System Software (5KS03)

Unit 1 : Introduction to Compiling

Lecture : 2 Phases of a compiler,

A S Kapse,

Assistant Professor,

Department Of Computer Sci. & Engineering

Anuradha Engineering College, Chikhli

Contents…

- Introduction to Compilers
- **Phases of Compiler**

Objectives…

- Upon completion of this lecture, you will be able

✓ To understand the basics of compiler

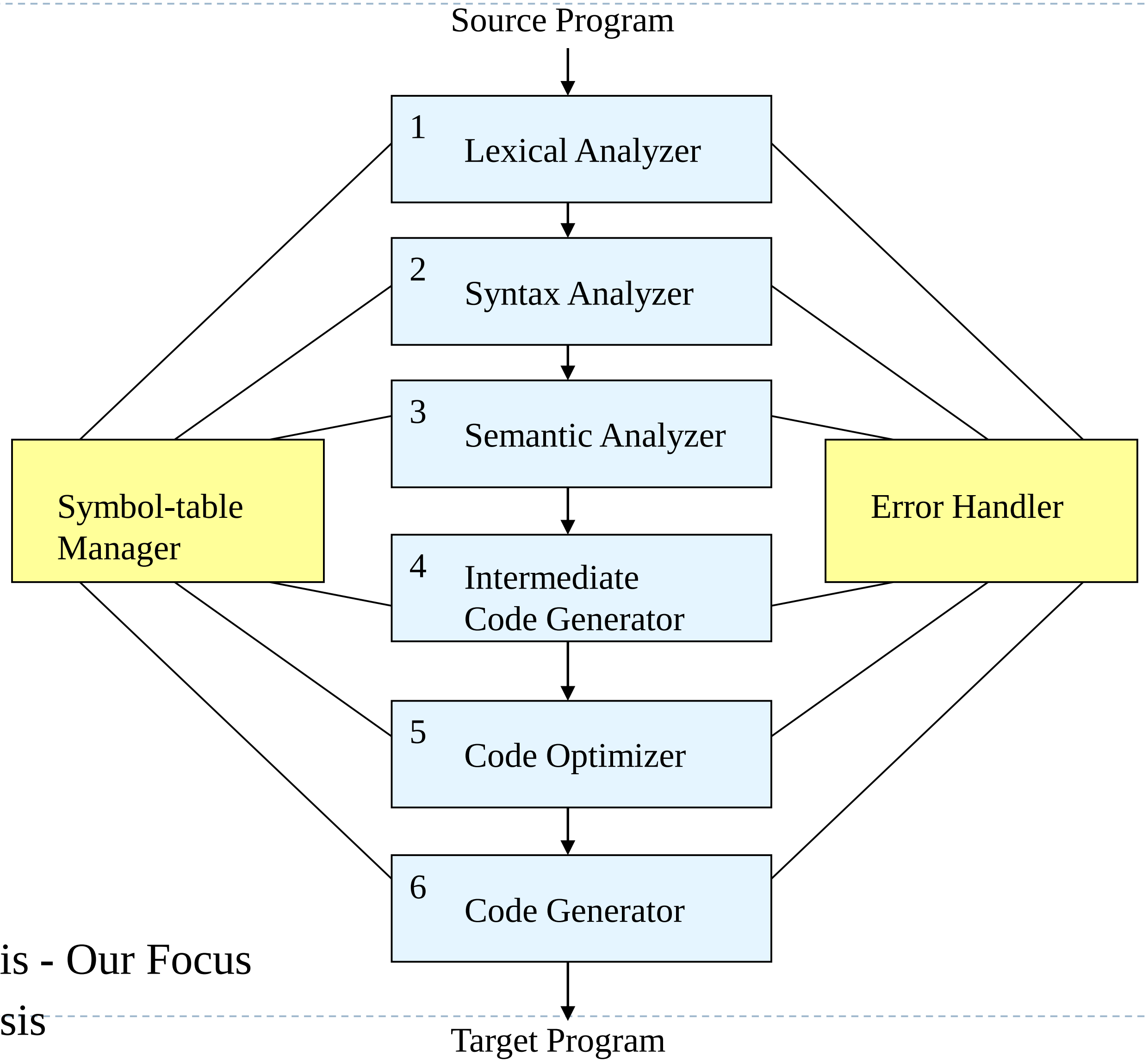
✓ To understand Application of compiler

✓ To understand phases of compiler

Review…./ Concepts

- What do you mean by compiler?
- What do you mean by Operating System?
- What do you mean by system?

