of the lexical analyses is to read input characters and produce a token. A token is an object which describes a lexeme.

* 1. **Phases of Compiler**

****

1. **Lexical Analyzer**

Lexical analyzer takes lexemes as input and generates the tokens, The job of lexical analyer is to find out the meaning of every lexeme, It recognize the tokens with the help of regex.

The Functions of a lexical analyzer:

* Tokenization

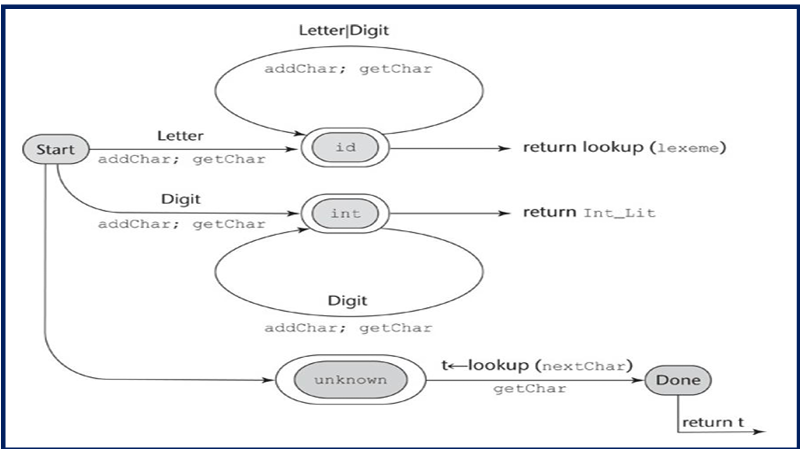
Tokenization is a process of breaking input text into words, phrases, symbols. These decomposed parts are called tokens.

* Remove white space characters

The lexical analyzer removes the white spaces by ignoring them. When the compiler scans the whole code, it groups the entire code into tokens. While this process is going on it skips the white space symbols.

* Removing comments
* Generating error messages

Lexical error Is a sequence of characters that does not match the pattern of any token. Lexical errors can be Spelling errors, Exceeding the length of an identifier, Existence of illegal characters, etc..

****

1. **Software Tools**

There are many tools but the most popular are:

1. Flex
2. Lex
3. ANTLR
4. Pygments
   1. **Computer Program**
   2. **Programming Language**

C, C++, Java, Python

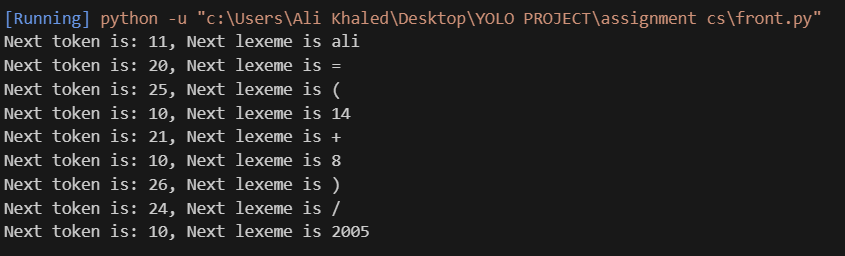
1. **Implementation of a Lexical Analyzer**

The code and its documentation is on github. (https://github.com/AliKhaled98/AliKhaled98-200042772-ass.git)

The Input statement:



The output:



The output shows the first lexeme which is ali and its token code is 11 as it is an identifier

The second lexeme is = which token code is 20 as it I an assign\_op

The third lexeme is ( which token code is 25 as It is a LPAREN

The fourth lexeme is 14 which token code is 10 as it an int\_literal

The fifth lexeme is + which token code is 21 as it is a plus\_op

The sixth lexeme is 8 which token code is 10 as it is an int\_literal

The seventh lexeme is ) which token code is 26 as it is a RPAREN

The eight lexeme is / which token code is 24 as it is a div\_op

The last lexeme is 2005 which token code is 10 as it is an int\_literal

|  |  |
| --- | --- |
| Lexeme | Token |
| Ali | identifier |
| = | assign\_op |
| ( | LPAREN |
| 14 | Int\_literal |
| + | plus\_op |
| 8 | int\_literal |
| ) | RPAREN |
| / | div\_op |
| 2005 | int\_literal |

1. **References**
2. (Medium.com) https://medium.com/@ltwrajapaksha/lexical-analyzer-c5e6690c830f
3. (Minia University)

<https://courses.minia.edu.eg/Attach/Comp404Lec2.pdf>

1. (Stack Overflow)