UEM
UNIVERSITY OF ENGINEERING & MANAGEMENT
श्रद्धावान लभते ज्ञानम्
Good Education, Good Jobs

Compiler Design

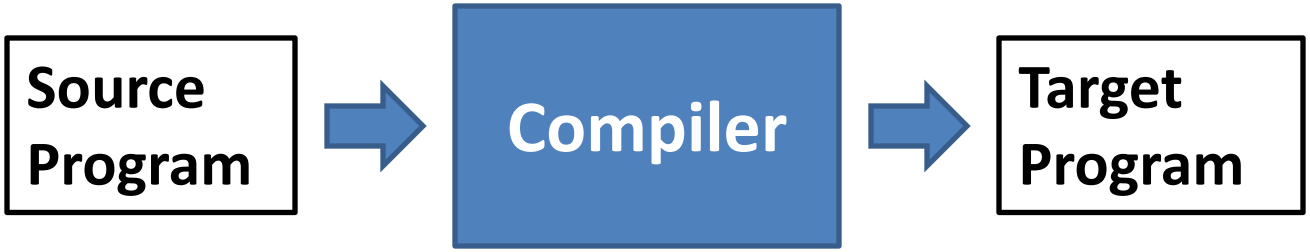
Prof. Sankhadeep Chatterjee
Department of Computer Science & Engineering, UEMK

Prerequisite

- Data Structure & Algorithms
- Formal language & Automata Theory

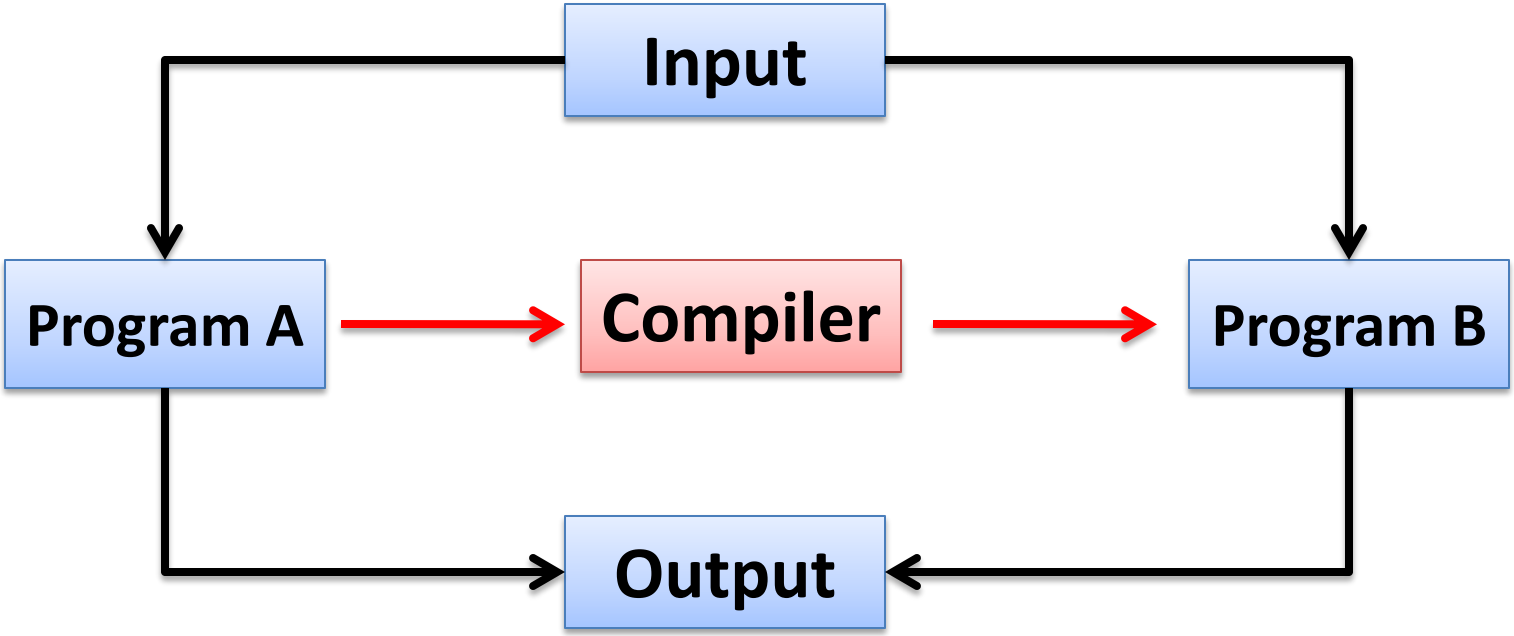
What is a Compiler?