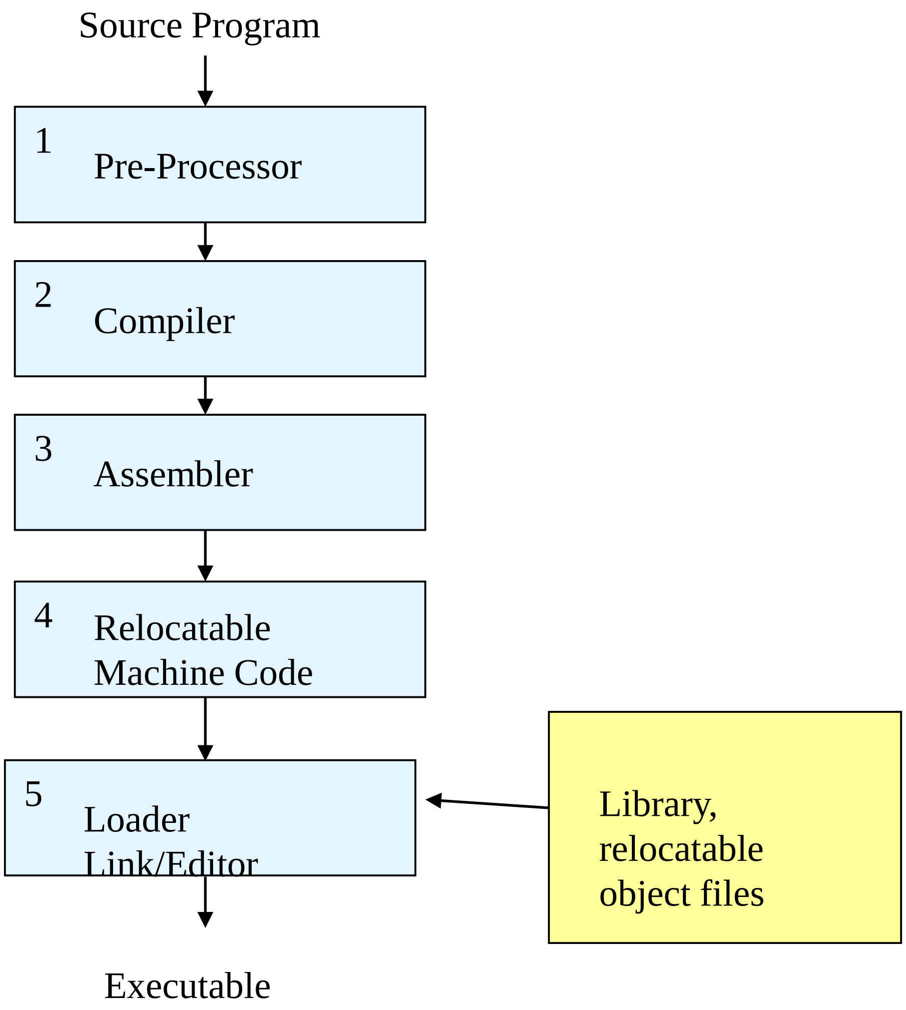
The Many Phases of a Compiler



1, 2, 3 : Analysis - Our Focus

4, 5, 6 : Synthesis

Language-Processing System



The Analysis Task For Compilation

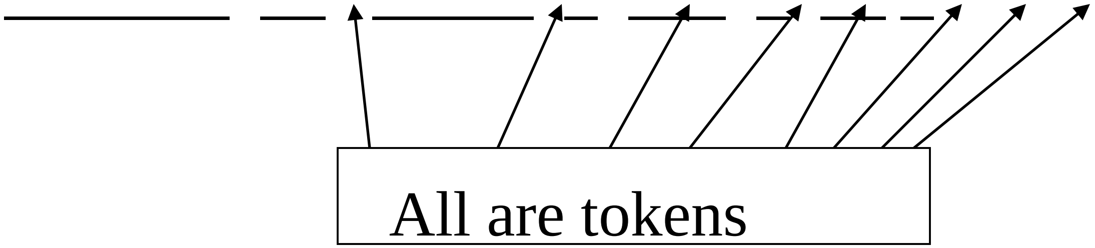
- Three Phases:
  - **Linear / Lexical Analysis**:
    - L-to-r Scan to Identify Tokens
      token: sequence of chars having a collective meaning
  - **Hierarchical Analysis**:
    - Grouping of Tokens Into Meaningful Collection
  - **Semantic Analysis**:
    - Checking to ensure Correctness of Components

Phase 1. Lexical Analysis

Easiest Analysis - Identify tokens which are the basic building blocks

For Example:

Position := initial + rate * 60 ;

