Camlin | Page
Date / /

Questions Definations

(I) Lexeme (II) Pattern (III) Token (IV) Linker (V) Loader
(VI) Pre Processor (VII) Interpreter (VIII) Assembler

1. Lexeme → Sequence of characters in the source program that are matched with the pattern of a table.
Eg. main is lexeme with type identifier.

2. Pattern → Set of rules that scanner follows to create a token.
Eg. For identifier to be identified as a valid token, the pattern is the pre defined rules that it must start with alphabet followed by alphabet or a digit

3. Token → A token is a pair consisting of a token name and an optional attribute value. Token name is an abstract symbol representing a kind of lexical unit.
Eg. Keyword or sequence of input characters denoting an identifier.

4. Linker :→ It is tool which is used to link all parts of the program together for execution. All these files might have been compiled by different assemblers.

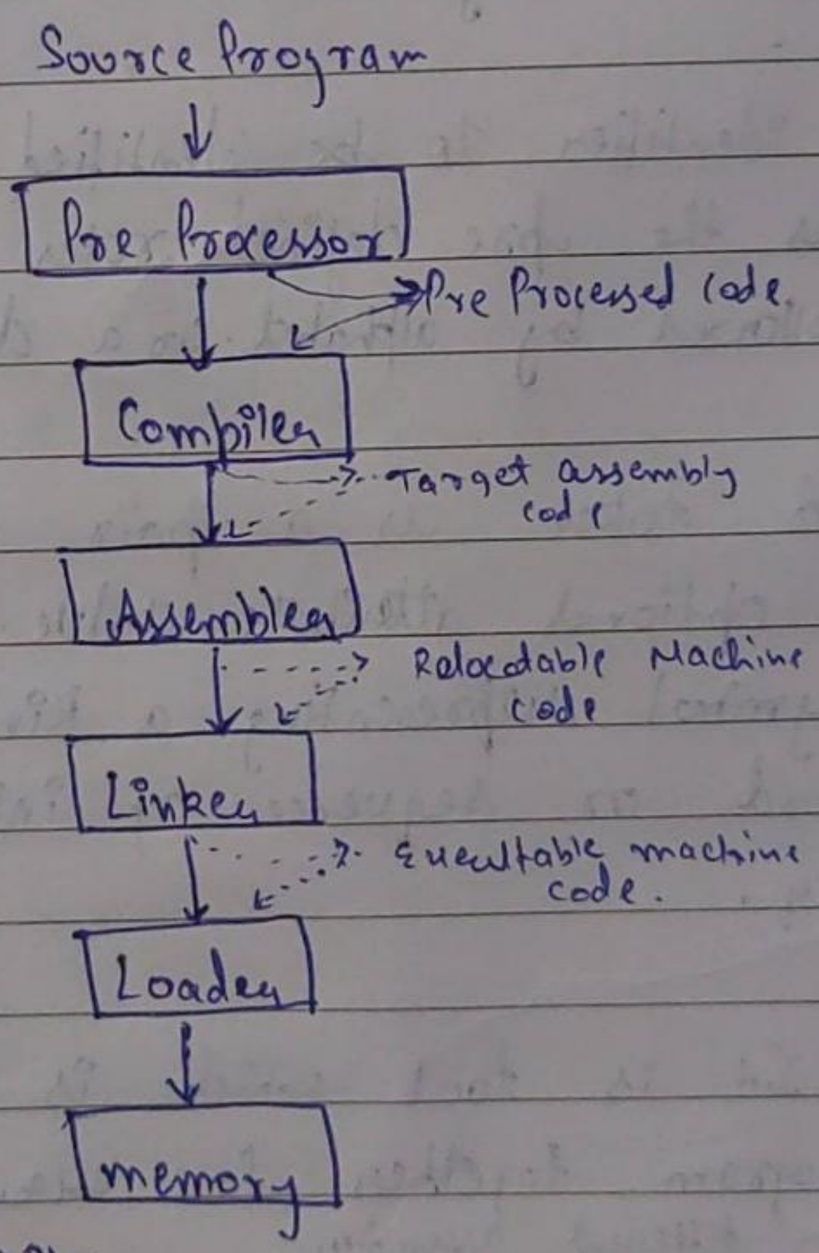
5. Loader → It loads all the program into memory and then the program is executed. It calculates size of a program and creates memory space for it.

