UNIT—3

INTRODUCTION TO COMPILER

Compiler :-

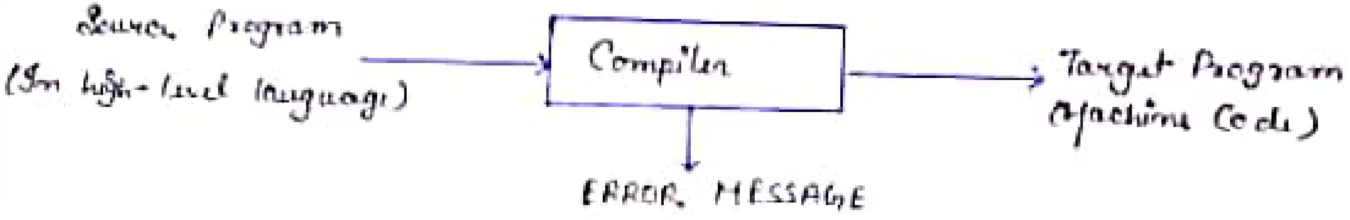
Compiler is a system tool which translate Source language into target language.

The Source Language may be in high-level language.

Generally, target language is Machine code but in some instance it may be in any other high-level language such as:- FORTRAN, Pascal, C etc.

Main aset of Compiler is to generate error report by performing error diagnostic of the S.program.

It report all the mistakes of programmer in the source program.



Approaches of Compiler Development :-

There are several approaches to compiler developments.

Here, some of them are:-

① Assembly Language Coding :-

1) Earlier compilers were mostly coded in assembly languages.

2) It is used to develop compiler which is easy to use & simple.

3) This approach worked very well for small high-level language.

4) But it is does not good with large set of instruction as High-level language.

5) As language & their computers become larger, lots of bugs started surfacing which were difficult to remove.

5) This coding is used in 1960 & 1970.

6) In current time high-level language coding for compiler is used.

7) Many computer are now coding in FORTRAN.

(Ex- Compiler IBM/360.

② Cross-Compiler :-

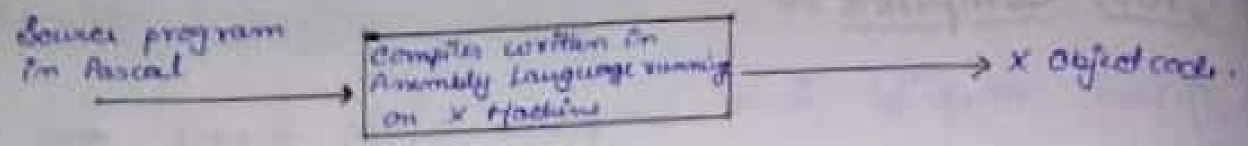
1) A cross compiler is a computer which runs on one Machine & generate a code for another Machine.

2) A cross compiler is a computer in which out of one Machine of object code (Target language) is Source language for the second machine.

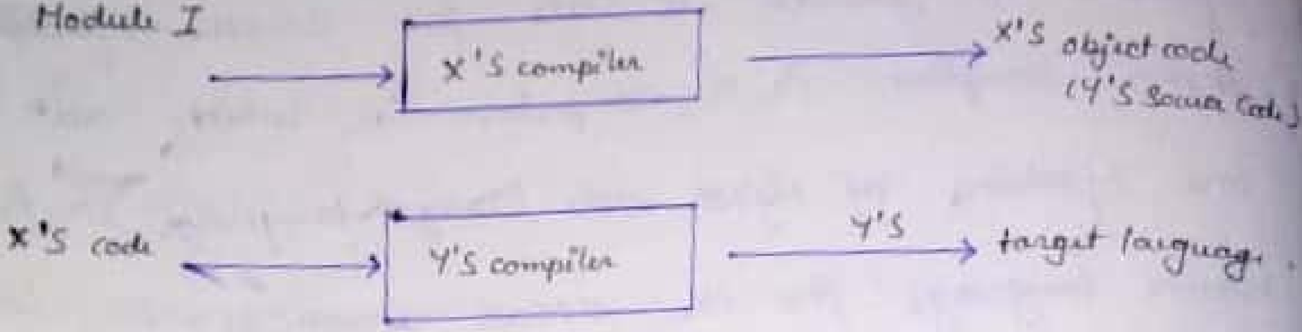
3) It means object code generated by one machine will executed in second machine.

