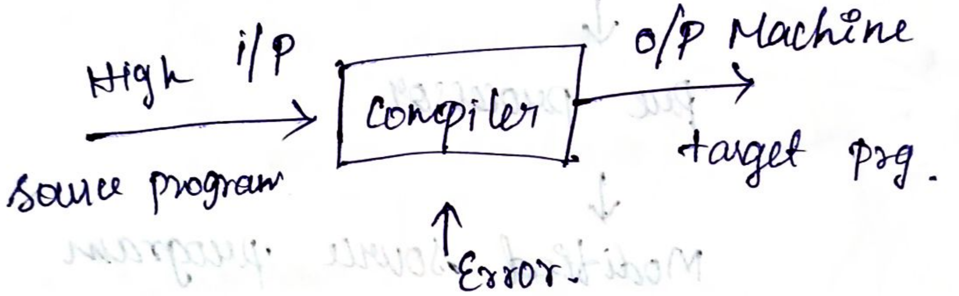
Compiler Design.

High level language → developed by humans & readable by humans

Machine level language → Not readable by humans.

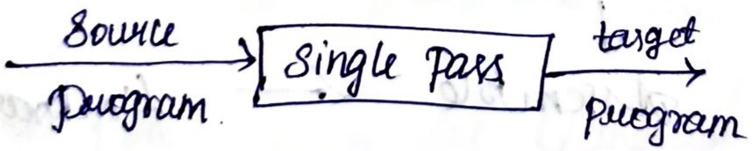
Compiler :-

It is a software which converts high level lang to Low Level language.

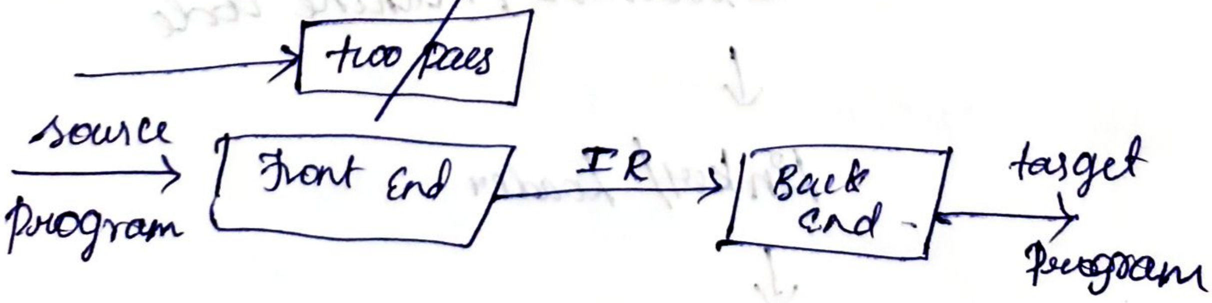


Types of Compiler :-

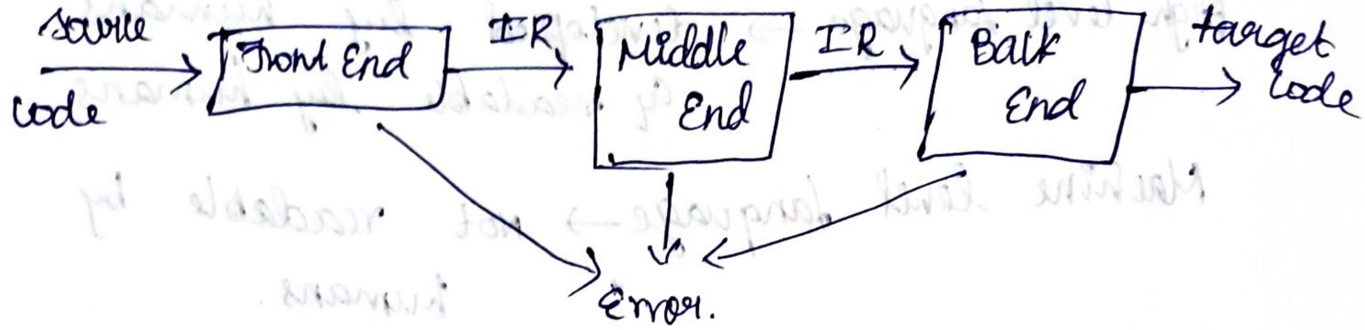
Single pass:



Two pass:



Multi pass :



Language processing system :-

Source program

↓

Pre processor.

↓

Modified source program

↓

Diff ← Compiler. / Interpretor.

