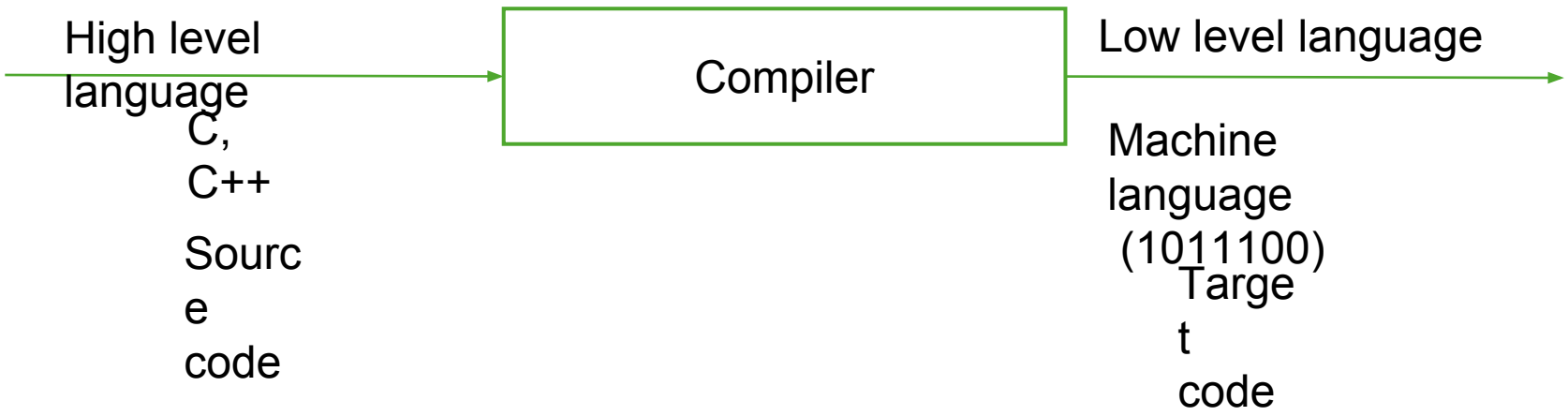
Lecture 1,2

Rakib Mahmud

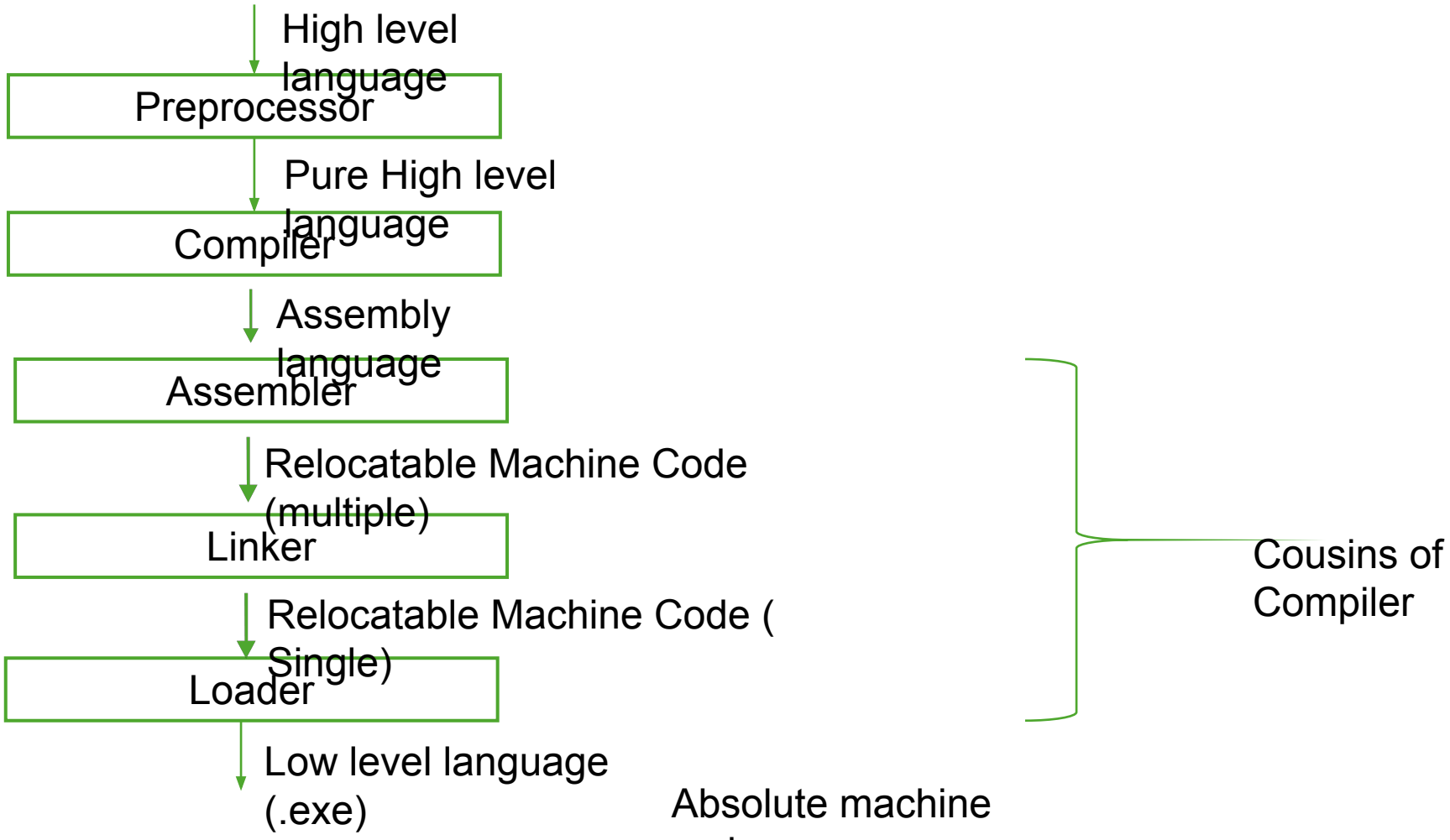
Introduction

- Compiler is a program or software.

