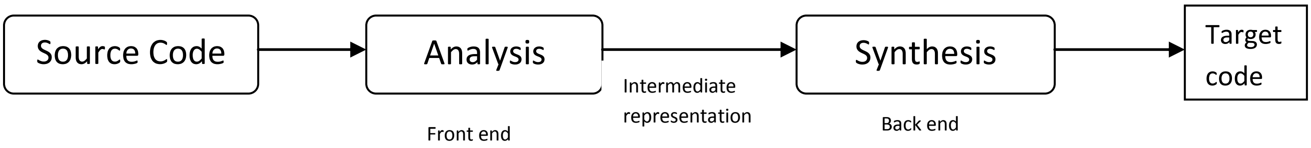
Scribe Report

Date: 30/07/2013

ARPIT KUMAR (11CS10007)

1. Structure of the Compiler

The compiler majorly consists of two parts the analysis part and the synthesis part.



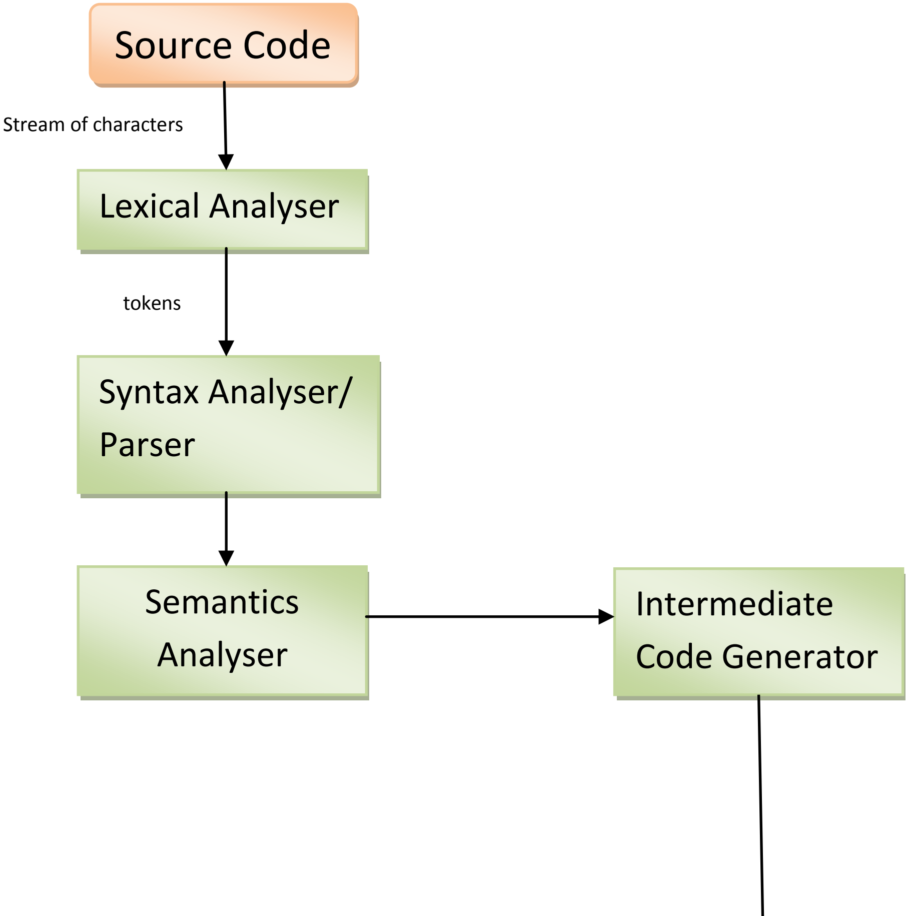
1.1. Analysis part

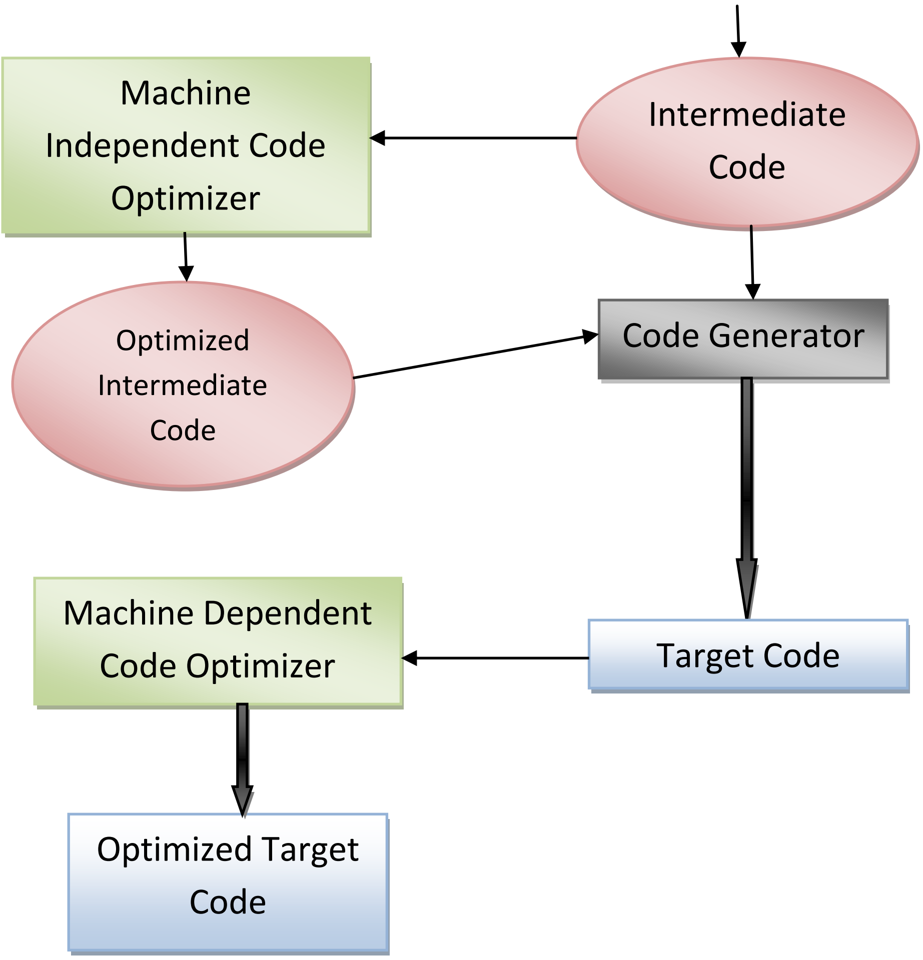
1. Splits the source program into constituent pieces.
2. Fits all those pieces based on grammatical rules.
3. Generates intermediate representation based on the grammatical rules.
   I. Generates errors.
   II. Creates data structures called symbol tables which contain information about the source program.

1.2. Synthesis part

1. It constructs the desired target program from the intermediate code and the information in the symbol table.
2. It may also pass the code to the optimizer which tries to reduce the size, execution time of the generated machine code.

Detailed Diagram




Machine Independent Code Optimizer
Intermediate Code
Optimized Intermediate Code
Code Generator
Machine Dependent Code Optimizer
Target Code
Optimized Target Code

1.1 Lexical Analysis