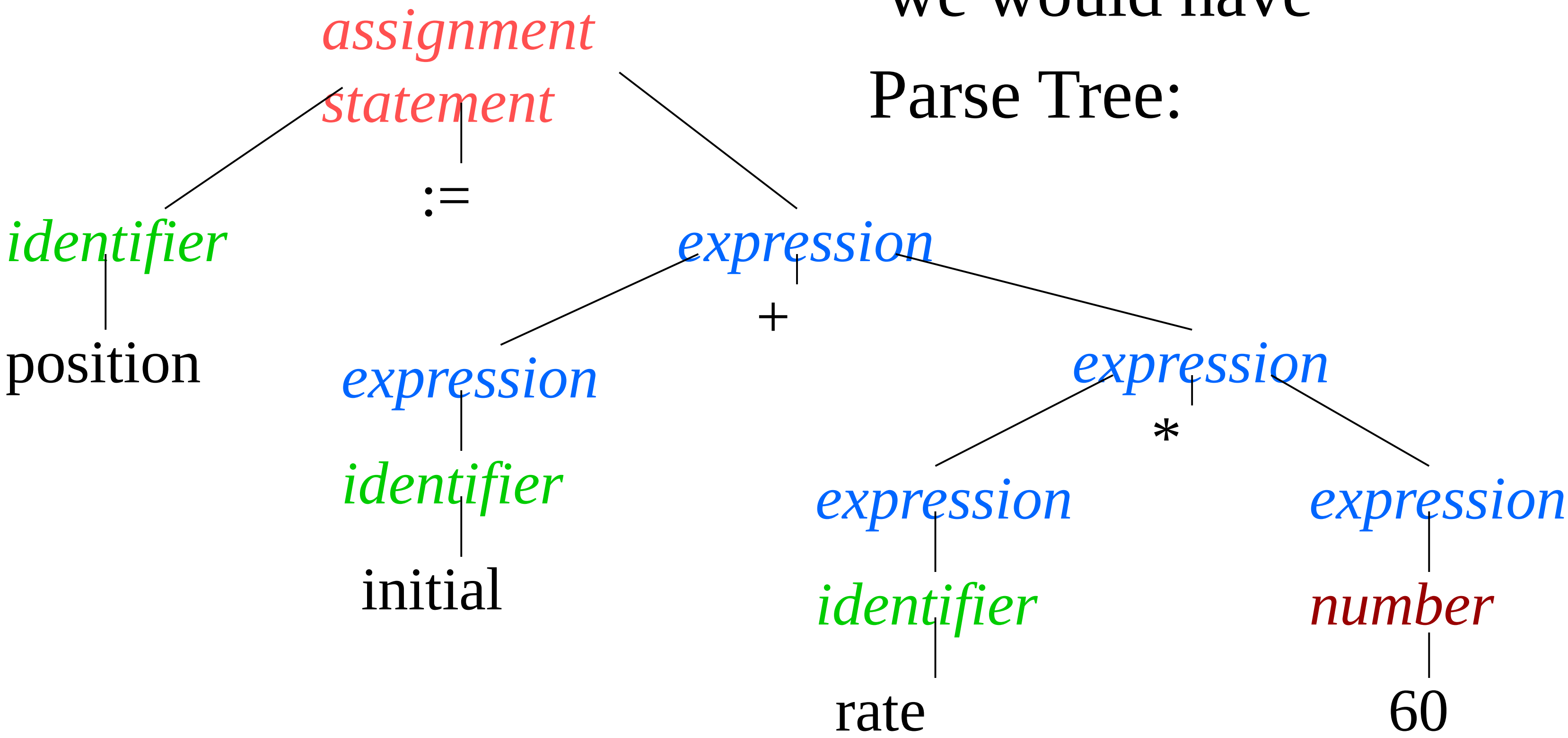


All are tokens

Blanks, Line breaks, etc. are scanned out

Phase 2. Hierarchical Analysis
aka Parsing or Syntax Analysis

For previous example,
we would have
Parse Tree:



assignment statement

identifier

:=

expression

position

expression

+

expression

identifier

*

initial

expression

expression

identifier

number

rate

60

Nodes of tree are constructed using a grammar for the language

What is a Grammar?

- Grammar is a Set of Rules Which Govern the Interdependencies & Structure Among the Tokens

statement is an assignment statement, or while statement, or if statement, or ...

assignment statement is an identifier := expression ;

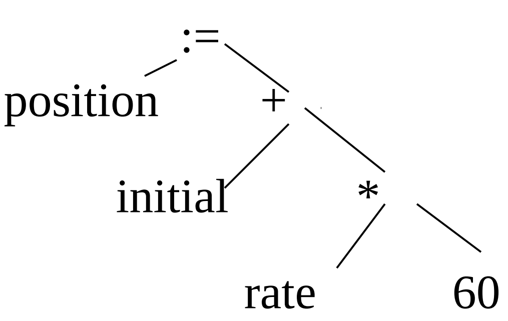
expression is an (expression), or expression + expression, or expression * expression, or number, or identifier, or ...

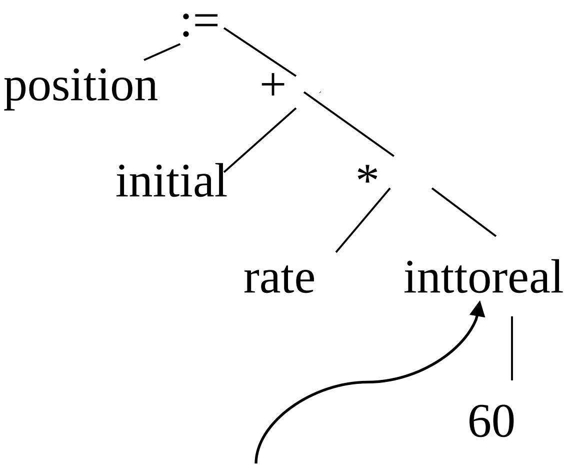
Why Have We Divided Analysis in This Manner?

- Lexical Analysis - Scans Input, Its Linear Actions Are Not Recursive
  - Identify Only Individual “words” that are the the Tokens of the Language
- Recursion Is Required to Identify Structure of an Expression, As Indicated in Parse Tree
  - Verify that the “words” are Correctly Assembled into “sentences”
- What is Third Phase?
  - Determine Whether the Sentences have One and Only One Unambiguous Interpretation
  - … and do something about it!
  - e.g. “John Took Picture of Mary Out on the Patio”

Phase 3. Semantic Analysis

- Find More Complicated Semantic Errors and Support Code Generation
- Parse Tree Is Augmented With Semantic Actions





Compressed Tree

Conversion Action

Phase 3. Semantic Analysis

- Most Important Activity in This Phase:
- Type Checking - Legality of Operands
- Many Different Situations:

```
Real := int + char ;

A[int] := A[real] + int ;

while char <> int  do

…. Etc.
```