6. Preprocessor :→ Generally considered as a part of compiler, is a tool that produces input for compilers to produce output that is used as input to another program. The output is said to be a preprocessed form of the input data.

Scanned by TapScanner

(vii) Interpreter :→ An interpreter, like a compiler, translates HLL to LLL. The difference lies in the way they read the input. An Interpreter reads a statement from the input, converts it to an intermediate code, executes it, then takes the next statement in sequence. If an error occurs it stops execution & reports it.

(viii) Assembler :→ It translates assembly language programs into machine code. The output of an assembler is called an object file.

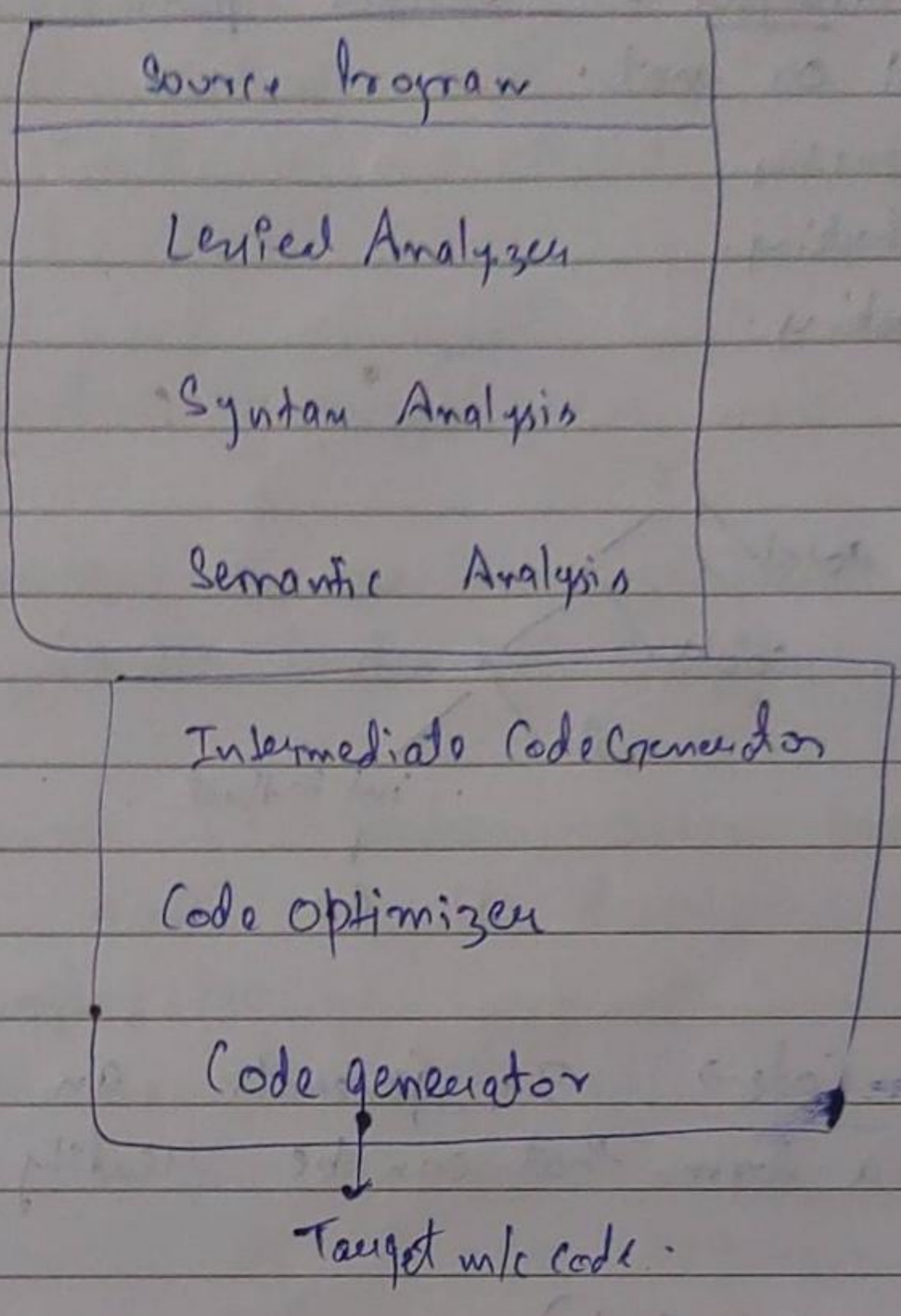


Properties of Compiler

(i) It must be bug free
(ii) Generate correct machine code
(iii) Have consistent optimization
(iv) Must be portable.
(v) Must run fast.

