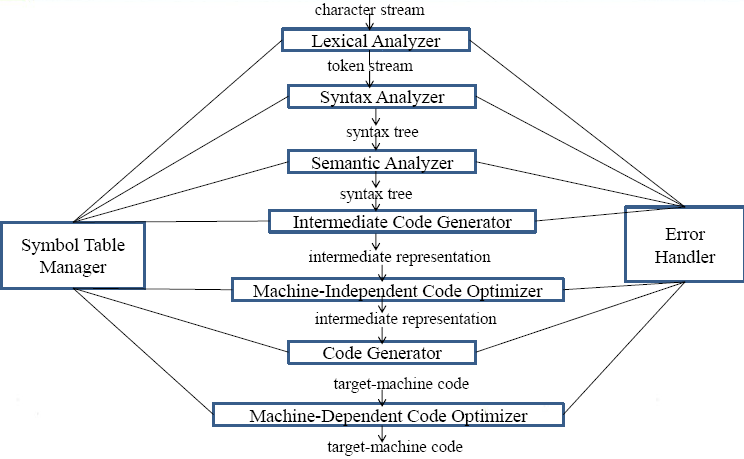
**Introduction:** Introduction to compiler, compiler and translator, the structure of a compiler.

**Automata Theory:** is a branch of computer science that deals with designing abstract selfpropelled computing devices that follow a predetermined sequence of operations automatically. An automaton with a finite number of states is called a **Finite Automaton**.

An automaton with a finite number of states is called a **Finite Automaton** (FA) or **Finite State Machine** (FSM).

**Question: 1(a) 2015 what is the different between compiler and interpreter.**

**Question: 1(a) 2017 What are the phases of the compiler? Explain with neet diagram**



### Lexical Analysis

The first phase of scanner works as a text scanner. This phase scans the source code as a stream of characters and converts it into meaningful lexemes.

### Syntax Analysis

The next phase is called the syntax analysis or **parsing**. It takes the token produced by lexical analysis as input and generates a parse tree (or syntax tree).

### Semantic Analysis

Semantic analysis checks whether the parse tree constructed follows the rules of language. For example, assignment of values is between compatible data types, and adding string to an integer.

### Intermediate Code Generation

After semantic analysis the compiler generates an intermediate code of the source code for the target machine. It represents a program for some abstract machine. It is in between the high-level language and the machine language.

### Code Optimization

The next phase does code optimization of the intermediate code.

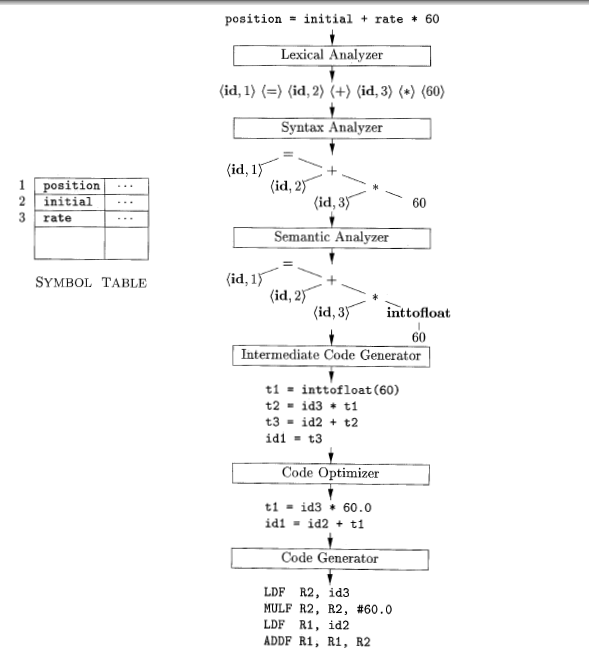
### Code Generation

The code generator translates the intermediate code into a sequence of (generally) re-locatable machine code. Sequence of instructions of machine code performs the task as the intermediate code would do.

### Symbol Table

It is a data-structure maintained throughout all the phases of a compiler. All the identifier's names along with their types are stored here.

For Understanding not for Exam written:



**Phase-1 Lexical Analysis**

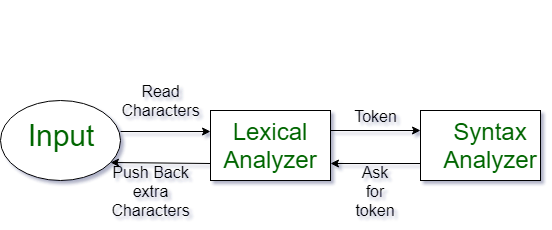
Lexical analysis is the first phase of a compiler. It takes the modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.

If the lexical analyzer finds a token invalid, it generates an error..

**Question: 1(b) 2017 Discuss the function of lexical analyzer**

Lexical Analysis is the first phase of compiler also known as scanner. It converts the High level input program into a sequence of **Tokens**.

* Lexical Analysis can be implemented with the [Deterministic finite Automata](http://quiz.geeksforgeeks.org/toc-finite-automata-introduction/).
* The output is a sequence of tokens that is sent to the parser for syntax analysis



Lexical analyzer reads the source program one character at a time, and grouped into a sequence of atomic units called tokens. Identifiers, keywords, constants, operators and punctuation symbols such as commas, parenthesis, are typical tokens.

**Question: 2(a) 2015 Define Token, lexeme and pattern. Give example**

**Token**

A lexical token is a sequence of characters that can be treated as a unit in the grammar of the programming languages.