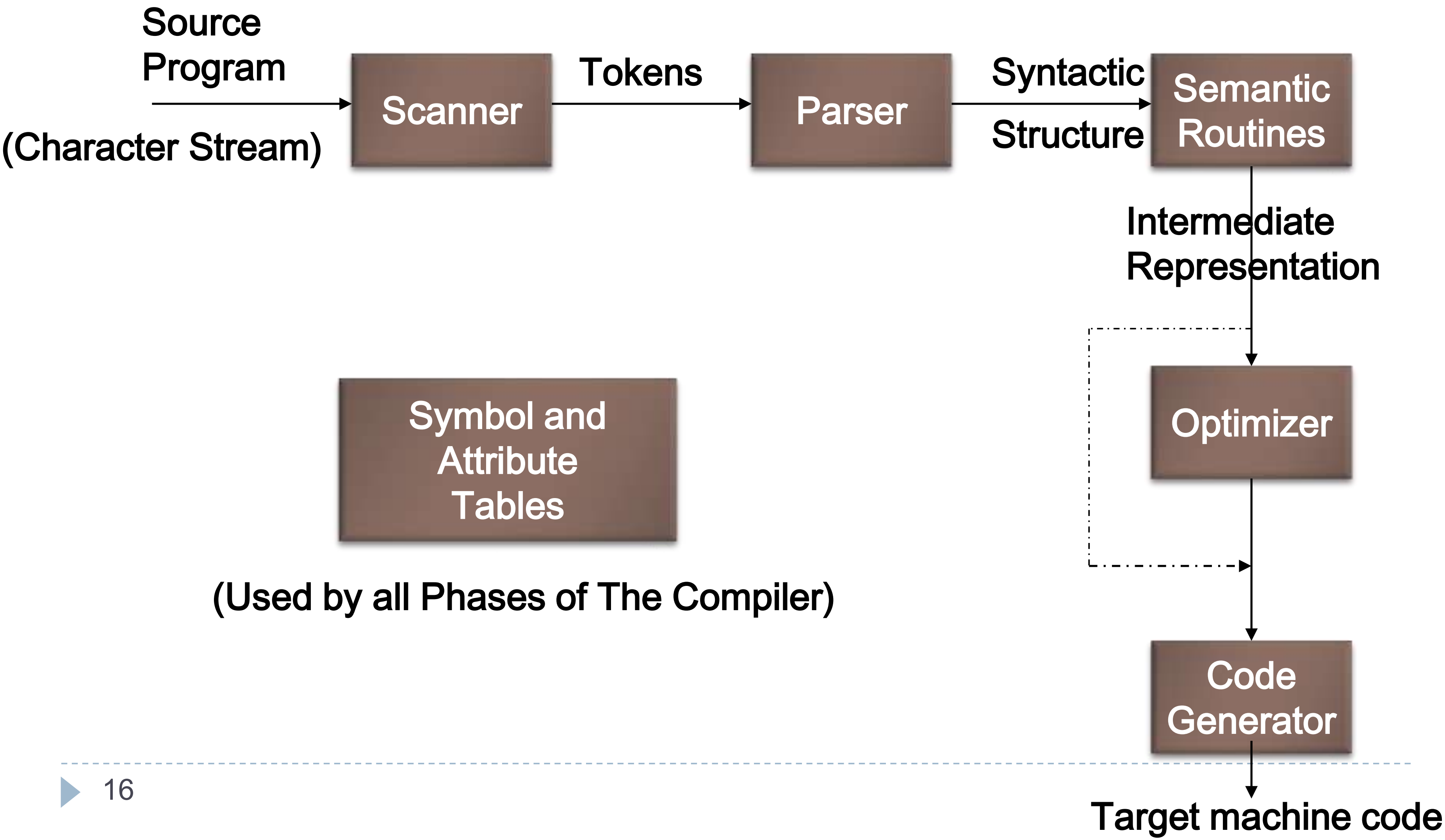
Supporting Phases/ Activities for Analysis

- Symbol Table Creation / Maintenance
  - Contains Info (storage, type, scope, args) on Each “Meaningful” Token, Typically Identifiers
  - Data Structure Created / Initialized During Lexical Analysis
  - Utilized / Updated During Later Analysis & Synthesis
- Error Handling
  - Detection of Different Errors Which Correspond to All Phases
  - What Kinds of Errors Are Found During the Analysis Phase?
  - What Happens When an Error Is Found?