Scanned by TapScanner

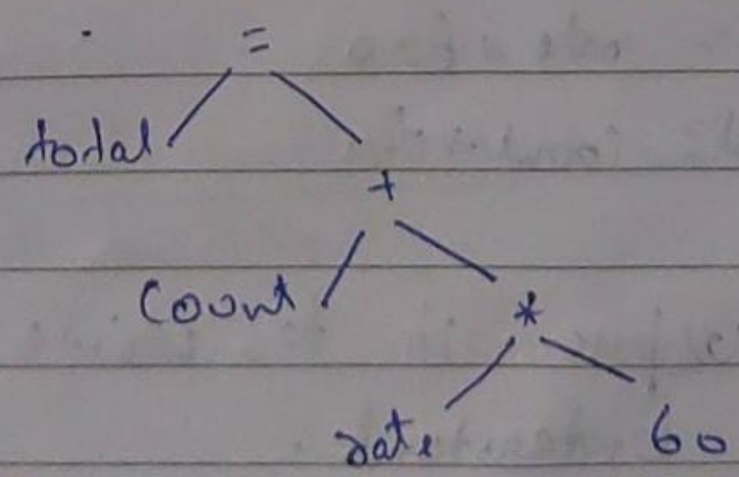
Phases of Compiler



(I) Lexical → It is also called as Scanner.
It reads character from the source program & group them into lexemes.

total = count + rate * 60

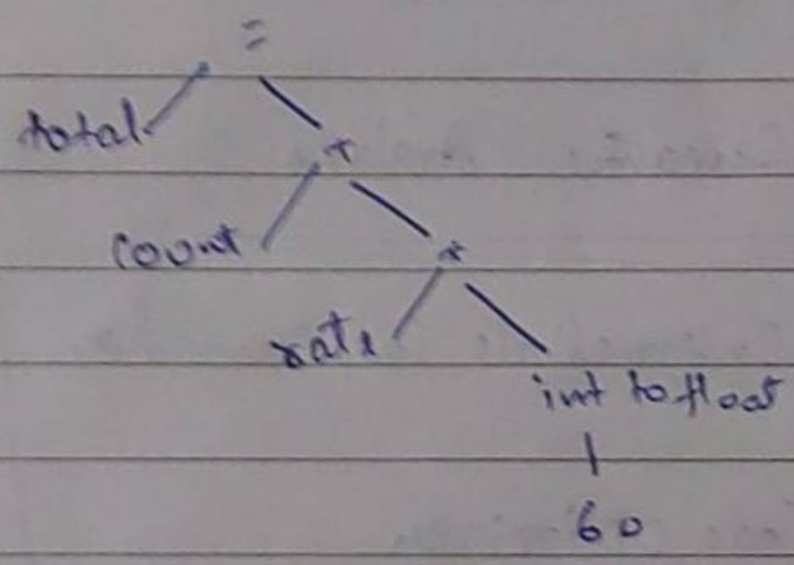
(II) Syntax Analyzer → It constructs a parse tree. It takes all the token one by one and uses CFG to construct a parse tree.



Scanned by TapScanner

(iii) Semantic Analyzer → Verifies parse tree whether it is meaningful or not.

- Error reporting
- Static checking.
- Name checking.



(iv) Intermediate Code → It generates an intermediate code, which is a form that can be readily executed by a machine.

t_1 = int to float(60)
t_2 = rate * t_1
t_3 = count + t_2
total = t_3

(v) Code optimizer → It transforms the code so that it consume less memory and increase the speed of the code.

t_1 = rate * 60.0
total = count + t_1

(vi) Code generator → Purpose is to write a code that the machine can understand.

Scanned by TapScanner

```
MOV   rate, R1
MUL   #60.0, R1
MOV   count, R2
ADD   R2, R1
MOV   R1, total.
```

Compiler Construction Tools

The compiler writer can use some specialized tool to help in implementing various phases in compiler.

(i) Parser Generator — It produces syntax analyzers from the input that is based on the description of programming language. Example PIC.

(ii) Scanner Generator :→ It generates lexical analyzers from the input that consits of Regular expressions. It generates a FA to recognise R.E. LEX.