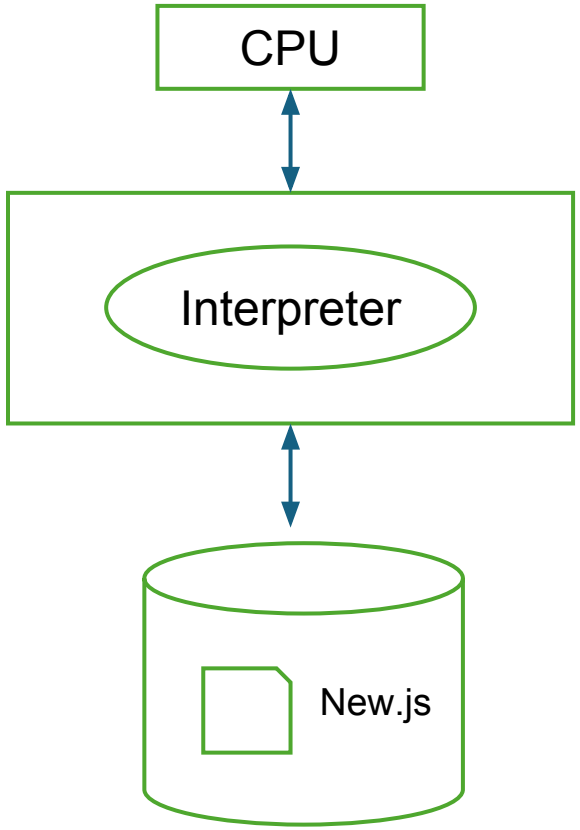


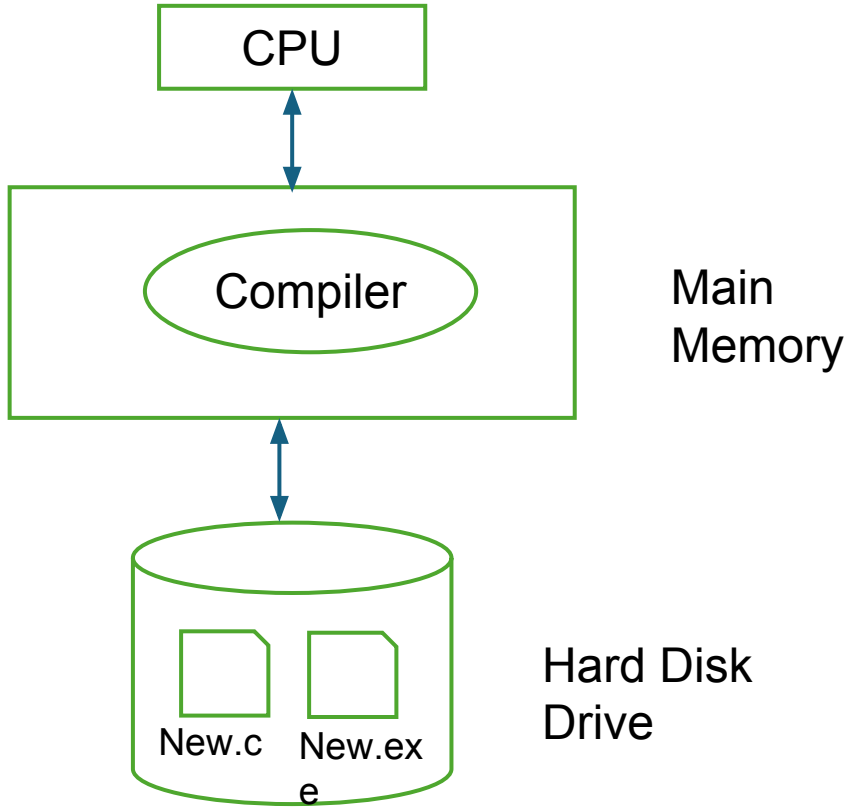
- It is a translator.
- In here machine means processor.

LPS (Language Processing System)



