30/09/20 (1)

COMPILER CONSTRUCTION

ASSIGNMENT : 01

Piyush Agarwal
17BCON173
Sec - D

Ques-1 What is the difference between assembler, Interpreter and compiler?

Ans-1 Assembler → It translates only assembly language code into machine understandable language. The o/p result of assembler is known as an object file which is a combination of machine instruction as well as the data required to stored these memory location where all modules are stored.

Interpreter → It translates high level language into low machine level language. The main difference b/w both is that interpreter reads & transforms code line by line. Compiler reads the entire code at once and creates the machine code.

Compiler → A Compiler is a computer program which helps you transform source code into another language code. It also makes the end code efficient which is optimized for execution time & memory space.

Ques.2 What do you understand by the term cousins of compiler?

Ans.2 Cousins of Compiler include these components :-

(i) Pre-processor → It converts the high level language into pure high level language. It is optional because if any language which does not support "# include" & macro pre-processor is not required.

(ii) Compiler → It takes pure high level langugage as a Input & converts it into assembly code.

(iii) Assembler → It takes the assembly code as Input & converts it into machine code.

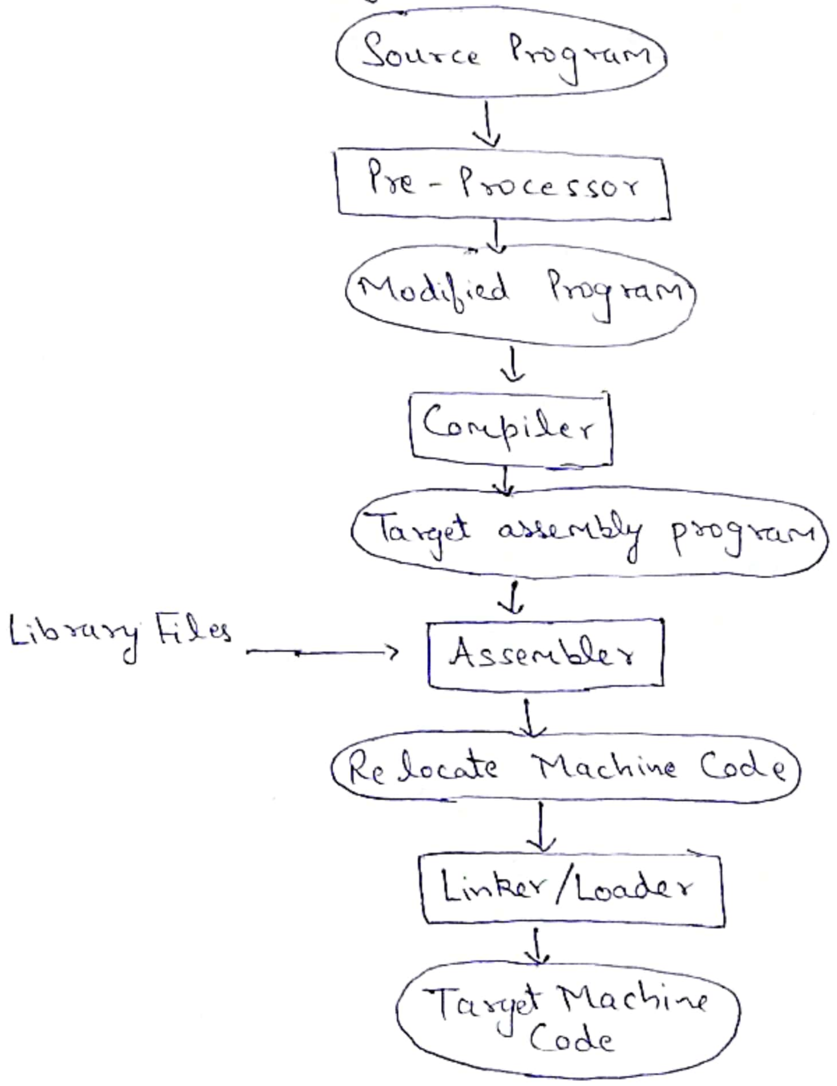
(iv) Linking & Loading → It has four functions :-
(a) Allocation (b) Relocation
(c) Linker (d) Loader

(a) Allocation → It means get the memory portion from OS and storing object data.

(b) Relocation → It maps the relative address to physical address & relocating the object code

(c) Linker → It Combines all the executable object module to pre single executable file

(d) Loader → It loads the executable file into permanent storage.



Ques. 3 What is the difference b/w one pass and two pass compiler ?