(iii) Syntax Directed Trees → It generates intermediate code with three address format from the input that consits of parse tree.

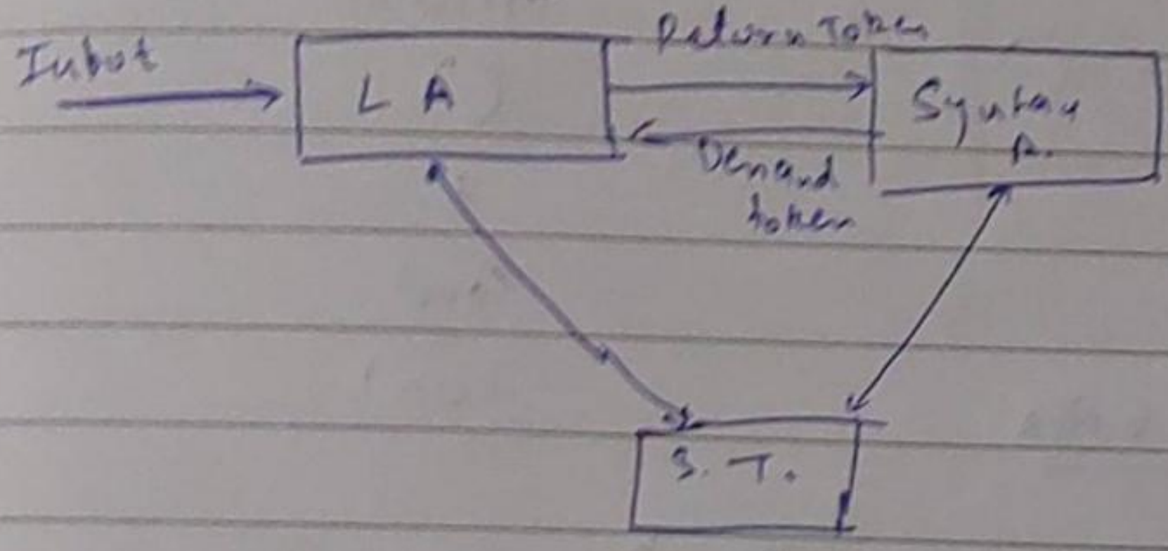
(iv) Data flow engines :→ It is used in code optimization. It is the key part of code optimization that gathers the information, ie., the value that flow from one part of a program to another.

(v) Automatic Code generators :→ It generates the machine language for a target machine.

Scanned by TapScanner

Lexical Analyzer

(Reads characters from source program & group them into lexeme)



Token (Token name, attribute value)

1. Produce string of token
2. Eliminates white space or comments.
3. Generate Symbol table that store information
4. Provide error msg. with line & column no.

Identifier → Starts with letter or underscore followed by digit or special Symbol

```
if (a<10)
    i=i+2;
else
    i=i-2;
```

1, (2,1) (5,100), (7,1), (6,105), (8,2) . . .

Seperation of ~~Syntax~~ LA from SA.

→ Increases Efficiency of compiler
→ Simplify design of a compiler.
→ Increases portability of a compiler.

Scanned by TapScanner

LEX

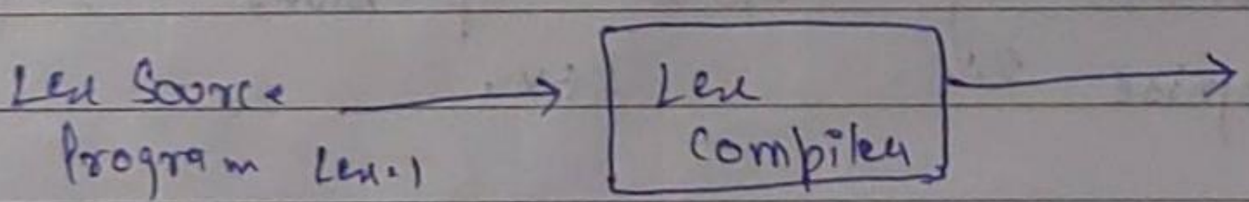
- It is a feature that generates lexical analyzers. It is used with YACC parsers generator.

- Lexical analyser reads characters from the source program and group them into lexeme.