- A compiler is a program that can read a program in one language (the *source language*) *and* ***translate*** it into an equivalent program in another language (the *target language*)

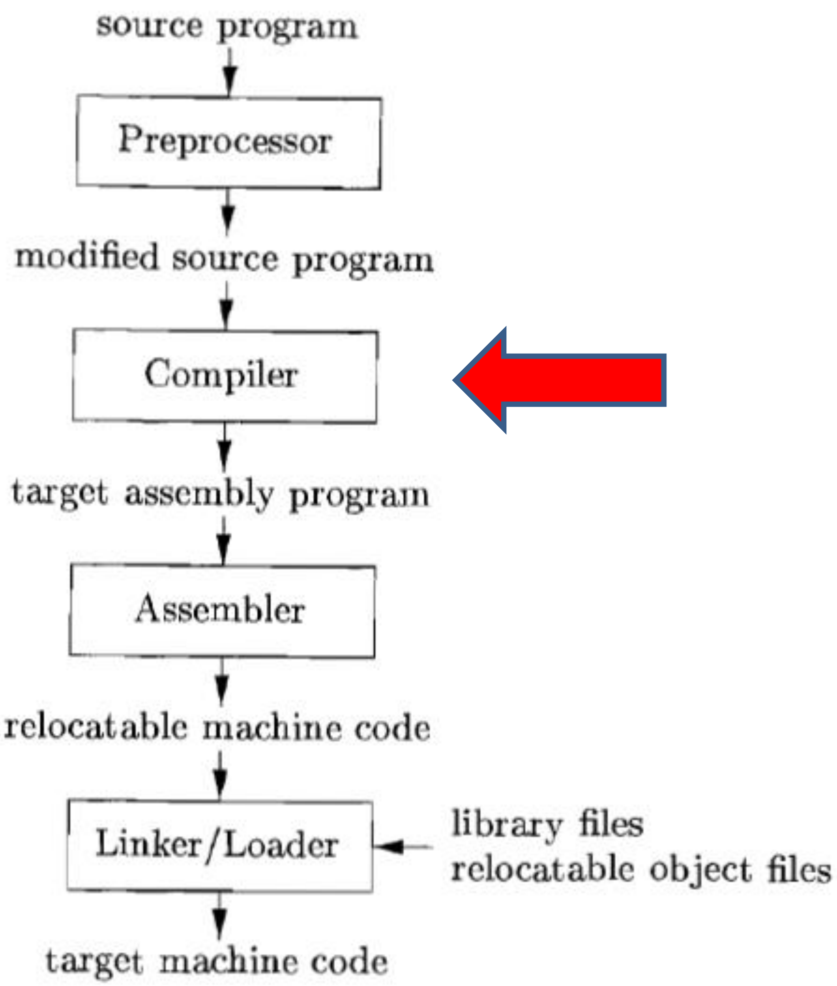


What is a Compiler?

- By **equivalent program** we mean that for **same set of inputs** the programs should produce **same output**.



Language Processing System



Compilation Phases

- The compilation (translating code) is not done at once.
- Instead, several small phases are involved.
- Each of these phases gradually converts the code to target language.

Initial Steps

- The first few steps can be understood by analogies to how humans comprehend a natural language
- The first step is recognizing/knowing alphabets of a language. For example
  - English text consists of lower and uppercase alphabets, digits, punctuations and white spaces
  - Written programs consist of characters from the ASCII characters set (normally 9-13, 32-126)

Initial Steps

- The next step to understand the sentence is recognizing words
  - How to recognize English words?
  - Words found in standard dictionaries

Initial Steps

- How to recognize words in a programming language?
  - a dictionary (of keywords etc.)
  - rules for constructing words (identifiers,numbers etc.)
- This is called lexical analysis

Initial Steps

- Recognizing words is not completely trivial. For example:

ify ouc an re adth is, yo uar eagen ius.

Initial Steps

- Recognizing words is not completely trivial. For example:

  ify ouc an re adth is, yo uar eagen ius.

  ify ouc an re adth is, yo uar eagen ius.

