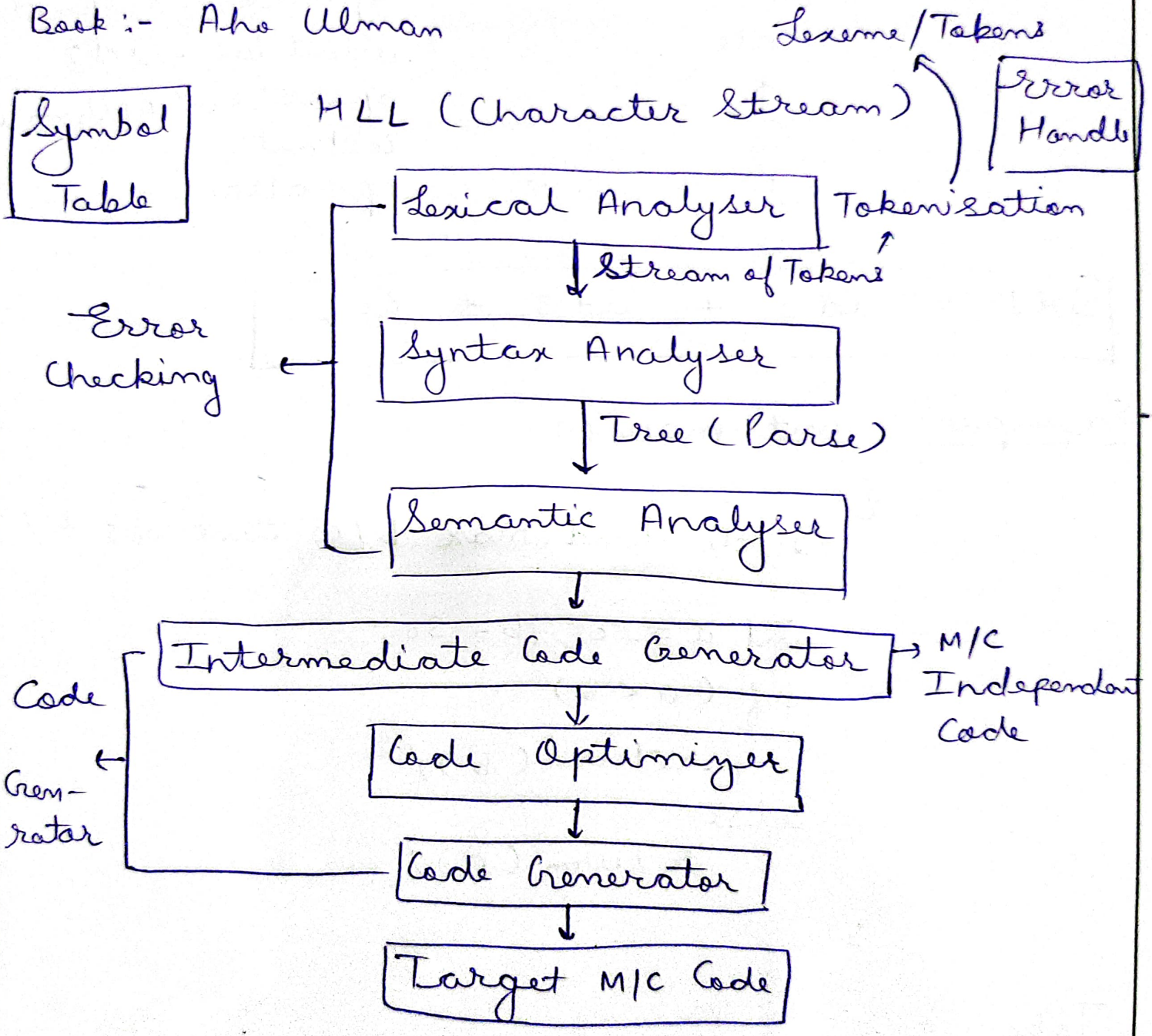
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

22/8/24 CD

Compiler Design

Book :- Aho Ullman



Main Ideas, Questions & Summary:

Library / Website Ref.:-

position = intial + rate * 60;

Lexeme	Tokens
position	identifier – id 1
=	operator (Assignment
initial	identifier – id 2
+	operator (Arthmatic
rate	identifier – id 3
*	operator (Arthmatic
60	costant
;	operater

id1 = id 2 + id 3 * 60;

Example:

```
int main()
{
  /* to find max b/w two nos */
   int a = 20, b = 30;
   if (a < b)
      return (b);
   else
      return (a);
}
```

Tokens = 32

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

Tokens :- a sequence of characters that can be treated as a in the grammar of programming language.