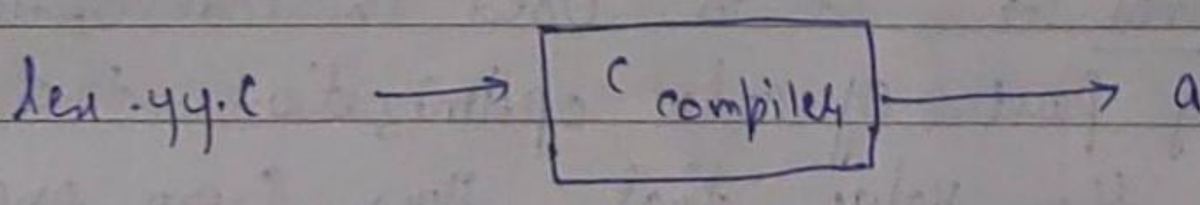
- It reads the input stream & produces source code as output through implementing the LA in C program.

Function

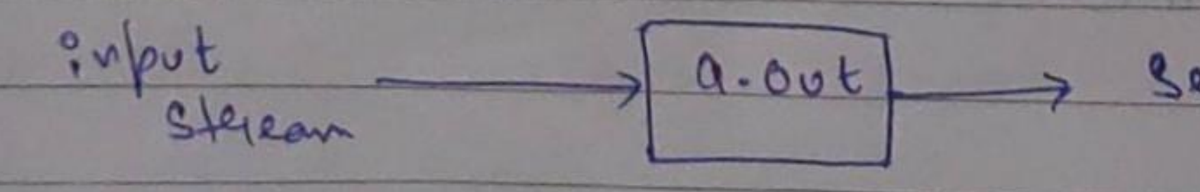
- Firstly lexical analyzer creates a program lex.l in lex language. Then Lex compiler runs lex.l program and produces a C program lex.yy.c



- C compiler runs the lex.yy.c program & produces an object program a.out.



- a.out is lexical analyzer that transforms an input stream into sequence of token.



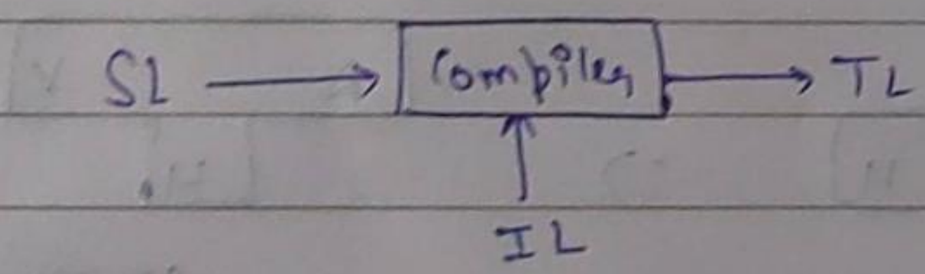
Scanned by TapScanner

Bootstrapping & porting

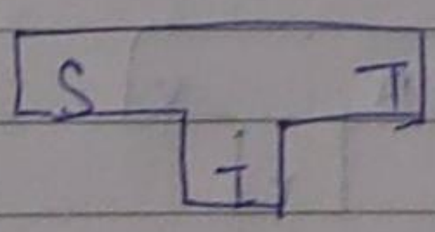
- Bootstraping is used in compilation development.
- It is used to produce a self hosting compiler. ie. which compile its own source code.

A compiler can be categorized by three languages.
- Source Language (S)
- Target language (T)
- Implementation Language (I)

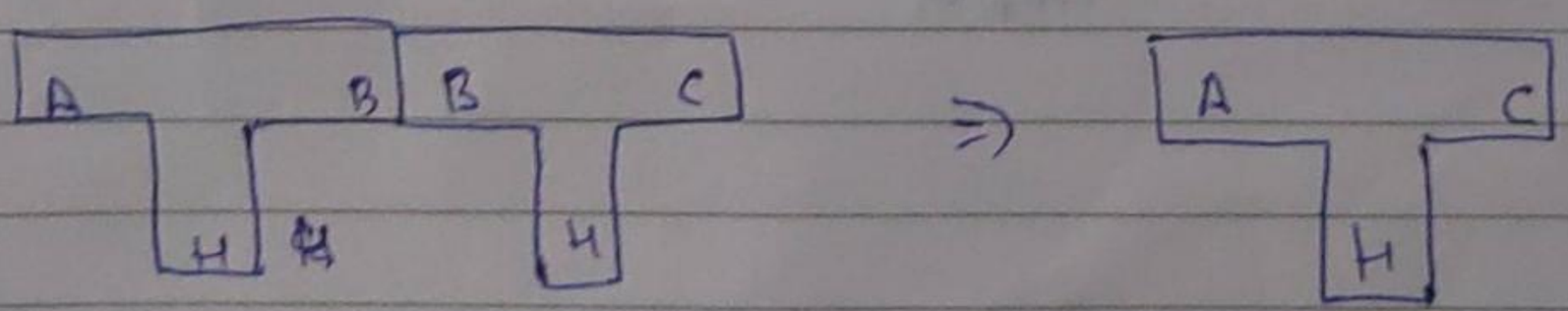
Bootstrapping is the process by which simple language is used to translate more complicated program, which in turn may handle an even more complicated program and so on.