4) For example:-

Consider the problem of implementing a pascal compiler on a new piece of H/w (a computer called X) on which assembly language is the only programming language already available. Under these circumstances, the obvious approach is to write the pascal compiler in assembler. Hence, the compiler in the case is a program that takes pascal source as input, produces machine code for the target machine as output & is written in Assembly language of the target machine.



5)



In this figure moduleI, display a high-level language program which pass to X's compiler for the generation of X's computer code.

The plus point on any other machinery in which module ~~&~~ 1's compiler is not available.

6) The difference between Normal compiler & cross-compiler is generation of codes.

3) Boots Trapping :-)

1) It is a concept of developing a compiler for a language by using subsets (small part) of the same language.

2) It is a technique which program is divided in several parts.

3) These parts are pass into system where high-level language program is available & these parts are translated one by one & the program which translate source program is worked as compiler for another program. This program will be transferred to of a computer for other system.

C, PASCAL, LISP Compiler are generated through Boot trapping.

4) Firstly, PASCAL was first implemented by writing a computer in pascal itself. This was done through several Boots trapping process

Compiler Design Phase :-

The compiler being a complex program is developed through several phases.

Each phase transforms the source

program from one representation to another.

The tasks of a compiler can be divided very broadly into two sub-tasks.

① The analysis (विश्लेषण) of a Source program.

② The synthesis of the object program.

① The analysis of a source program (संकलन, मिलावट)

In a typical compiler, the analysis task consists of 3 phases.

① Lexical (शाब्दिक) analysis.

② Syntax (वाक्य [illegible]) analysis.

③ Semantic analysis. (language की उत्पत्ति)

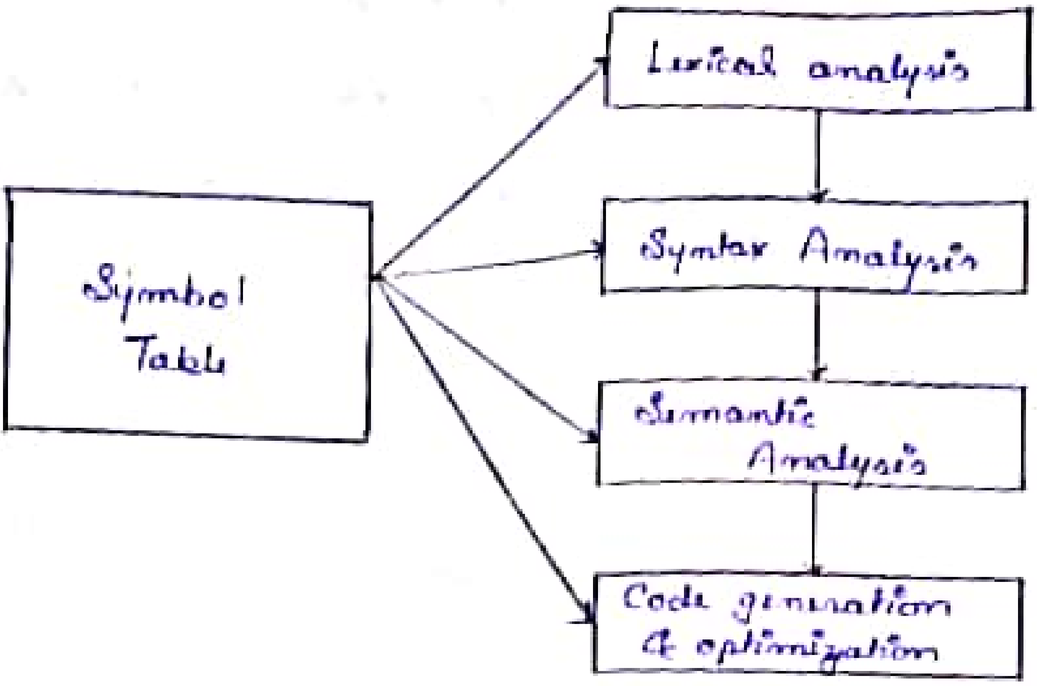
② The synthesis of the object program:-

The synthesis task is usually considered as a —

a) Intermediate code generation.

b) Code optimization.

