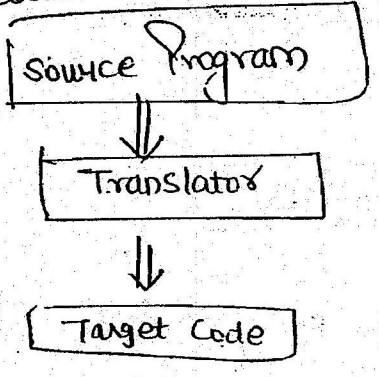
③. Search in google

Translator – :. Translator is a program, it converts one programming language code into another language code.



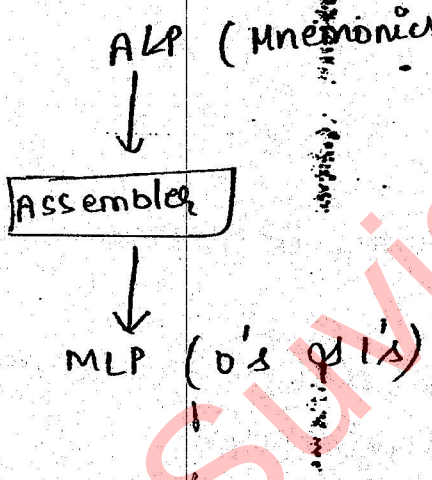
Input of the Translator are called source code whereas output of the Translator are called Target code or object code.

Based on inputs & outputs of the translator, they are classified into - 3.

1. Assembler
2. Compiler
3. Preprocessor.

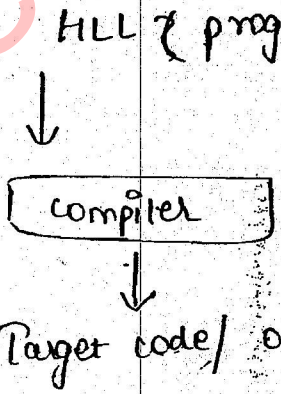
★ ASSEMBLER

Assembler is a translator which converts asemly language codes (Mnemonics) into machine language code (0's & 1's).



★ Compiler

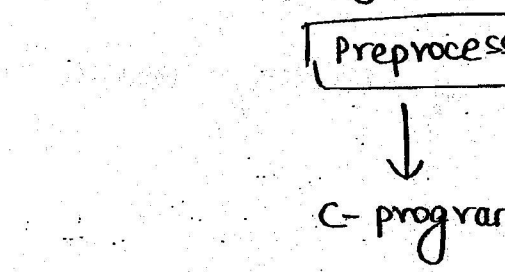
Compiler is a translator which converts High level languages into object code or target code.



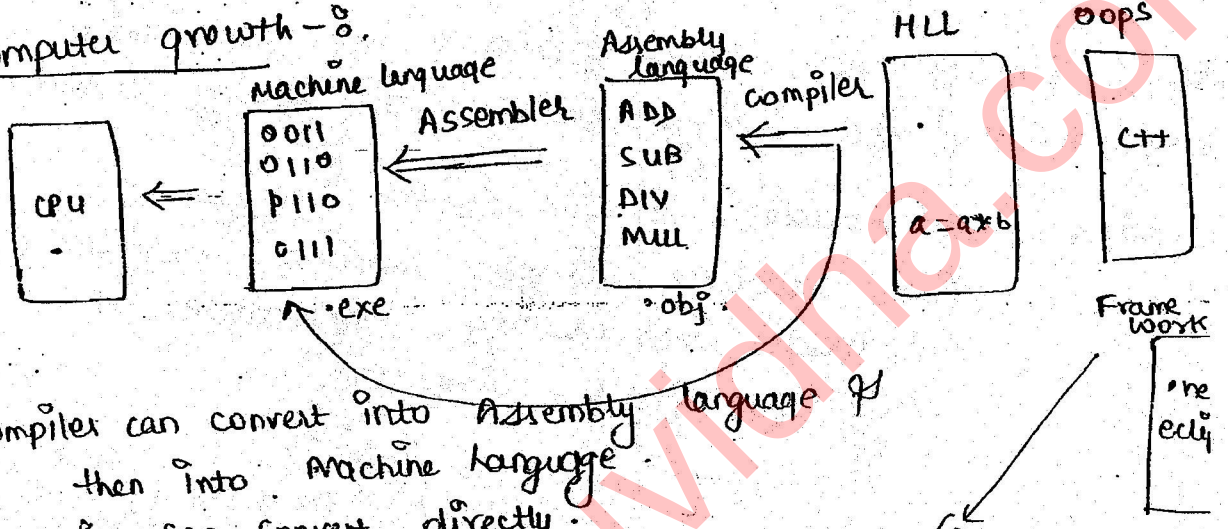
★ PREPROCESSOR

Preprocessor is a translator which converts one HLL large program into another HLL.