Compiler vs Interpreter

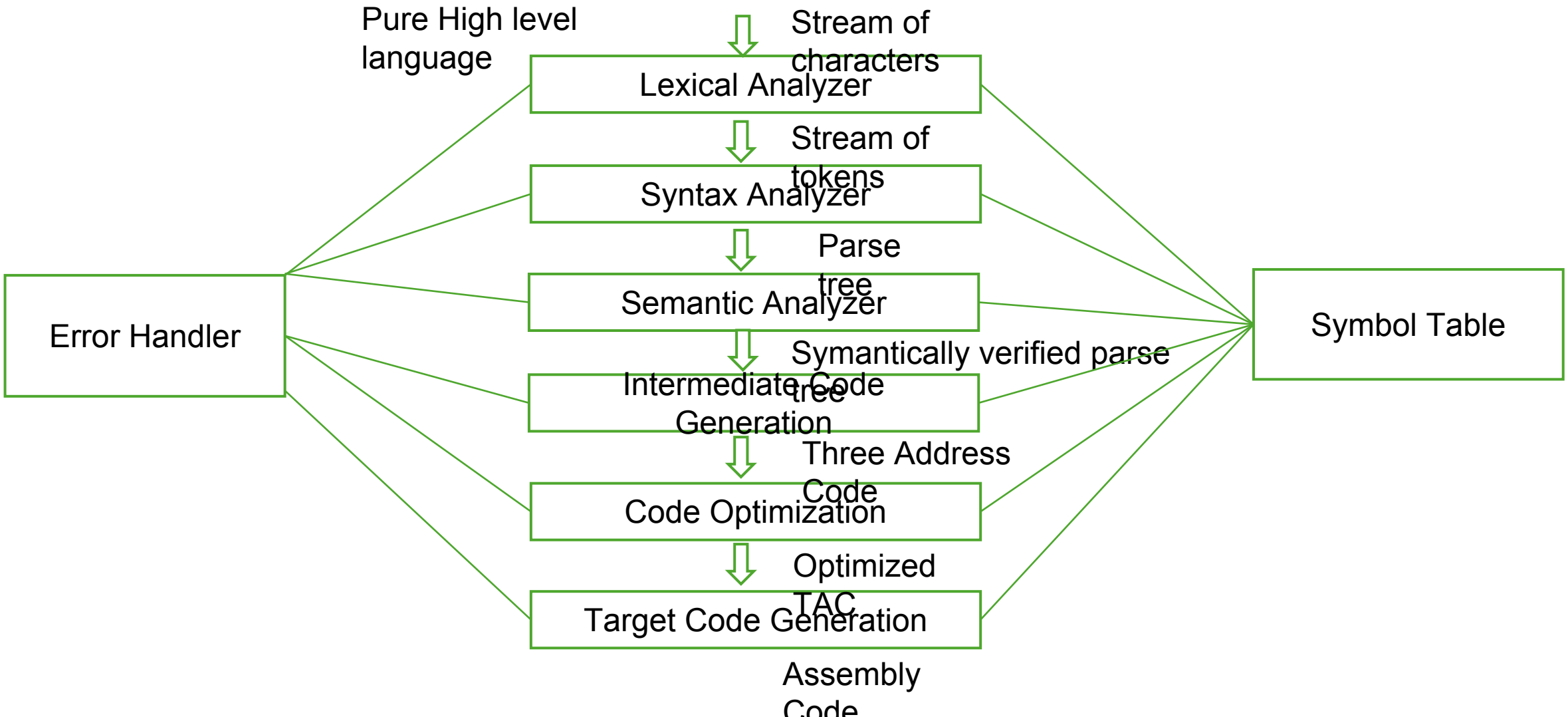




Compiler vs Interpreter

Compiler	Interpreter
Compiler reads the full program.	Interpreter reads line by line.
It generates .exe file.	It doen’t generate .exe file.
Compiled codes run faster than Interpreter. So, it required less time.	Interpreted codes run slower than Compiler. So, it required more time
It does not require source code for later execution.	It requires source code for later execution.
It takes more space.	It takes less space.
Debugging is difficult here.	Debugging is easy here.

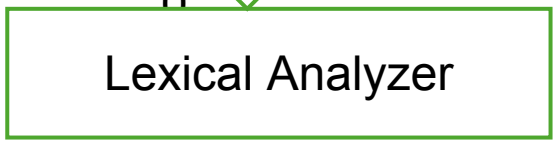
6 Phases of Compiler



Lexical Analyzer

Pure High level language

float
x,y,z
x=y+z*6
0



X : identifier <id,1>
= : Assignment operator
Y : identifier <id,2>
+ : Addition operator
Z : identifier <id,3>
* : Multiplication Operator
60: integer consonant