- In bootstraping the compiler diagram is represented as 'T' diagram.

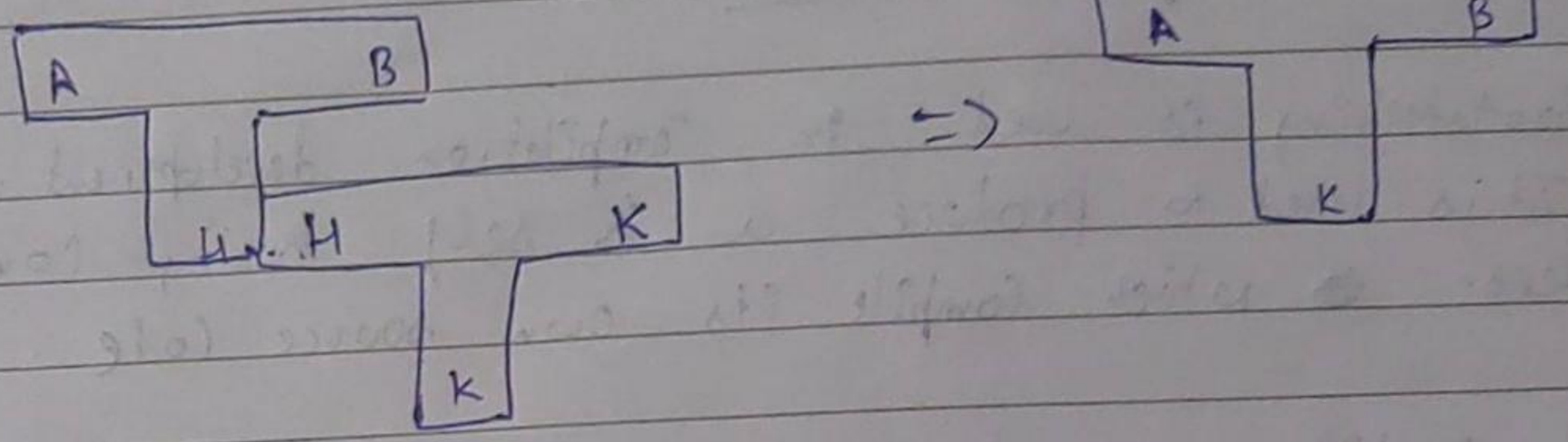


Combination of T diagram

① 

Scanned by TapScanner

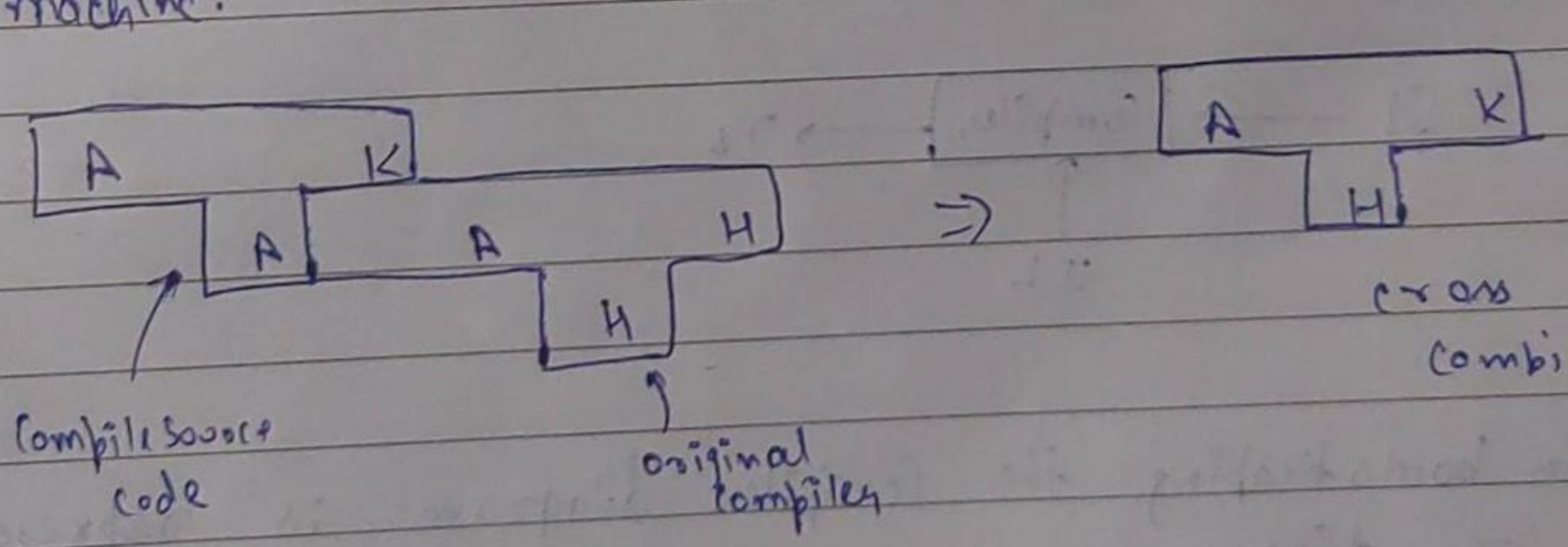
(ii)



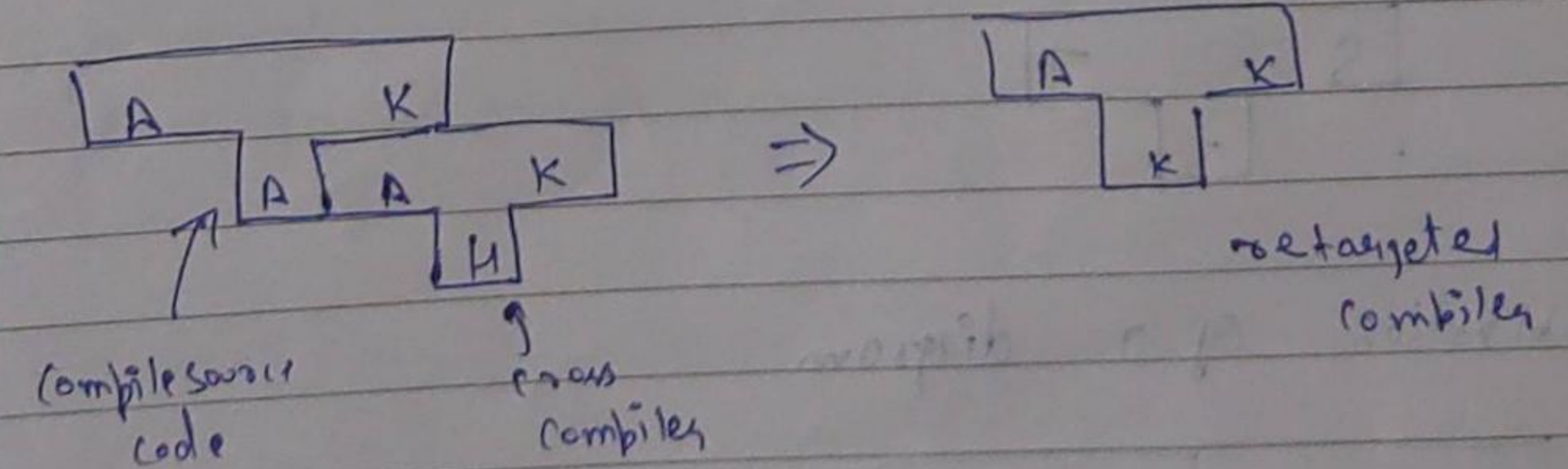
Porting :→ It only requires the back end of the source code be rewritten to generate code for the new machine.

This rewritten code is then compiled using the old compiler to produce cross compiler & compile it again to produce working code for new machine.

(i)



(ii)



Scanned by TapScanner