In typical only code generation exist.



Symbol Table :- It is a set of locations containing a record for each identifier with field for the attributes of the identifier (variable).

A symbol table allows us to find the record for each identifier (variable) & to store or retrieve data from that record quickly.

Lexical Analysis :-

Lexical Analysis is first phase of a compiler. Lexical Analysis is. also called Scanning, scan a source program from left to right. Character by character & group them into token having a collective meaning. It has two main function :-

(1) Generation of tokens, based on regular expression. These token are identifiers, keywords, Multicharacter operator. (<=, etc) (if, then, else) in the program.

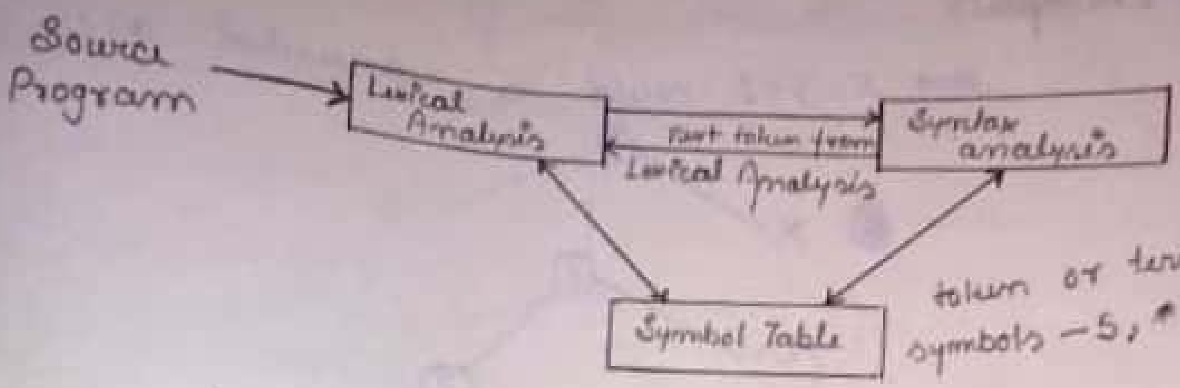
Tokens are cohesive (जोनेवाला) chain by word (प्रमाण चिन्ह), variables & multi-character operator etc.

(2) Storage of missed or not available keyword in symbol table.

There are two optional function of Lexical analysis.

a) Remover of all tables, comments spaces used in.

b) Generation of error message if occurs.



token or terminal symbols — 5, *, 8, −, 6, +, 2.

I Syntax Analysis Phase :-)

(वाक्य अनुभाग)

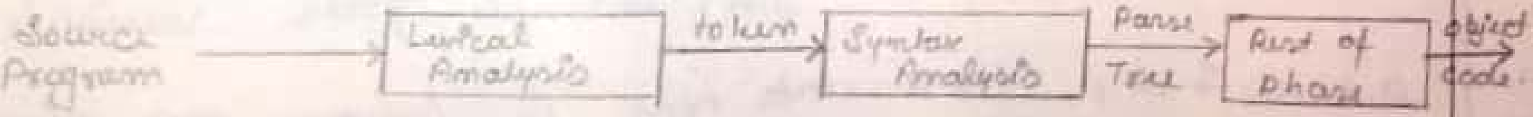
It is 2nd phase of compiler design. It has three main functions.

* Get token for lexical analysis phase.