Lexical Analysis: Challenges

- We must know what the word separators are

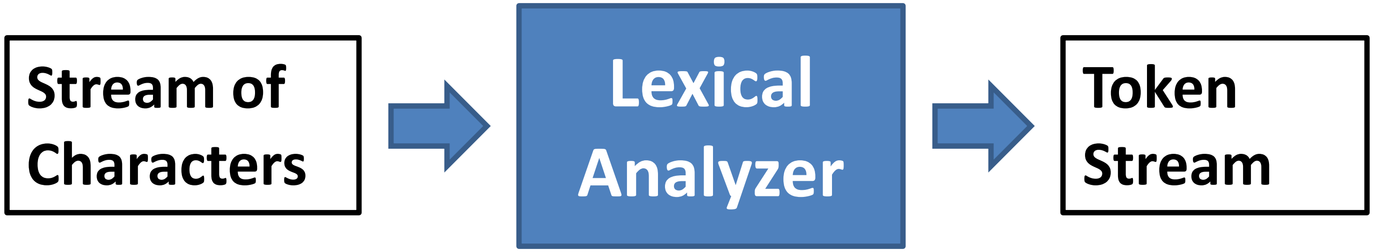
- The language must define rules for breaking a sentence into a sequence of words.

- Normally white spaces and punctuations are word separators in languages.

Lexical Analyzer

- *The lexical* analyzer reads the stream of characters making up the source program and groups the characters into meaningful sequences called ***lexemes***. *For each* lexeme, the lexical analyzer produces as output a *token of the form;*

<token_name, attribute_value>



Lexical Analysis contd.

Consider the following ‘C’ language statement;

```
a = b + c ;
```

```
< a > < = > < b > < + > < c > < ; >
```

```
a -> identifier
= -> Assignment operator
b -> identifier
+ -> operator
c -> identifier
; -> Terminating symbol
```

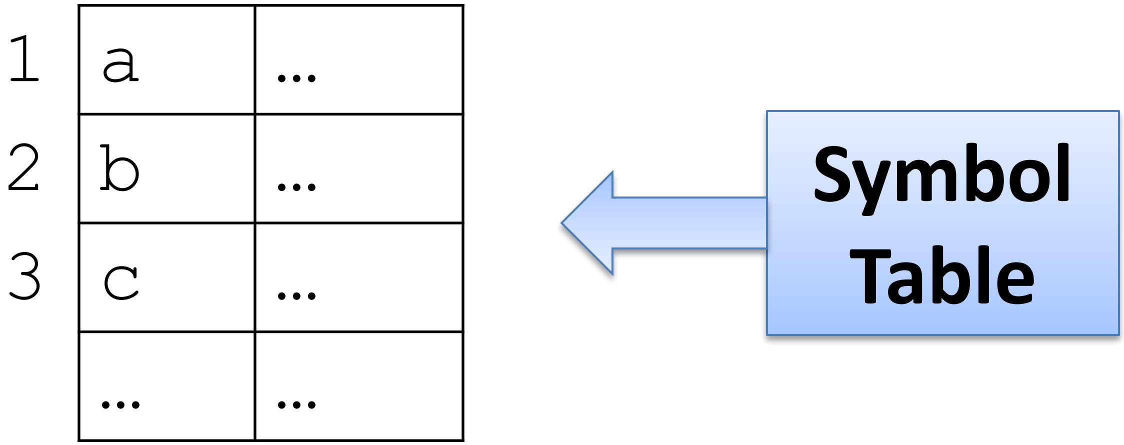
Token Stream Representation

- Input character stream : **`a = b + c ;`**
- Lexemes : **`< a > < = > < b > < + > < c > < ; >`**
- Each lexeme is mapped to a token.
- For example;
  - Lexeme **`<a>`** is mapped to **`<id,1>`**
  - Lexeme <=> is mapped to <=> and so on.

Token Stream contd.