↓

Target Assembly program

↓

Assemble ← Limbrary files

↓

Relocatable / Machine code

↓

Linker/ Loader

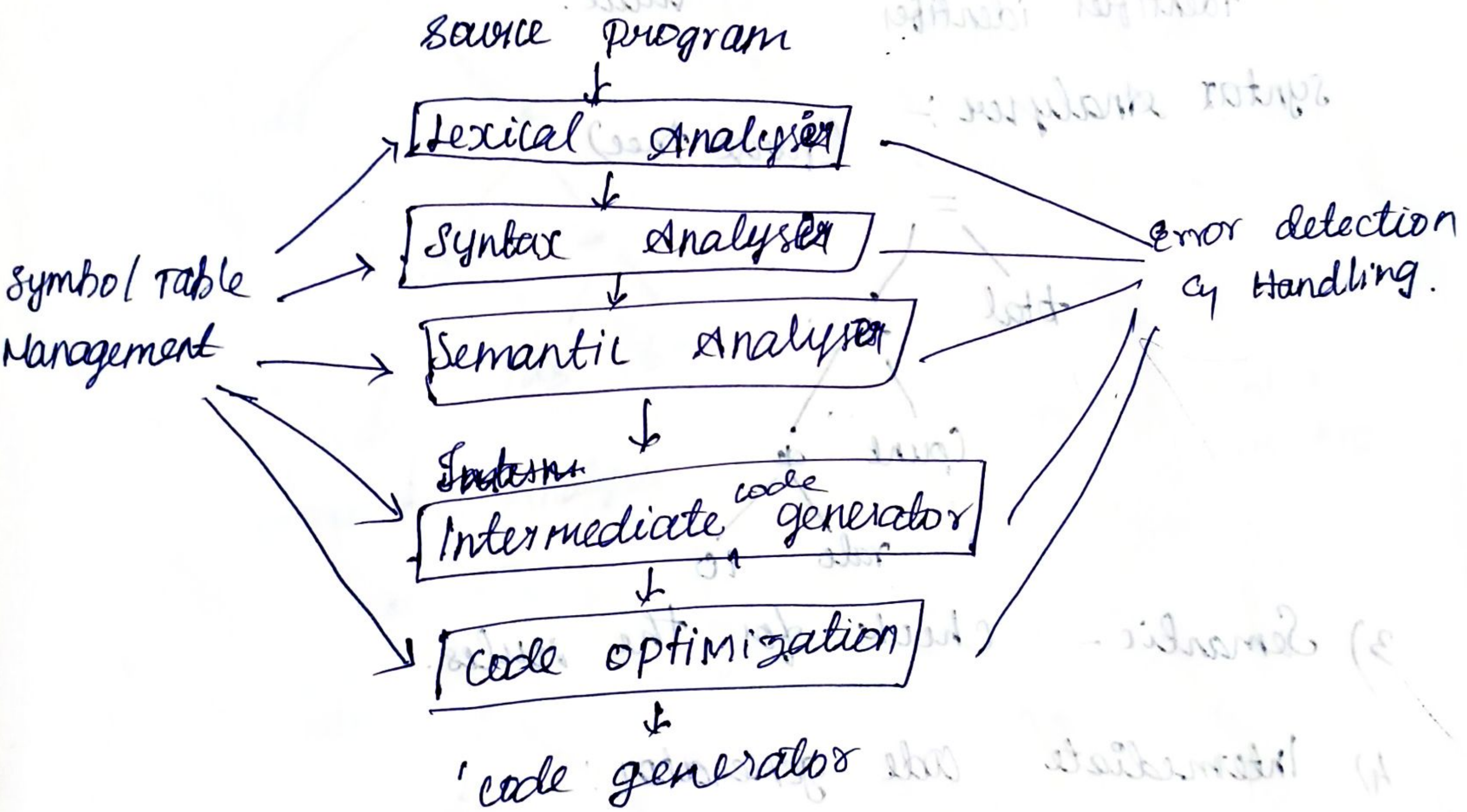
↓

Target Machine code.

Cousins of Compiler ÷

- → Compiler
- → Loader
- → Interpreter
- → Assembler.
- → Preprocessor

Phases of compiler :-



Analysis part : Lang. dep (Language dependent) Machine independent

Synthesis part : Mac. dep

Total = count + rate * 10

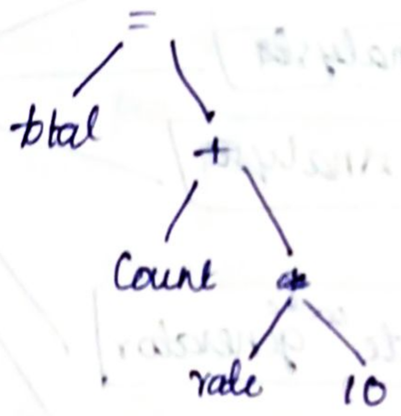
Lexical Analyer :-

It slan source program and gives sequence of strings.

Total = count + rate * 10

↓ identifies, = ↓ operator, count ↓ identifier, + ↓ operator, 10 ↓ value.