* Check this token with it own grammer set if no grammatical error mistakes. Convert it into Parse Tree.

Parse Tree :-) The whole working of syntax analysis is called Parsing & the structure of working is called Parse Tree.

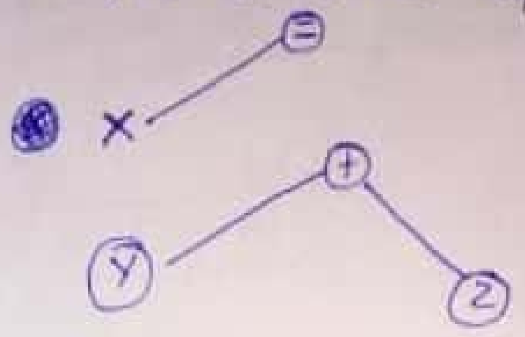
* Generate error message based on syntax error.



(Process of passing).

For Example:-

~~✱~~ X=Y+Z could be represented as:-



Parse Tree.

Parse Tree of the statement in this form means that the first Y & Z will be added & then it's result will be assigned to X.

III Semantic Analysis:-

(language की उत्पत्ति)

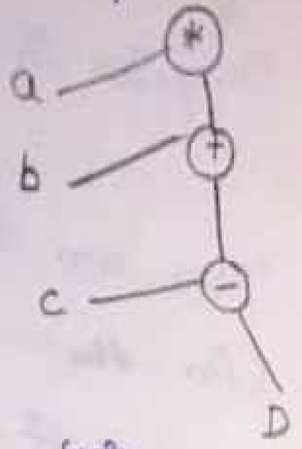
It is 3rd phases of compiler design. It's main function is to check semantic error in the source program, also datatype information of variable used in source program.

It's main components type checking example

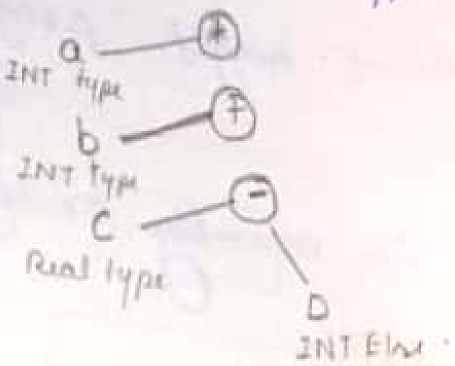
a * b + c - d

In example four variable a, b, c & d, whose data type is integers are given

The Parse Tree of this expression is.

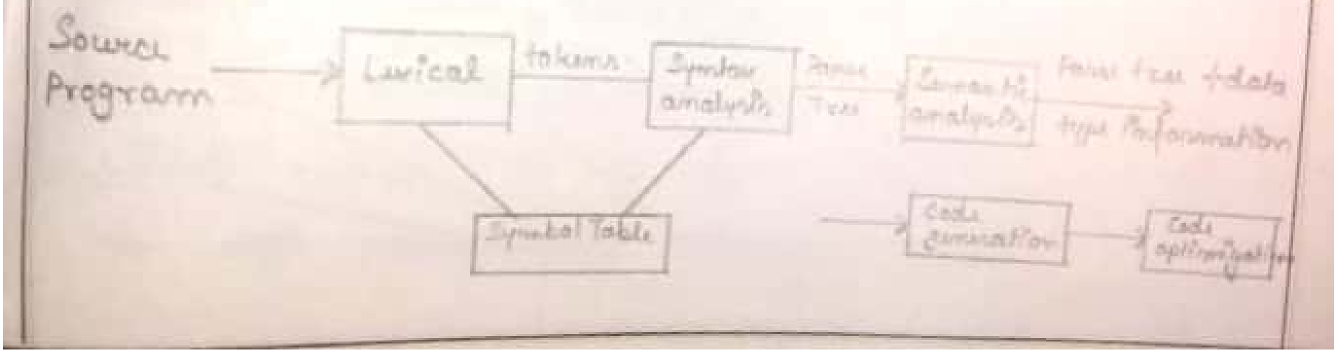


Parse Tree with data type information is:



A the resultant parse tree with type checking, which will be passed to the intermediate code generation phase.