- Input character stream : `a = b + c ;`
- Lexemes : `< a > < = > < b > < + > < c > < ; >`
- Token Stream :

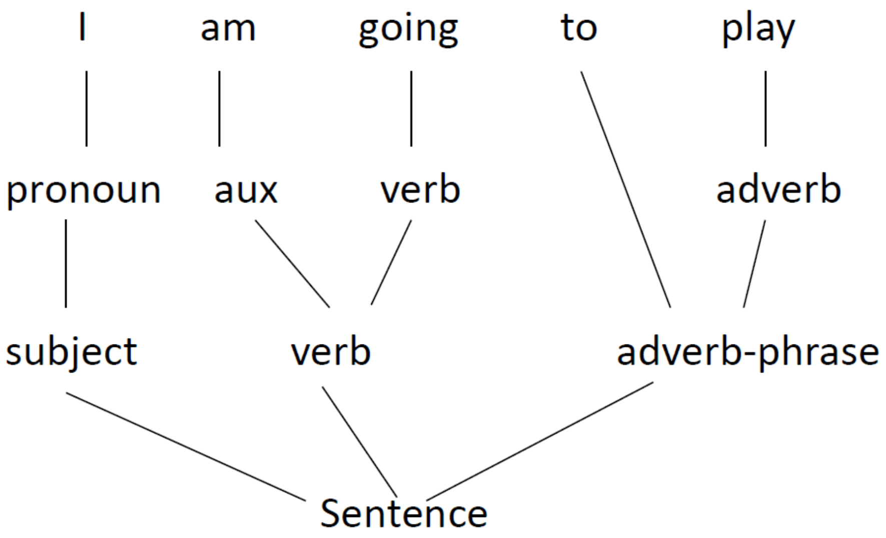
```
<id, 1> <=> <id, 2> <+> <id, 3> <;>
```



1	a	…
2	b	…
3	c	…
	…	…

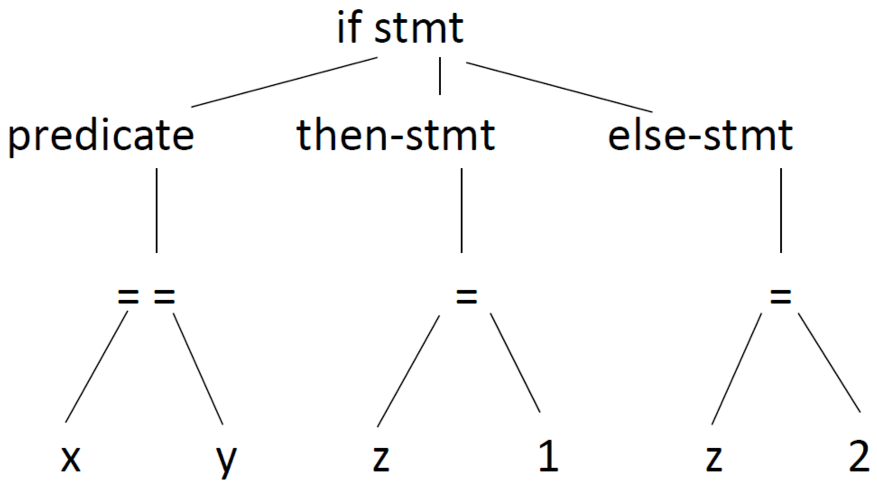
Next step

- Once the words are understood, the next step is to understand the structure of the sentence
- The process is known as *syntax checking or parsing*



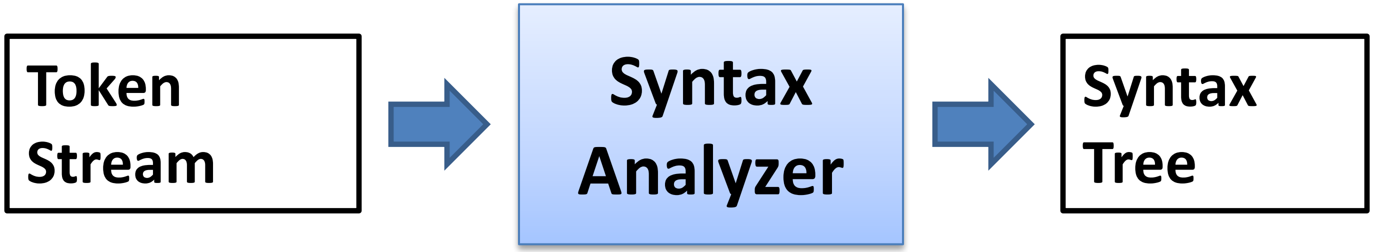
Syntax Analysis

- Parsing a program is exactly the same process as shown in previous slide.
- Consider an expression
  - if x == y then z = 1 else z = 2



