Overview of Compiler

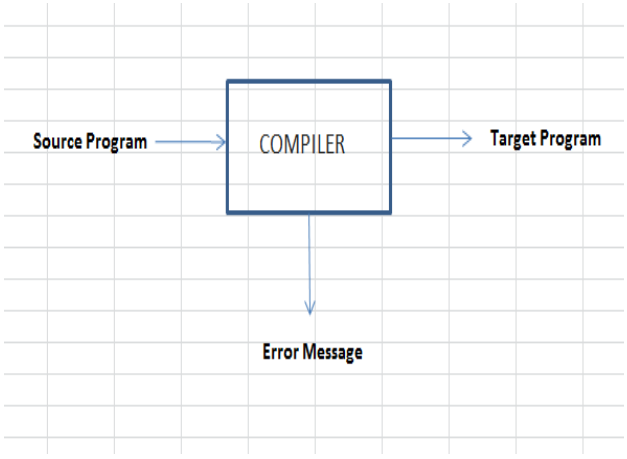
Taruna Rohdia ,Tushar Mehmi , Upender Yadav
Dronacharya College Of Engineering

***Abstract*- Compiler is a bridge between high level and low level language. Compilers are essential programming tool which improves software productivity by hiding low level details. It is a tool for designing and evaluating computer architecture. Little language and program translations can be used to solve other problems. Compiler construction is a widely used software engineering exercise, but because most students will not be compiler writers, care must be taken to make it relevant in a core curriculum**

I. INTRODUCTION

Compiler translates an executable program in one language into an executable program in another language. The term compilation denotes the conversion of an algorithm expressed in a human-oriented source language to an equivalent algorithm expressed in a hardware-oriented target language. Programming languages are tools used to construct formal descriptions of finite computations (algorithms). Each computation consists of operations that transform a given initial state into some final state. Interpreter reads an executable program and produces the results of running that program.

II. COMPILER



Compiler: A Bridge Between PL and Hardware

Applications (High Level Language)
Compiler
Operating System (Low Level Language)
Hardware

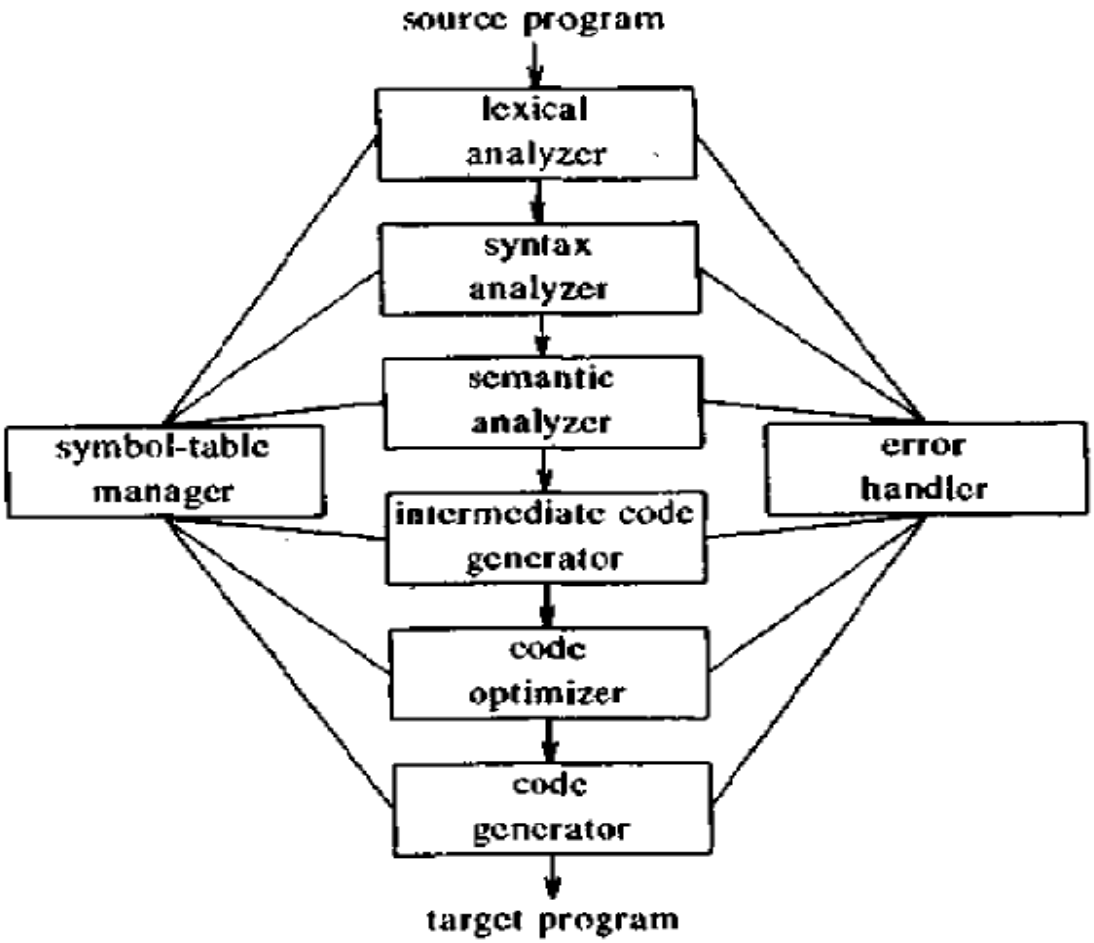
In addition to a compiler; several other programs may be required to create an executable target program. A source program may be divided into modules stored in separate files. The task of collecting the source program is sometimes entrusted to a distinct program called a preprocessor. The target program created by compiler may require further processing before it can be run. The compiler creates assembly code that is translated by an assembler into machine code and then linked together with some library routines into the code that actually runs on the machine.