Lexeme :- a sequence of characters in source program that matches the pattern for token and the identifiers lexical analyser as an instance of that token.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

23/8/24

Types of Compiler :-

1. Single - Pass Compiler
2. Multipass Compiler
3. Just - In Time Compiler
4. Ahead - In Compiler

(i) **Single - Pass Compiler :-** process source code in single pass from start to finish.

(ii) **Multiple - Pass Compiler :-** it takes multiple passes over the source code, analyzing it at different stages.

(iii) **Just - In Time Compiler :-** it translates code in ML while program is running

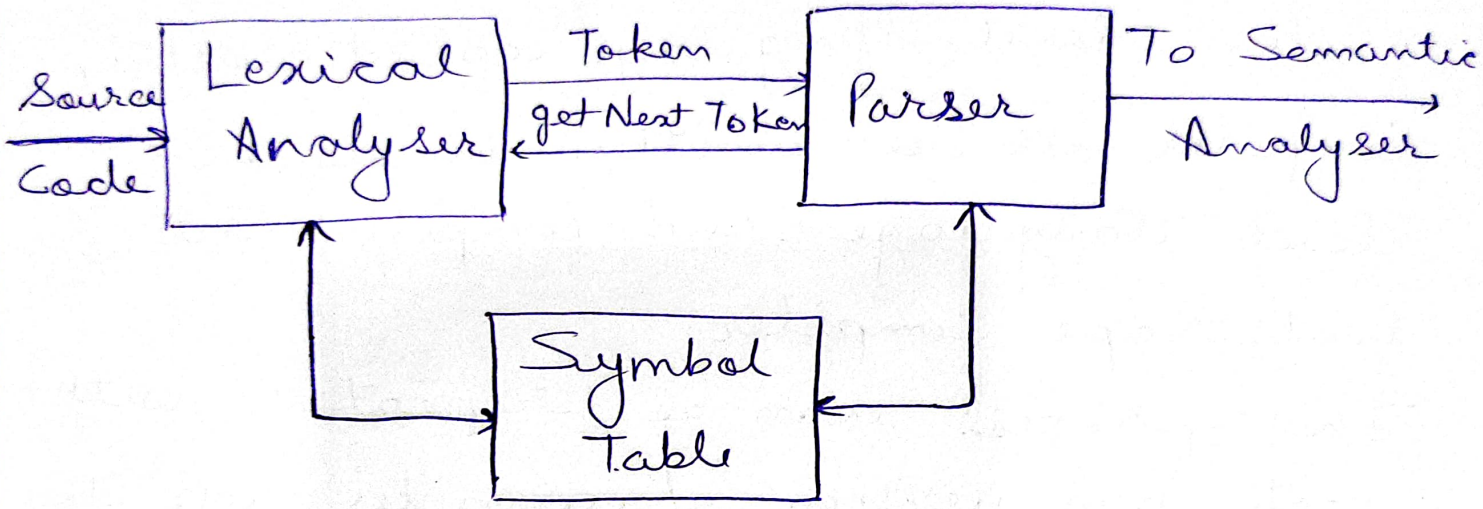
ex :- used in Javascript to improve performance by converting the code as needed.

(iv) **Ahead Time :-** translates code in ML before program is done.

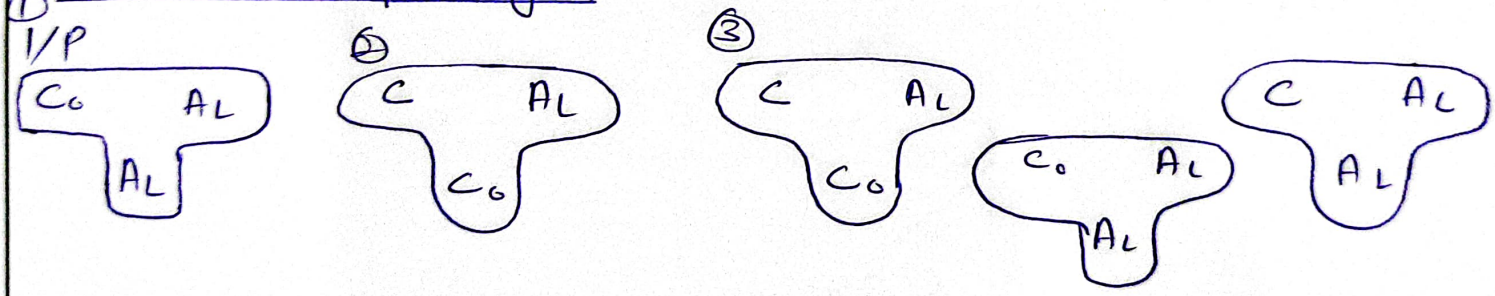
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

Lexical Analyser



Bootstrapping :-



1. Initially compilers were written in assembly language or low level language specific to hardware
2. As programming language evolved the need for writting compilers in a higher

Main Ideas, Questions & Summary:

Library / Website Ref.:-

level language became evident.