Syntax Analyzer

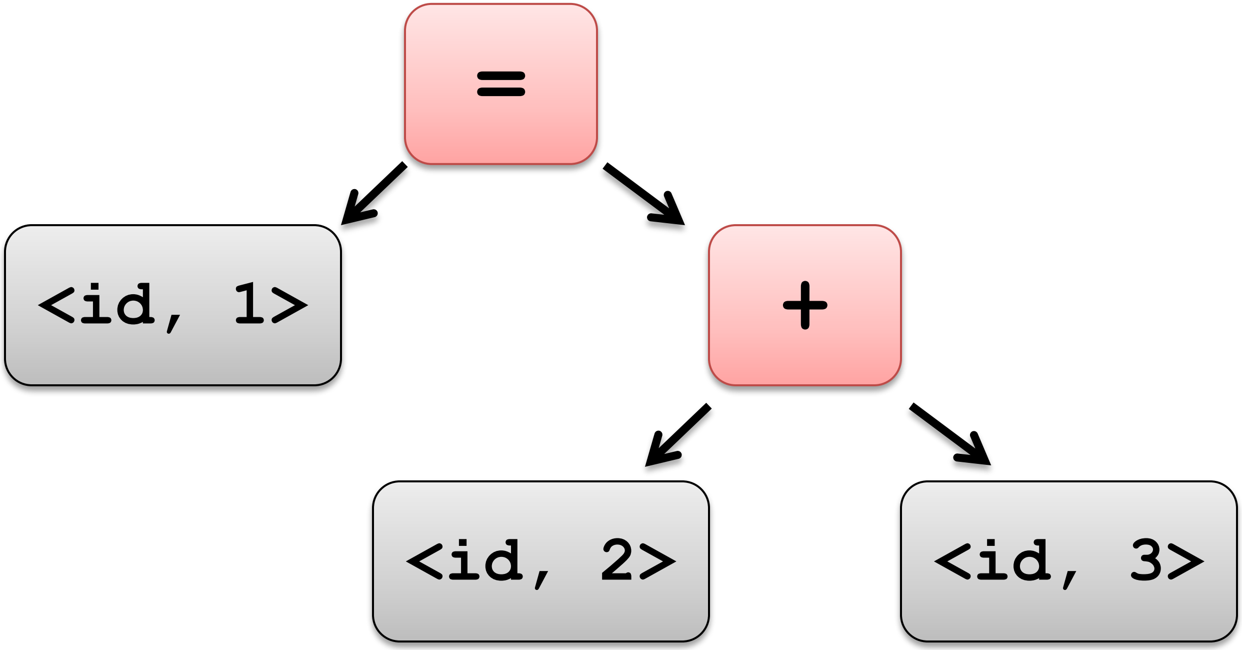
- The second phase of the compiler is *syntax analysis* or *parsing.*
- The parser uses the tokens produced by the lexical analyzer to create a tree-like intermediate representation that depicts the grammatical structure of the token stream.



Syntax Tree

- Input character stream : **a = b + c ;**
- Token Stream :