This process Is called tokenization

Table 1: Symbol Table

SL No	Variable Name	Type
1	x	float
2	y	float
3	z	float

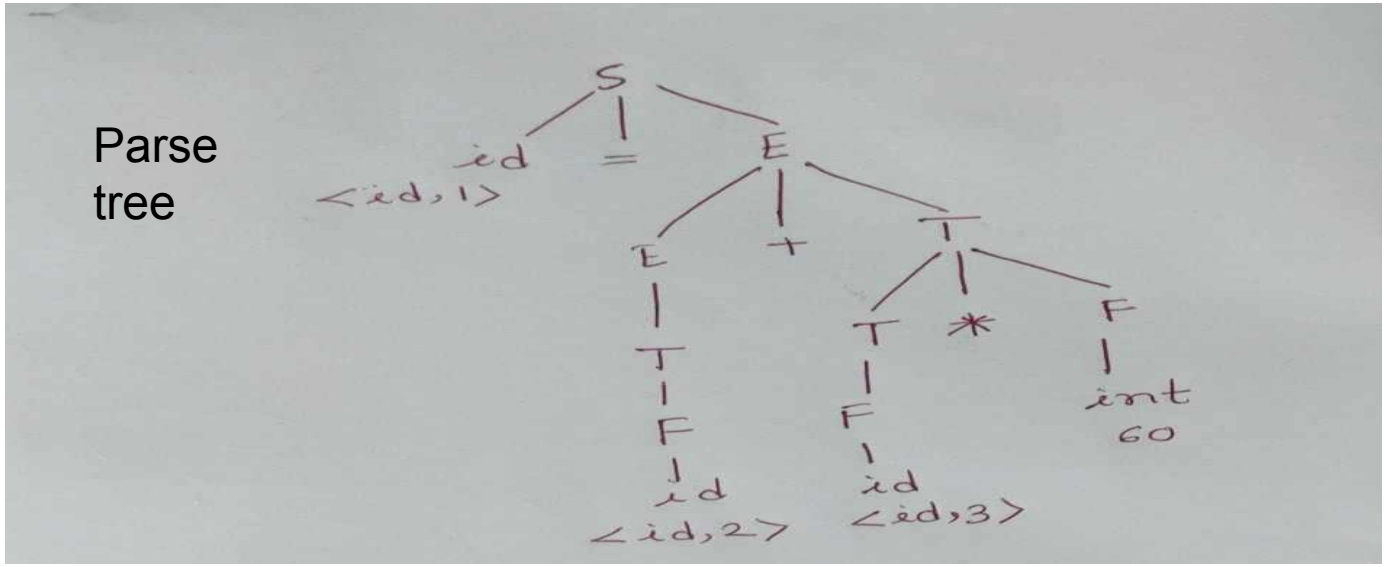
<id,1> = <id,2> + <id,3> * 60

Syntax Analyzer

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

Syntax Analyzer

Gramme
r
S -> id =
E
E ->
E+T|T
T ->
T*F|F
F ->id|int



Semantic Analyzer

Parse tree

Semantic Analyzer

1. Type Checking
2. Undeclared variable
3. Multiple Decleration

Symantically Verified Parse Tree