Syntax Analyser :- (parse tree)



3) Semantic - check for the rules.

4) Intermediate code generator :

- quatriple
- triple
- posix

5) code optimizalor

6) code generator (Assembly langs)

1385
1303

a = b+c * 60.

↓

Lexical Analyzer

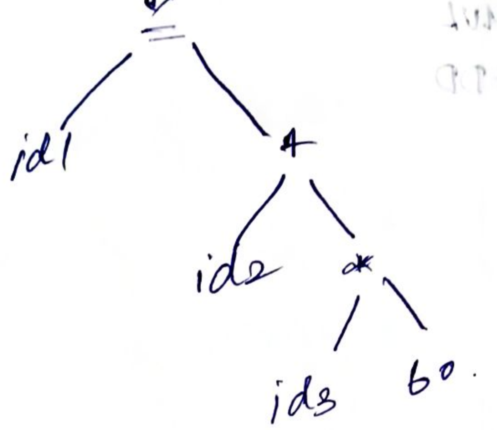
↓

id1 = id2 + id3 * 60.

↓

Syntax Analyser

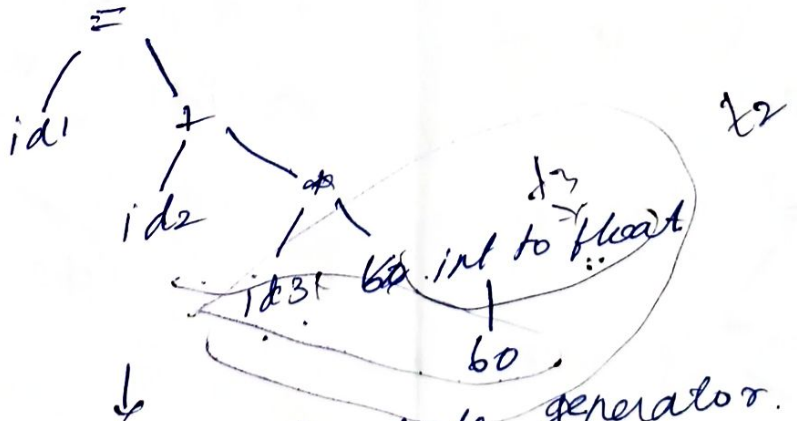
↓



total = count + rate *10

↓

Semantic Analyser.



↓

intermediate code generator.

t3 = int to float (60)

t2 = id3 × t3

t3 = id2 + t2

id1 = t3.

↓

optimize

optimize

t1 = id3 * float(60)

t1 = id3 * 60.0

id1 = id2 + t1

↓

Code generate

MOVF id3, R2

MULF #60, R2

MOVF id2, R1

ADDF R1, R2

MOV

MUL

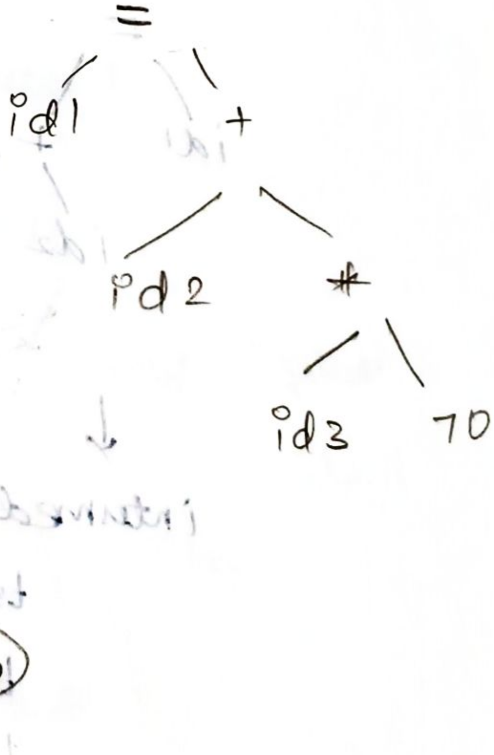
ADD

total = count + rate * 70.

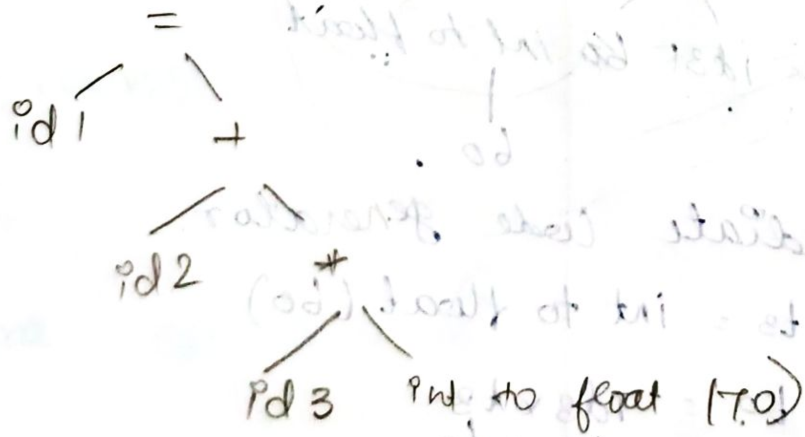
lexical

id1 = id2 + id3 * 70

syntax



seemantic



In code generator. Ei. in' to float(70)

50)

t1 = int to float (70)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3

code optimizer

t1 = id3 * 70.0

id1 = id2 + t1

MOVF id3, R2
MULF #70, id3
movF id2, R1
ADDF R1, R2.

code generator

MOVF id3, R2.