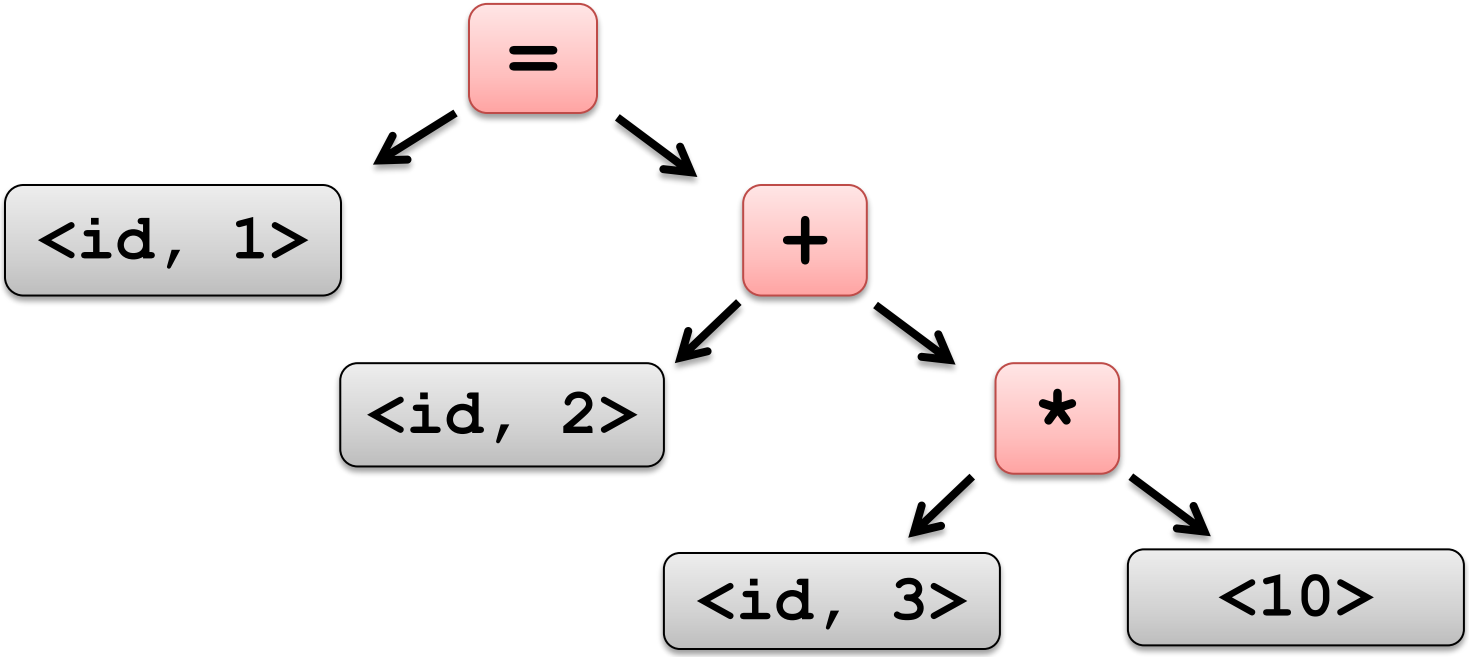
```
<id, 1> <=> <id, 2> <+> <id, 3> <;>
```



Syntax Tree contd.

- Input character stream : **`a = b + c * 10`**
- Token Stream :

`<id, 1> <=> <id, 2> <+> <id, 3> <*> <10>`



Semantic Analysis

- Once the sentence structure is understood we try to understand the meaning of the sentence (semantic analysis)
- A challenging task
  - Ravi said Ajay got his job offer from Google
  - Whom does ‘his’ referring to?
  - Who is the Lucky person!

Semantic Analysis

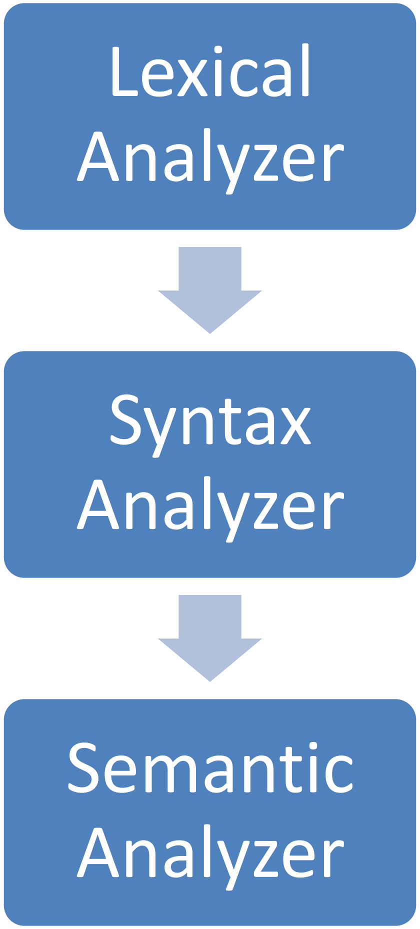
- Worse case:
  - Amit said Amit left his assignment at home
- Too hard for compilers. They do not have capabilities similar to human understanding
- However, compilers do perform analysis to understand the meaning and catch inconsistencies

Semantic Analysis

- Programming languages define strict rules to avoid such ambiguities
- {
  - int Amit = 3;
  - {
    - int Amit = 4;
    - cout << Amit;
  - }
- }

Front End Compilation

- Front End Compilation
- Machine Independent



Thank You