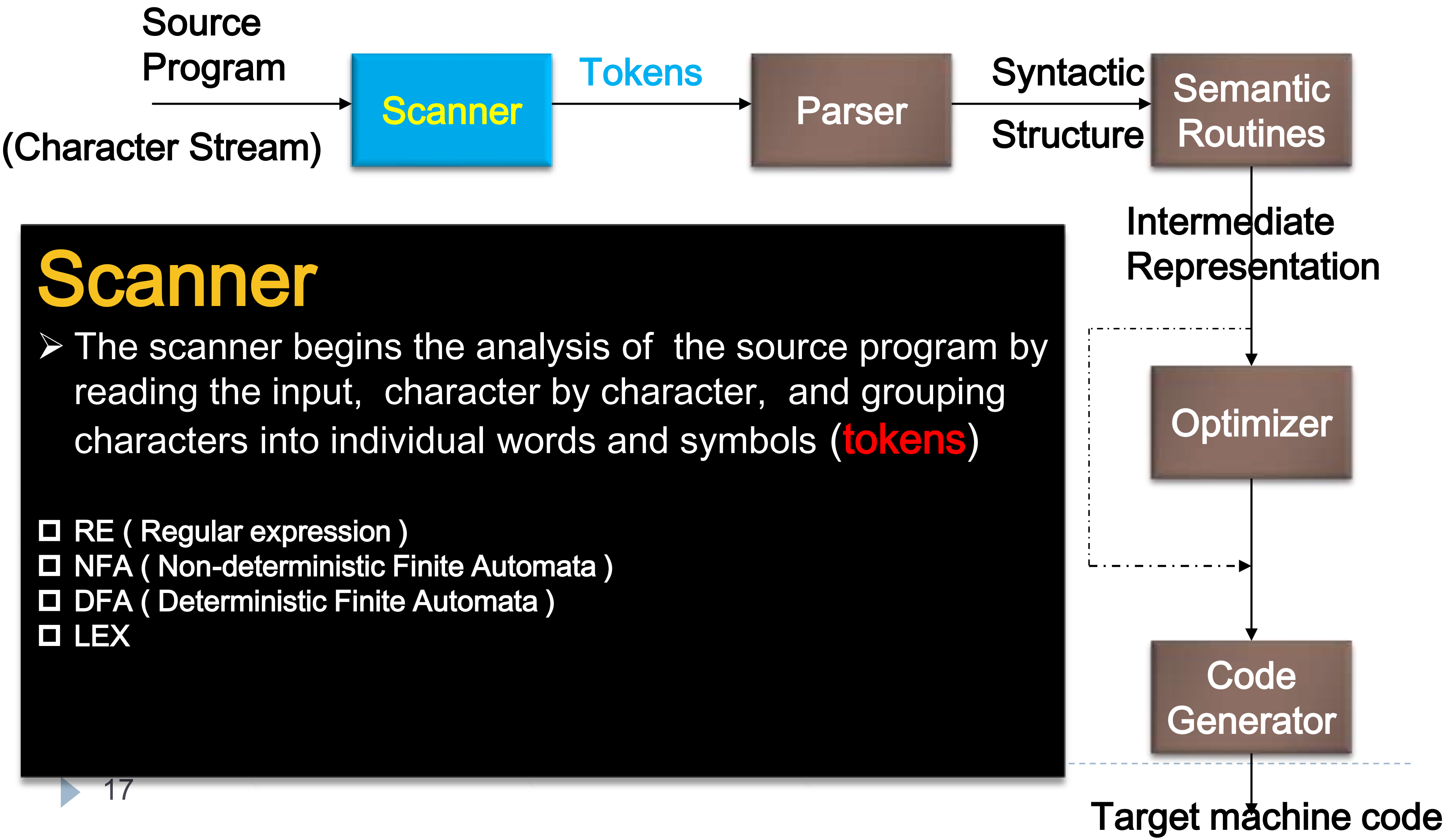
The Synthesis Task For Compilation

- Intermediate Code Generation
  - Abstract Machine Version of Code - Independent of Architecture
  - Easy to Produce and Do Final, Machine Dependent Code Generation
- Code Optimization
  - Find More Efficient Ways to Execute Code
  - Replace Code With More Optimal Statements
  - 2-approaches: High-level Language & “Peephole” Optimization
- Final Code Generation
  - Generate Relocatable Machine Dependent Code

The Structure of a Compiler (2)



The Structure of a Compiler (3)