**Example of tokens:**

* Type token (id, num, real, . . . )
* Punctuation tokens (IF, void, return, . . . )
* Alphabetic tokens (keywords)

**Example of non-tokens:**

* Comments, preprocessor directive, macros, blanks, tabs, newline,  . .

### Patterns

There is a set of strings in the input for which the same token is produced as output. This set of strings is described by a rule called a pattern associated with the token.

Regular expressions are an important notation for specifying patterns.

For example, the pattern for the Pascal identifier token, id, is: id → letter (letter | digit)\*.

### Lexeme

A lexeme is a sequence of characters in the source program that is matched by the pattern for  a token.

For example, the pattern for the RELOP token contains six lexemes ( =, < >, <, < =, >, >=) so the lexical analyzer should return a RELOP token to parser whenever it sees any one of the six.

**What is a token?**

A lexical token is a sequence of characters that can be treated as a unit in the grammar of the programming languages.

**Question: 1(c) 2015 what do you mean by alphabet and string? Discruss with various string operation.**

**Answer:**

## Specifications of Tokens

Let us understand how the language theory undertakes the following terms:

### Alphabets

Any finite set of symbols {0,1} is a set of binary alphabets, {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F} is a set of Hexadecimal alphabets, {a-z, A-Z} is a set of English language alphabets.

### Strings

Any finite sequence of alphabets is called a string. Length of the string is the total number of occurrence of alphabets, e.g., the length of the string tutorialspoint is 14 and is denoted by |tutorialspoint| = 14. A string having no alphabets, i.e. a string of zero length is known as an empty string and is denoted by ε (epsilon).

### Special Symbols

A typical high-level language contains the following symbols:-

|  |  |
| --- | --- |
| Arithmetic Symbols | Addition(+), Subtraction(-), Modulo(%), Multiplication(\*), Division(/) |
| Punctuation | Comma(,), Semicolon(;), Dot(.), Arrow(->) |
| Assignment | = |
| Special Assignment | +=, /=, \*=, -= |
| Comparison | ==, !=, <, <=, >, >= |
| Preprocessor | # |
| Location Specifier | & |
| Logical | &, &&, |, ||, ! |
| Shift Operator | >>, >>>, <<, <<< |

### Language

A language is considered as a finite set of strings over some finite set of alphabets. Computer languages are considered as finite sets, and mathematically set operations can be performed on them. Finite languages can be described by means of regular expressions.

**For example:**

int intvalue;

While scanning both lexemes till ‘int’, the lexical analyzer cannot determine whether it is a keyword *int* or the initials of identifier int value.

The lexical analyzer also follows **rule priority** where a reserved word, e.g., a keyword, of a language is given priority over user input. That is, if the lexical analyzer finds a lexeme that matches with any existing reserved word, it should generate an error.

**Define Tokens.**

The token can be defined as a meaningful group of characters over the character

set of the programming language like identifiers, keywords, constants and others.

**4. Define Symbol Table.**

A Symbol Table is a data structure containing a record for each identifier, with

fields for the attributes of the identifier.

5**. Define lexeme?**

The character sequence forming a token is called lexeme for the token.

**6. Name minimum 4 compiler construction tools.**

• LEX

• YACC-Parser generator

• Syntax directed translation scheme.

• Automatic code generator

• Data flow engines

**7. Write a short note on LEX.**

A LEX source program is a specification of lexical analyzer consisting of set of regular expressions together with an action for each regular expression. The action is a piece of code, which is to be executed whenever a token specified by the

**8. What is the role of lexical analysis phase?**

Lexical analyzer reads the source program one character at a time, and grouped into a sequence of atomic units called tokens. Identifiers, keywords, constants, operators and punctuation symbols such as commas, parenthesis, are typical tokens.

**Question:1(c)2017,2015 Differentiate between single-pass and multi-pass compiler**

|  |  |
| --- | --- |
| **single-pass** | **multi-pass compiler** |
| A one-pass compiler is a compiler that passes through the source code of each compilation unit only once.. | A multi-pass compiler is a type of compiler that processes the source code or abstract syntax tree of a program several times |
|  |  |
| A one-pass compilers is faster than multi-pass compilers | A multi-pass compilers is slower than one-pass compilers |
| Many programming languages cannot be represented with a single pass compilers, for example Pascal can be implemented with a single pass compiler | where as languages like **Java** require a multi-pass compiler. |
| A one-pass compiler has limited scope of passes. | but multi-pass compiler has wide scope of passes |

**2. Write the syntax for three-address code statement, and mention its properties.**

**Syntax: A= B op C**

• Three-address instruction has at most one operator in addition to

the

assignment symbol. The compiler has to decide the order in which

operations are to be done.

• The compiler must generate a temporary name to hold the value

computed by each instruction.

• Some three-address instructions can have less than three operands.

**13. What are the various phases of a compiler?**

The different phases of a compiler are lexical analyzer, syntax analyzer, semantic

analyzer, intermediate code generator, code optimizer, code generator, symbol table

manager, error handler.

**14. Define prefix of S.**

A string obtained by removing zero or more trailing symbols of string s;

Ex.,ban

is a prefix of banana.

**15. Define suffix of S.**

A string obtained by removing zero or more leading symbols of string s;

Ex.,nana is a suffix of banana.

**Question: 1(b) 2016 Suppose a source program contains the assignment statement,**

**Position=initial +rate\*60**

**Explain how this statement is processed and finally translated at different phases of a traditional compiler.**