Ans. 3 In single pass compiler source code directly transform into machine code. For example :- Pascal language

$$\begin{tikzcd}\text{Source Code} \arrow[r, Rightarrow] & \boxed{\text{Compiler}} \arrow[r, Rightarrow] & \text{Target Code}\end{tikzcd}$$

whereas, In two pass compiler there are two sections :

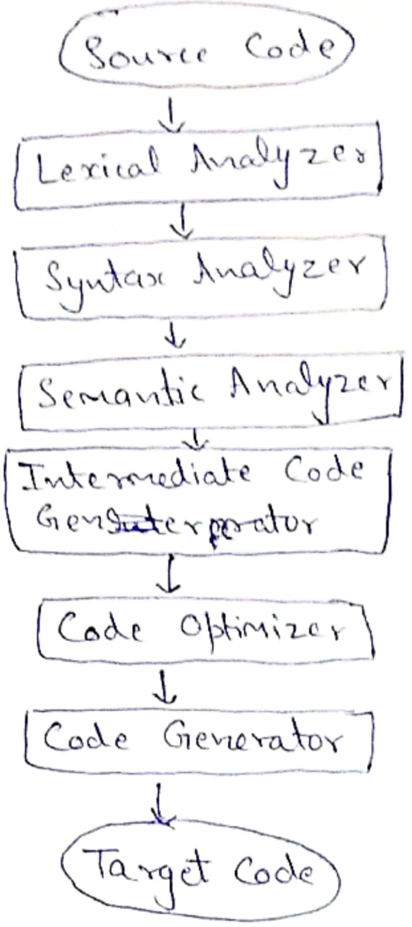
⇒ Front End → It maps legal code into Intermediate representation (IR).

⇒ Back End → It maps IR onto target machine, It also allows multiple front-ends

$$\begin{tikzcd}\text{Source Code} \arrow[r, Rightarrow] & \boxed{\text{Front End}} \arrow[r, Rightarrow, "\text{IR}"] & \boxed{\text{Back End}} \arrow[r, Rightarrow] & \text{Target Code.}\end{tikzcd}$$

Ques - 4 Explain different phases of Compiler with diagram.

Ans - 4 Lexical Analysis



The first phase of scanner works as text Scanner. This phase scans the source code as a stream of characters & converts it into meaningful lexemes. It represents these lexemes in the form of token as :-

< token-name, attribute-value >

Syntax Analysis

The next phase is called the syntax analysis or parsing. It takes the token produced by lexical analysis as Input and generates a parse tree. The parser checks if the expression made is ~~syntacally~~ Syntactically correct.

Semantic Analysis

It checks whether the parse tree constructed follows the rules of language. For example, assignment of values is b/w compatible data types or adding string in Integer.

Intermediate Code Generator

It represents a program for some abstract machine. It is in between the high level language and the machine level language. This Intermediate code should be generated in such a way that is easier to translate into target machine code.

Code Optimization

optimization can be assumed as something that removes unnecessary code lines, and arranges the sequence of statements in order to speed up program execution without wasting resources.

Code Generator

It takes the optimized representation of the intermediate code & maps it to the target machine language.

Ques-5 What is Cross Compiler? Explain the concept of boot strapping.

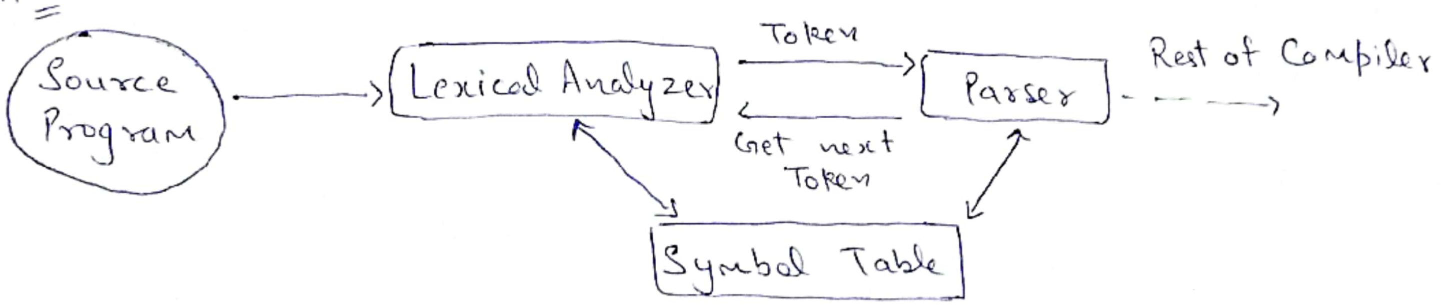
Ans-5 A Cross Compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is running. e.g.: a compiler that runs on windows 7 but generates code that run on Android smartphones.

=> Boot Strapping is a process in which simple language is used to translate more complicated programs which in turn may handle for more complicated program.

Example :- Suppose, we want to write a cross-compiler for new language X. The Implementation language of this compiler is say Y and the target code is in language Z. That is, we create XYZ. Now, if we run XYZ on YMM then we get compiler XMZ. That means, a ~~f~~ compiler for source language X that generates at a target code in language Z & which runs on M machine.