The first phase of a compiler is called a lexical analysis. The lexical analyser reads the stream of characters making up the source program and groups the characters into meaningful sequences called lexemes. For each lexeme, the lexical analyser produces as output a token of the form <*token name, attribute value*>. In the token, the first component, the token name is an abstract symbol used during syntax analysis, and the second component, the attribute value points to the entry in the symbol table for this token.

For example consider,

position = initial + rate * 60

position would be mapped to <**id**,1>, **id** is abstract symbol for identifier and 1 points to the entry of position in the symbol table.

= would be mapped into token <=>

initial would be mapped to <**id**,2> and 2 points to symbol table entry for initial.

+ would be mapped to <+>

rate would be mapped to <**id**,3> and 3 points to symbol table entry for rate.

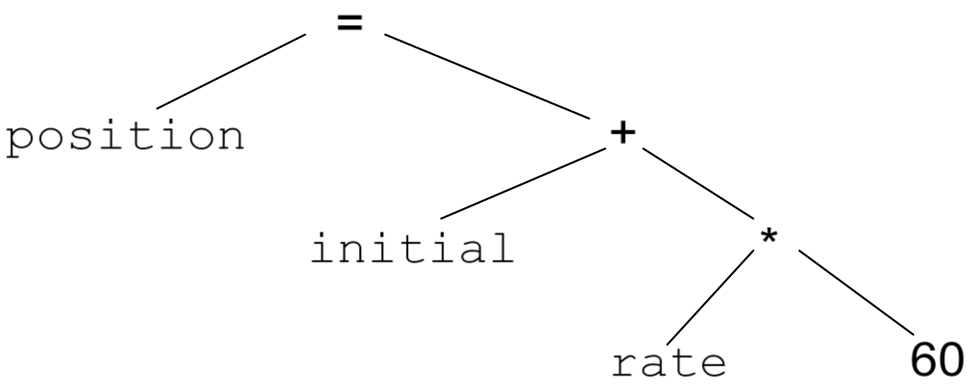
* would be mapped to <*>

60 is a lexeme which would be mapped to <60>

Now after all this the representation of the assignment statement becomes

<**id**,1> <=> < **id**,2> <+> < **id**,3> <*> <60>

The corresponding syntax tree will be:


=
position
+
initial
*
rate
60