Lecture 7

Lexical Analysis

Stream of character (Source code) → Lexical Analysis → Stream of Token

1st half	2nd half
Scan	Clean code এবং token generate

cmt, extra white space বাদ দিতে হবে।

Tokens:

```
int x,y,z;
float b=1.0;
x = y = z = 10;
float c = 2*b*(x+y+z);
```

এই Generalize catagory গুলো ⊛ Token name

- identifier: x, y, z, b, c
- Operator: = , * , +
- Constant number: 1.0 , 10
- Keyword: int, float
- String literal: "...."

→ Lexeme program এর মৌলিক উপাদান।

Token: < Token_name , Lexeme >

For example: Token → < id , x >

id ↓ identifier

কোনটা কোন token-name সেগুলো pattern analysis করে বের করতে হবে।

প্রত্যেকটা Lexeme প্রত্যেকটা Token name এর সাথে match করে দেখতে হবে কাদের pattern এর সাথে match করে।

→ঐ pattern match করানো হয় DFA দিয়ে।

int num = 10

table index-এ রেখেও করতে পারি

1	num	int	10

← symbol table.

lexeme+delimeter
↓

delimeter → seperator · (space, coma, ব্রাকেট, operator)

Lecture-1

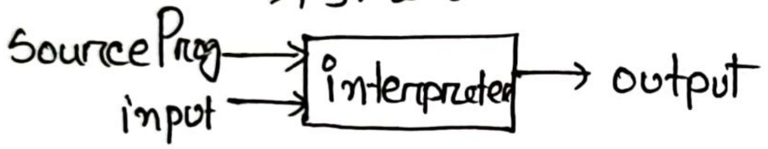
natural Language → আগে language then grammar define করি

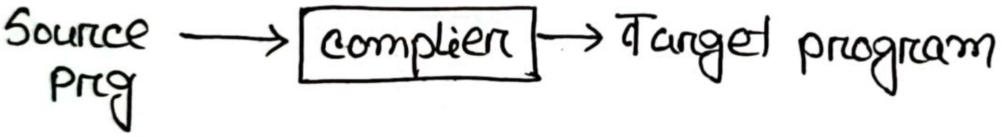
formal Language → আগে grammar define করি then Language

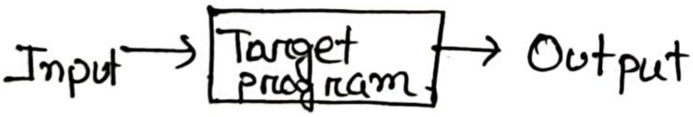
Compiler → source program কে target program-এ convert
↳ c, java
↳ binary তে convert করে
→ high level Language → Low level L. convert

programme → Build করা যাবেই compiler.

Interpreter → Compiler + Excebition line by line হয়
↳ python (input







* Interpreter slower than Compiler
কারণ Line by Line compile করে execute করে তাই time বেশি লাগে।

* Compiler → Error detect করা Hard
* Interpreter → " " " easy.

* Assembly language -

→ Linker/ Loader → memory তে ছোট ছোট ভাগজেরে whole code রাখা হয় যেখানে প্রতি ভাগের Last-এ পরের ভাগের কিছু info লিখা থাকে যাকে Linker বলে এবং পরে সব marge করা হয় যা loader করে।

Analysis phase ← Front End: Lexical ⟶ Intermediate

Back End: Machine ⟶ Machine-Dependent
↳ Synthesis phase.

Token: punctuation marks, keyword ...

Symbol Table: Token — attribute (Type, কত bit ...)
Analysis phase-এ symbol table এ entry জমা হয়।

Lecture-2

$\Sigma \rightarrow$ Alphabet (finite set of symbols)

$\varepsilon \rightarrow$ epsilon → Empty.
↳ valib for Alphabet

$|w| \longrightarrow$ length

Prefix → (ε , a , ab & abc) [proper prefix main wo নিবে না। so (ε, a, ab)
ε → কিছুই নিবে না

Power of Alphabet →

$\Sigma = \{0, 1\}$

Σ^k = k-lenght এর যতগুলো string বানানো possib

$\Sigma^+ = \Sigma^* - \Sigma^0$

Formal language → A set of string . আর এই string গুলো আসবে Σ^* থেকে।

↳ all possible lenght র যত string হওয়া possib সব আছে।

So Formal L. is subset of Σ^*

$\Sigma_2 = \{a, b, c, d\}$

$\Sigma_1 = \{a, b, c\}$

Here Σ_2 is superset of Σ_1

so if $L \underset{\text{Subset}}{\subseteq} \Sigma_1$ so $L \subseteq \Sigma_2$

$\epsilon \neq \phi$

ϵ ↓ String যার lenght zero

ϕ ↳ যেখানে কোন word বা কিছুই নেই।

$L \subseteq \{0, 1\}^*$