Ques-6 Explain Lexical Analyzer Generator (LEX) and Its Structure.

Ans-6



Lexical Analyzer Scans the entire source code of the program. It identifies each token one by one. Scanners are usually Implemented to produce tokens only when requested by & parser.

1. "Get next token" is & command which is sent from the parser to the lexical analyzer.

2. On receiving this command, the lexical analyzer scans the input until ~~#~~ it finds the next token

3. It returns token to parser.

Lexical Analyzer skips whitespaces and comments while creating this tokens. If any error is present then lexical analyzer will corellate that error with the source file and line number.

Ques. 7 Find the tokens for the given code

(a) for i = 1 to 100 do

for	-	keyword
i	-	identifier
=	-	assigngment operator
1	-	number
to	-	keyword
100	-	number
do	-	keyword

(b) if (i = 20) then goto 100

if	-	keyword
( )	-	operator
i	-	identifier
=	-	assignment operator
goto	-	keyword
100	-	number

Ques -8 What do you mean by derivation? what are its types? what are canonical derivations?

Ans. 8 A Derivation is basically a sequence of production rules, In order to get the Input string. During parsing we take two decisions for some sentential form of Input :-

(a) deciding the non-terminal which is to be replaced

(b) deciding the production rule, by which non-terminal would be replaced.

It is of 2 types :-

(1) Left Most derivation → If the sentential form of an Input is scanned and replaced from left to right, It is called left most derivation & the form derived is called left sentential form.

(2) Right most derivation → If we scan & replace the Input with production rules from right to left, It is right most derivation & the sentential form generated is right sentential form.

Ques. 9 What is an ambiguous grammar? Specify the demerits of It. Explain with the help of an example that how ambiguity can be removed.

Ans. 9 A Grammar G is said to be ambiguous if it has more than one parse tree (left / right derivation) for at least one string. Since Ambuiguous grammar has a capability to produce two parse trees for same expression, ~~If~~ Its often confusing for a compiler to figure out which one among all available parse tree is the correct one according to context.

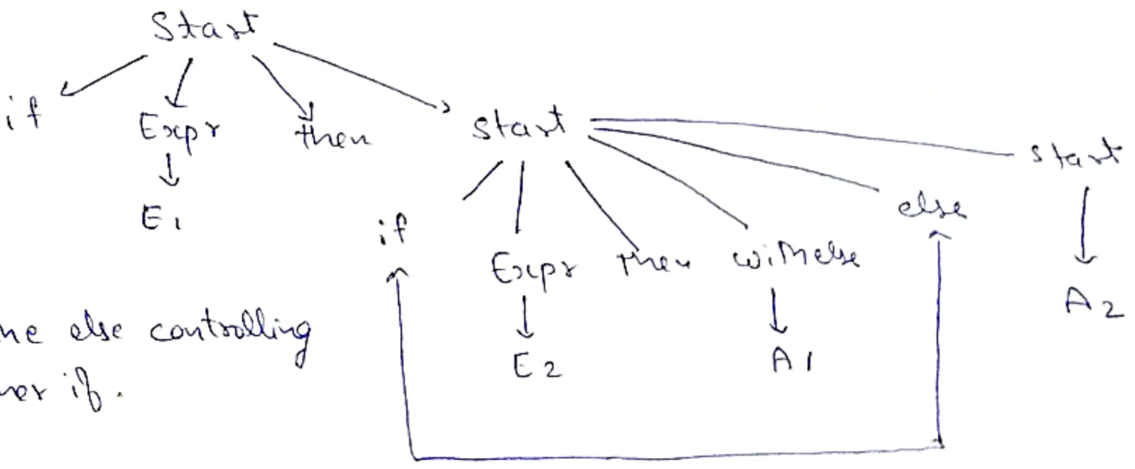
Removing Ambiguity :-

(i) Must rewrite grammar to avoid generating the problem

(ii) Match each else to Innermost unmatched if.

Let us derive the following using the rewritten grammar :-

if E₁ then if E₂ than A₁ else A₂



This binds the else controlling A₂ to inner if.

Ques-10 Explain "left factoring" with the help of an example.

Ans-10 Left factoring is removing the common left factor that appears in two productions of the same non-terminal. It is done to avoid back-tracing by the parser. Suppose the parser has a look-ahead, Consider this example:-

$A \rightarrow qB \mid qC$, where A, B, C are non-terminals & q is a sentence

In this case, the parser will be confused as to which of two productions to choose & It might have a back trace. After left factoring, the grammar is converted to

$A \rightarrow qD$

$D \rightarrow B \mid C$

In this case, a parser with a look-ahead will always choose the right production.