The result of semantic analysis is parse tree with data type information.



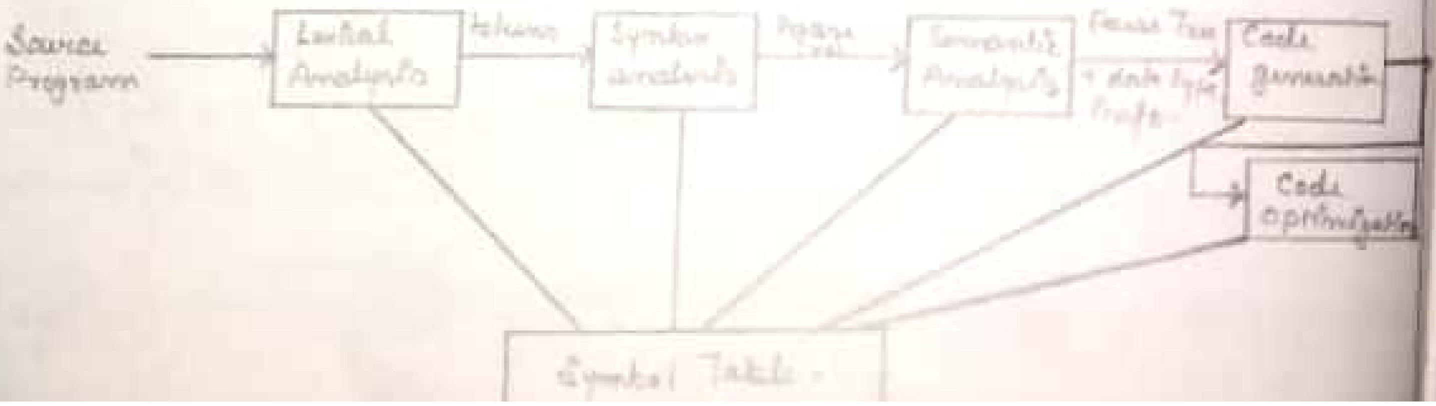
IX Code Generation & Optimization :-

The final phase of the compiler is the code generator.

The code generator takes an input as intermediat representation (in the form of parse tree) of the source program & produces as output an equivalent target program.

Generates a target program is assembly or high-level language.

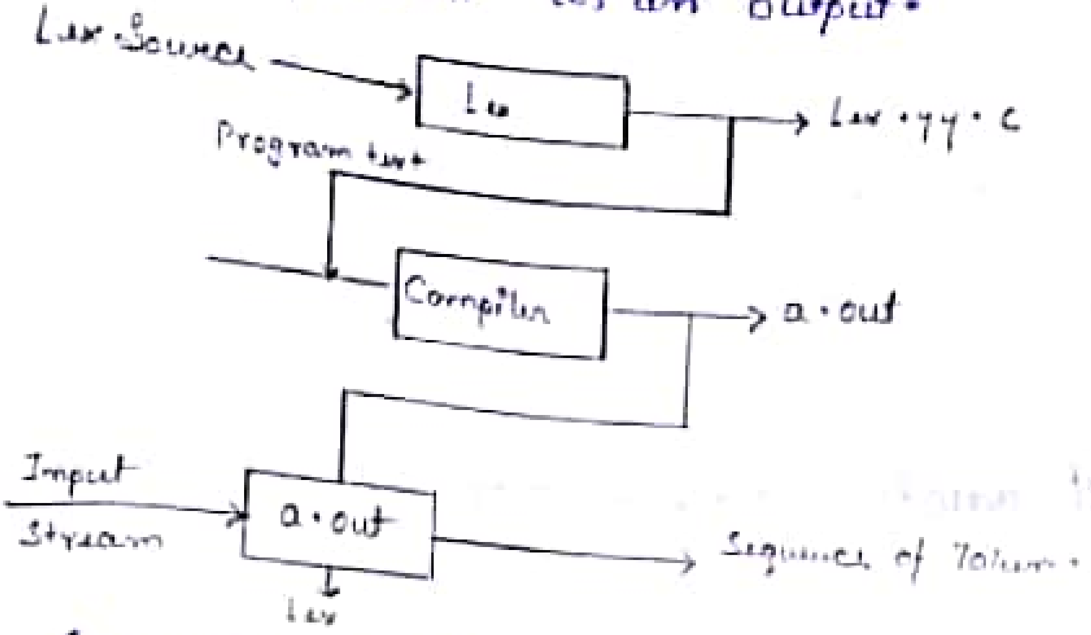
Code optimization is a technique through which object program will be compactal with the help of specialized registers for the optional use of program. ~~In~~ It also include memory address registers (MAR) etc.