eg. C++ program



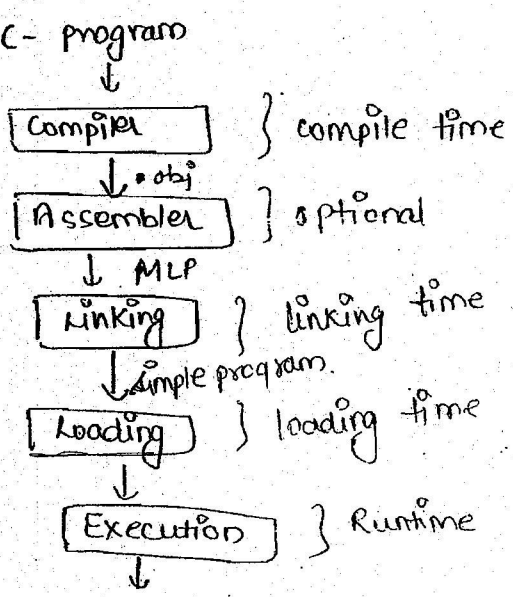
* computer growth - :



Compiler can convert into Assembly language & then into Machine language.
Also it can convert directly.

(jo apne ap structure de dete hai structure yaad rakhne ki jrut nai pad

* Language Processing System - :



* LINKER –: Linker is a system program that attaches all other modules to the object code and will produce single module output program.

There are 2 types of linking operations –:

1. Static linking.
2. Dynamic linking.

Static linking will takes place after compilation whereas Dynamic linking takes place during Runtime.

* Loader –: Loader is a system program that accepts single module program and loads into memory ~~for~~ & make it ready for further execution.

Compile time + Link Time + Load time + Runtime

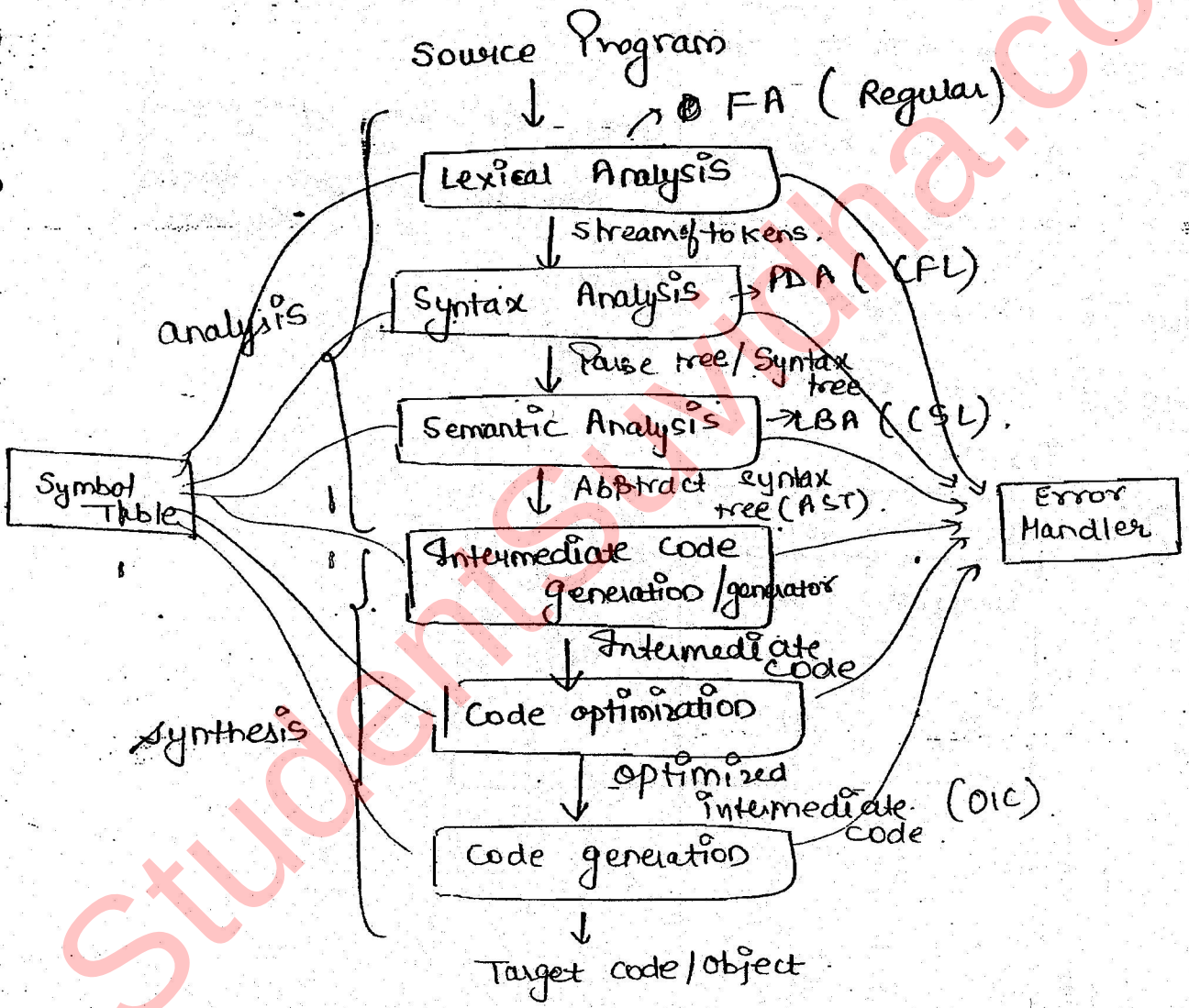
1.	FC + DE + EX
2.	FC + DE + EX
3.	
4.	