Tasks of compiler:- A compilation is usually implemented as a sequence of transformations (SL;L1);(L1;L2);:::; (Lk; TL), where SL is the source language and TL is the target language. Each language Li is called an intermediate language. Intermediate languages are conceptual tools used in decomposing the task of compiling from the source language to the target language. The design of a particular compiler determines which (if any) intermediate language programs actually appear as concrete text or data structures during compilation. Any compilation can be broken down into two major tasks:

1) Analysis: Discover the structure and primitives of the source program, determining its meaning.
2) Synthesis: Create a target program equivalent to the source program.

This breakdown is useful because it separates our concerns about the source and target languages. The analysis concerns itself solely with the properties of the source language. It converts the program text submitted by the programmer into an abstract representation embodying the essential properties of the algorithm. This abstract representation may be implemented in many ways, but it is usually conceptualized as a tree. The structure of the tree represents the control and data flow aspects of the program, and additional information is attached to the nodes to describe other aspects vital to the compilation.

III. PHASES OF COMPILER



IV. ANALYSIS OF SOURCE PROGRAM

Linear Analysis:-

In which the stream of characters making up the source program is read from left to right and grouped into tokens that are sequences of characters having a collective meaning. In compiler, linear analysis is also called LEXICAL ANALYSIS or SCANNING.

Hierarchical Analysis:-

In which characters or tokens are grouped hierarchically into nested collections with collective meaning. In compiler, hierarchical analysis is called parsing or syntax analysis.

Semantic analysis:-

In which certain checks are performed to ensure that the components of a program fit together meaningfully.

Syntax Analysis:-

It involves grouping the tokens of the source program into grammatical phrases that are tied by the compiler to synthesize output. Usually the grammatical phrases of the source program are represented by a parse tree.

Compiler-Construction Tools:-

- Software development tools are available to implement one or more compiler phases
    - *Scanner generators*
    - *Parser generators*
    - *Syntax-directed translation engines*
    - *Automatic code generators*
    - *Data-flow engines*

V. CONCLUSION

It gives the overview of compiler. Compiler is a bridge between high level and low level language. This report shows the phases of compiler, basic view of compiler and tasks of a compiler.

REFERENCE

1) http://www.ijsrp.org/research-paper-0413/ijsrp-p16108.pdf
2) http://research.cs.wisc.edu/vertical/papers/2012/pldi12-idem.pdf
3) http://users.elis.ugent.be/~leeckhou/papers/hipeac08-eyerman.pdf
4) http://suif.stanford.edu/~courses/cs243/lectures/L1-handout.pdf
5) http://symbolaris.com/course/Compilers/waitegoos.pdf
6) http://people.cs.pitt.edu/~mock/cs2210/lectures/lecture1.pdf