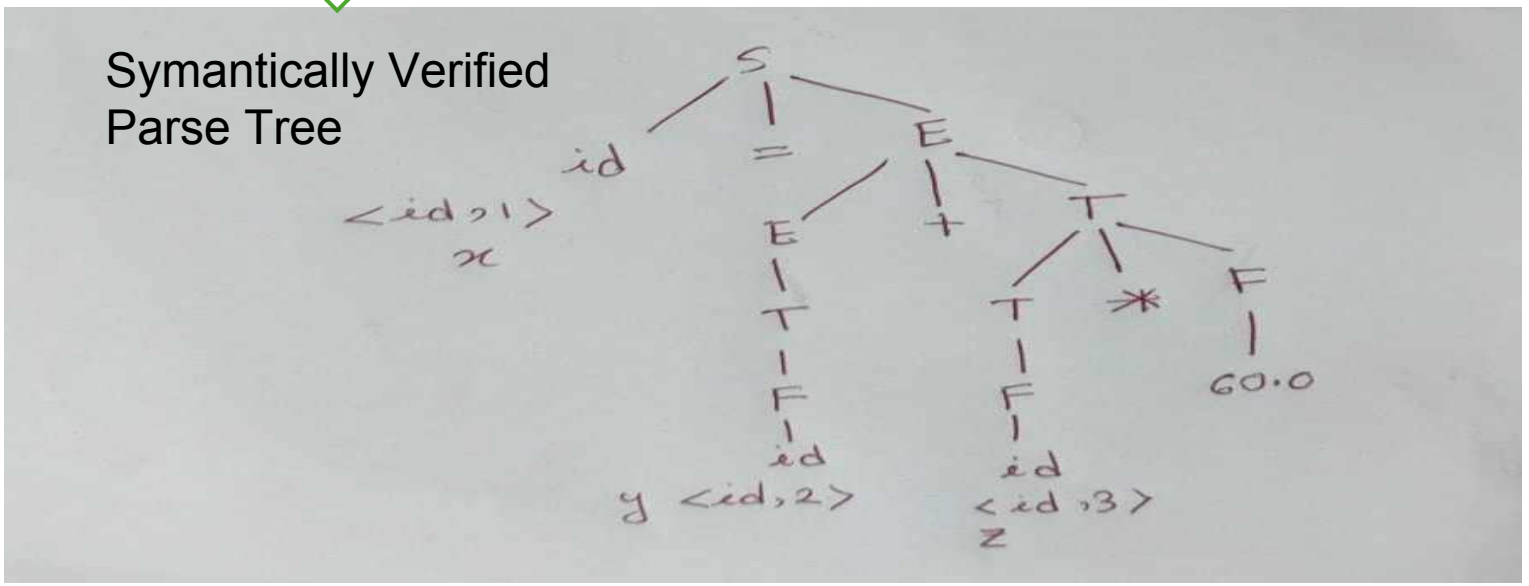


Table 1: Symbol Table

SL No	Variable Name	Type
1	x	float
2	y	float
3	z	float

Intermediate code Generation

- In here the most valuable is three address code.

```
t1 =
z*60.0
t2 = y+t1
X  = t2
```

Code Optimization

- Reduced the no of lines in the code.
- It is optional

```
t1 =
z*60.0
X  = y+t1
```

Target Code Generation

- It generates assembly language

R_0 <-
z
R_1 <-
y

```
MUL     R0,
60.0
ADD     R1, R0
STORE   X, R1
```