(Fetch decode execute)

C, & Java is application program (language to develop program).
Linker, Loader, Compiler, Assembler are all system program. (which help to develop our program).

✱ COMPILER PHASES —%.

Compiler is a translator that consists of several modules. Each module is called as phase of the compiler. In other words, the compilation is a very complex process, partition into several subprocesses, Each subprocess is called as phase of the compiler.



* LEXICAL ANALYSIS

Altname - : token recognizer / scanner;

function - : token recognization.

i.e. it read source statement character by character and breaks the statement into stream of tokens.

ip - : Source statement.

o/p - : Streams of tokens.

token - : A basic building block of any programming language. (keyword, operator, variables, special symbols).

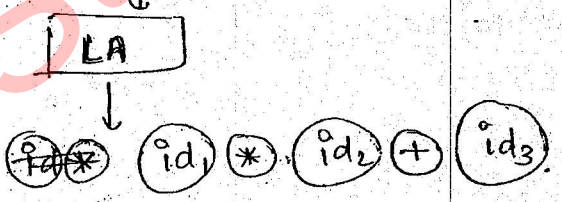
lexeme - : Value given to token.

Pattern - : Rules follow to design lexeme.

↳ eg. int abc;

token	lexeme	pattern
identifier	abc	letter followed by any letter or digit.
int	int	int.

eg. A * B + C



* Syntax Analysis

Alt. name = Parsing

function - : checks syntactic structure of the statement.
i.e, it reads streams of tokens and gpc thems into syntactic structure such as statements, expression etc.

ip - : Stream of tokens
o/p - : Parse tree or syntax tree

* Semantic Analysis.

Alt name - : Syntax directed translation. (SDT).

function - : It checks the meaning of the statement
i.e, it checks whether statement is fit for execution or not.
Main function - : Type checking.

ip : - Parse Tree
o/p : - Abstract syntax tree (AST)

* Intermediate code generator. (ICG).
The intermediate code generator translates AST into some intermediate forms such that the producing of Target code process becomes simple and we can also improve quality of the object code.

I/p ip - : AST
op - : Intermediate code form.

most commonly used intermediate code is 3- address code.

General form ⇒ $A = B \; op \; C$

where A, B, C — operand
op → operator.

(3 address → almost 3 operands must be there.)

eg. $a = b * c + d$

$t_1 = id_2 * id_3$

$t_3 = t_1 + id_4$

$a = t_3$

{ Each and every operation must have 1 statement }

eg. $x = x * 1$

$t_1 = id * 1$

$id = t_1$

eg. $a = -b * c$

$t_1 = -id_2$

$t_2 = t_1 * id_3$

$a = t_2$

* Code optimization

It accepts intermediate code as input and outputs optimized intermediate code.