MULF #70, R2

MOVF id2, R1

ADDF R1, R2.

M.

M

unit - II (Lexical Analyzer) → wht &

↳ role.

↳ Terminologies.

DFA - NFA

$\Sigma = (0\ 1)$ ↳ regular expr

DFA to NFA → tokens uses.

Regular expression → NFA → Regular expression.

NFA to DFA → Regular expression to NFA.

E-NFA Kleens theorem.

DFA - Minimization. identifies recogonize pandhalthun regular expression

Lang processing Model. unit - I (PT)

Assembler, preprocesser, Linker, loader, diagram (intrepter, compiler diff)

lexical anlaysis convertion (phases of compiler (A= B+ C*60)

Types Of Compiler

Backend, front-end, problms in single pass. } → diagrams.

Language processing Model.

cousins of compiler -

Assembler, compiler, loader → diagramatic ~~ele~~ representation

unit - III Semantic analyser
syntax

→ Grammer, CFG convertion

→ left, right most derivation :

→ Parsey tree. from Given grammer or input string.

→ TOP, down pariag

→ Bottom, ~~down~~ up parsing. → converting data from one format to another.

Grammer - used to derive some language.

G = { V, T, P, S }.

- V ↓ Non Terminal (capital letters)
- T ↓ Terminal
- P ↓ production rule.

TOP down parsing :-
↳ process of converting or transfering data from one part or format to another.

↓ TO select correct appropriate production rule.

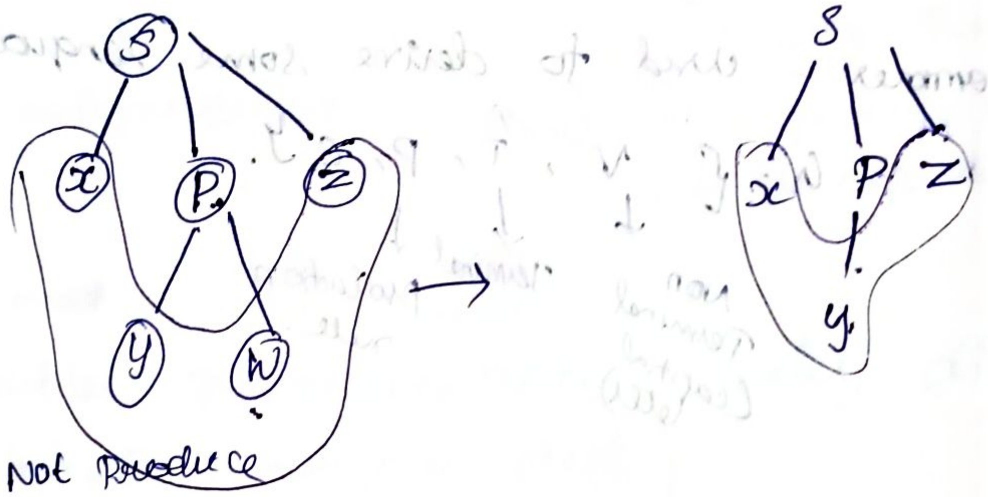
- → TOP down parsel (Recursive decent)
  - → Backtracking
  - → Predictive
- → Bottom up parsel.
  - → operator precedence
  - → LR parcel
    - → SLR
    - → LR → LR(1)
    - → LALR

Top down parcel (Back tracking)

from root to children or leaf.

$S \rightarrow x P z$ i/p : xyz

$P \rightarrow (yw) / y$..



left recursion

$A \rightarrow A\alpha / B$

$\left\{ \begin{array}{l} A \rightarrow BA' \\ A' \rightarrow \alpha A' / \varepsilon \end{array} \right\} \rightarrow$ proof

$S \rightarrow ABC$

$A \rightarrow Aa / Ad / c$

soln :

$S \rightarrow ABC$

$S \rightarrow Aa / b$

$A \rightarrow bA'$

$A' \rightarrow aA' / \varepsilon$

(implementing proof)

$A \rightarrow Ad / b$

$A \rightarrow bA'$

$A' \rightarrow bA' / \varepsilon$