Lex :-)

1) Lex is a S/w tools, which is used in Lexical analysis phase of compiler development phase.

2) It's Input is a specification of source program based on regular expression & it gives the sequence of token as an output.



(creating a Lexical analyzer with Lex).

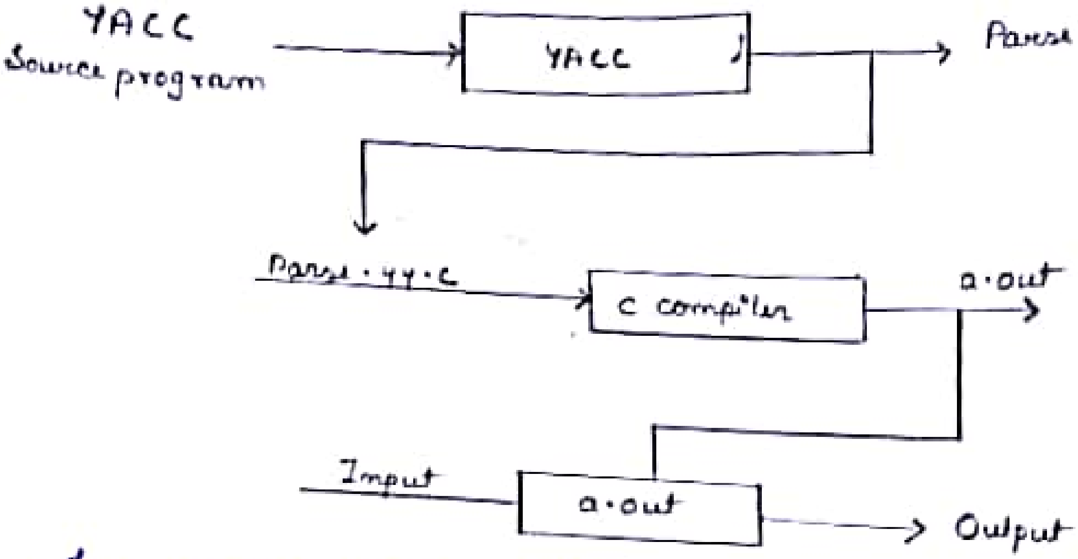
In above fig. Lex Source program Lex.l is a specification of source program as a set of regular expression & also the actions which is to be taken on the source program lex.yy.c

This output of C program contains recognizer for ~~diff. programming languages~~ regular expression with the user code. Lex.yy.c passed in the C compiler to generate object program which is a lexical analyzer, that transforms an input stream as a sequence of token.

* Lex is a widely used Lexical analyzer for diff. programming languages.
* It is S/w tools which takes input as specification of regular expression which one use for the generation of tokens.

* YACC (Yet another compiler compiler).

* YACC is used in syntax analysis phase.
* It takes input as a YACC specification of source program & gives the result in parse tree from which is use in next phase.
* YACC provides a general tool of input to describe computer program.



In above fig. YACC source program pause.yy.c passed into YACC compiler & it generates a C program which is a specification of YACC source program parse.y written in C program. This C program passed into C compiler for the generation of object program which is base based on Parse Tree.