3. Bootstrapping was introduced as solution to this.

Concept of Bootstrapping :-

1. A simple compiler is first written in a low level language called Bootstrap.
2. Compiler is re-written in its own level language and compiled using bootstrape compiler.
3. This process can be repeated with each new version of compiler use to compile its new version.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

27/8/24

position = initial + rate * 60

↓

Lexical Analyser

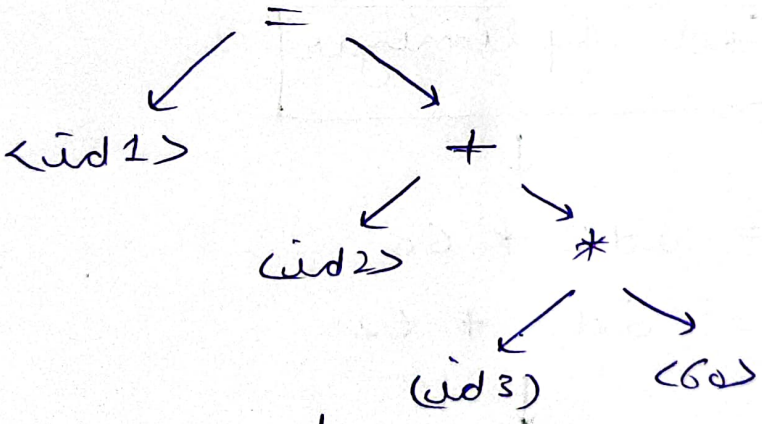
↓

<id 1> <=> <id 2> <+> <id 3> <*> <60>

↓

Syntax Analyser

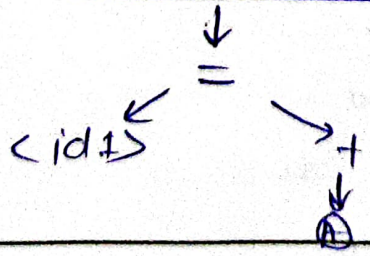
↓



↓

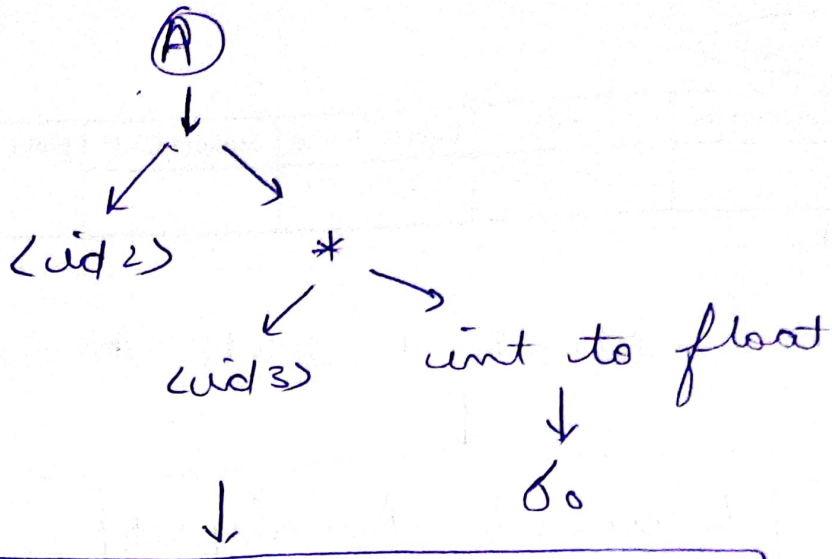
Semantic Analyser

↓



Main Ideas, Questions & Summary:

Library / Website Ref.:-



Intermediate Code Generator

$t1 = int\ to\ float\ (60)$

$t2 = id\ 3 * t1$

$t3 = id\ 2 + t2$

$t4 = t3$

Code Optimizer

$t1 = id\ 3 * 60$

$id2 = id\ 2 + t2$

Code Generator

```
LDF   R2, id 3
MULF  R2, R2, #60.0
LDF   R1, id 2
ADDF  R1, R1, R2
STF   id 2, R1
```

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

Symbol Table :-

Position	
initial	
rate	

★ YACC- Yet Another Compiler Compiler

Main Ideas, Questions & Summary:

Library / Website Ref.:-