The main objective of optimization is improving quality of intermediate code such that it occupies less space in memory, less time for execution.

* Code generator.
* Code generation is the toughest job as designer should know about architecture of all machines as compiler is not only limited to single machine.

It accepts optimized intermediate code as input and will outputs object code or target code as.

i.e, Compiler output is .obj file not .exe.

The most difficult phase among all is code genrati both practically and theoretically.

* Symbol table –

Symbol table is a datastructure that records the information about all identifiers used in the progra along with their attributes.

eg. B1
{ int a, b, c;
By
}

attributes

id	data type	size	scope
a	int	2	B1 – By
b	int	2	B1 – By.
!	!	!	!

Same name se jo aega like array, structure, function, variable, then error dega.

* Error Handler –

Error handler diagonis the error in each phase and reports them to the user by calling appropriate error recovery routines.

Symbol Table is associated with each phase to ensure that all attributes values needed by some phase at a time is available.

* In some cases, the compiler consists of 2 parts
  ① Analysis.
  ② Synthesis.

In analysis phase the source program will be converted into Intermediate code form.

In synthesis phase, the intermediate code form will be converted into target code.

Symbol Table & error handler are part of both families

* In some of the cases, compiler treated as 2 ends, one is - front end and backend.

① front end - :- The front end of the compiler mainly depends on source program. It includes Lexical analysis, Syntax Analysis, Semantic Analysis, Symbol table and error handler.

② Back end - :- The backend of the compiler mainly depends on target machine and independent on source code. It includes code optimizer, code generator, Symbol table and error handler.

Phase - :- A phase is a each module in a compiler program.
Eg. lexical analysis.
Syntax analysis
Semantics analysis etc.

Pass - :- A pass is group of modules, when all modules of compiler are grouped as one called single pass compiler otherwise

is called as multiparse compiler.

A single parse compiler occupies more space in the memory but takes less time for compilation. Where a multipass compiler occupies less space in the memory but it takes more time for compilation

↓

due to swap in, swap out time in mem

★ COMPILER VS INTERPRETER.

COMPILER	INTERPRETER
① It is a translator	① It is not a translator
② It converts a source statement into object code.	② It produce a result for each statement.
③ Compiler uses 2 times (Compile time + execution time) to produce a result.	③ It produce a result with runtime.
④ Less user friendly.	④ More ~~eqfriendly~~ user friend as it consider line by line.
⑤ Suitable for HLL.	⑤ Suitable for command based languages.

Q. Which data structure in a compiler is used for managing information about variables and their attribut

(a) AST (b) Symbol Table (c) Semantic Stack

(d) parse Table.

Q. One of the purpose of using intermediate code in a compiler is true?
GATE 2014 Set 3.

(a). Making a parsing and semantic analysis simpler
(b). Improves error recovery and error reporting.
(c). Increases chances of reusing code optimization techniques. to produce optimized code.
(d). Improves the register allocation.

Q. In a compiler Keywords of a Language are recognized during
GATE 2011

(a). parsing of the program
(b). Code generation
(c). Lexical analysis of program
(d). Data flow analysis.

Q. Consider a program P that consists of 2 source modules M_1 and M_2 containing 2 different files. If M_1 contains reference to a function defined in M_2, the reference will be resolved at -
GATE 2010

(a). edit time
(b). Compile time
(c). Link time
(d). Load time

Q. Type checking is normally done during
(a). Lexical analysis
(b). Syntax Analysis
(c). Syntax directed Translation
(d). Code optimization

Q. A compiler of a HLL that runs on 1 machine and produce a code for different machine is called (a) optimizing compiler
(b) one pass compiler
(c) cross compiler
(d) Multipass compiler.

Cross compiler – : runs on 1 machine and produce a code for different machine

optimize compiler – : optimize the time.

Q. Which translator program converts Assembly language program to object program?
↳ common term also for machine code.
(a) Microprocessor
(b) Linker
(c) Assembler
(d) compiler.

Q. In a compiler module that checks every characte of source text is called
G1988
(a) code generator
(b) optimizer
(c) scanner
(d) syntax analyser