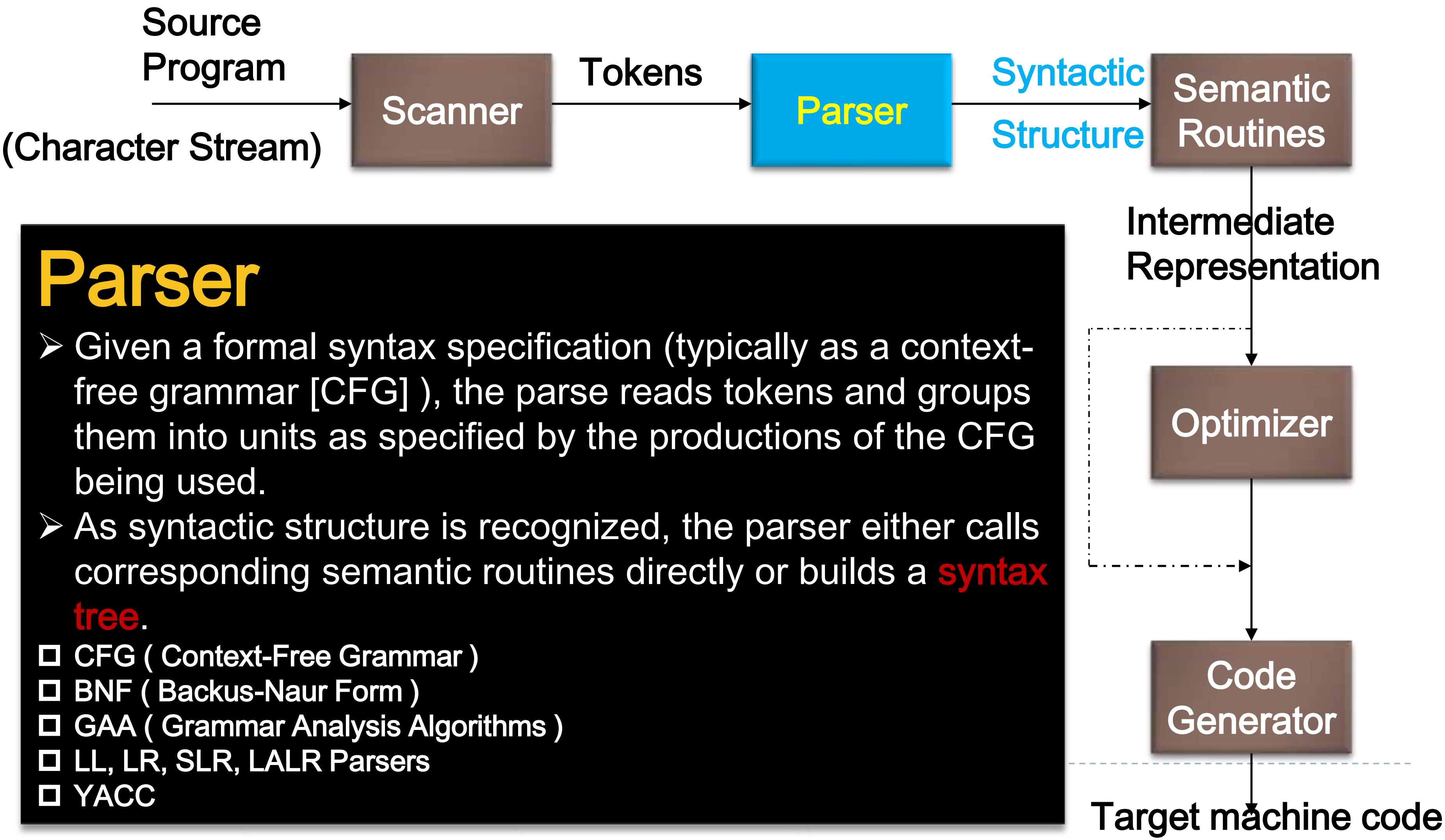


Scanner

- ➢ The scanner begins the analysis of the source program by reading the input, character by character, and grouping characters into individual words and symbols (**tokens**)

- □ RE (Regular expression)
- □ NFA (Non-deterministic Finite Automata)
- □ DFA (Deterministic Finite Automata)
- □ LEX

The Structure of a Compiler (4)

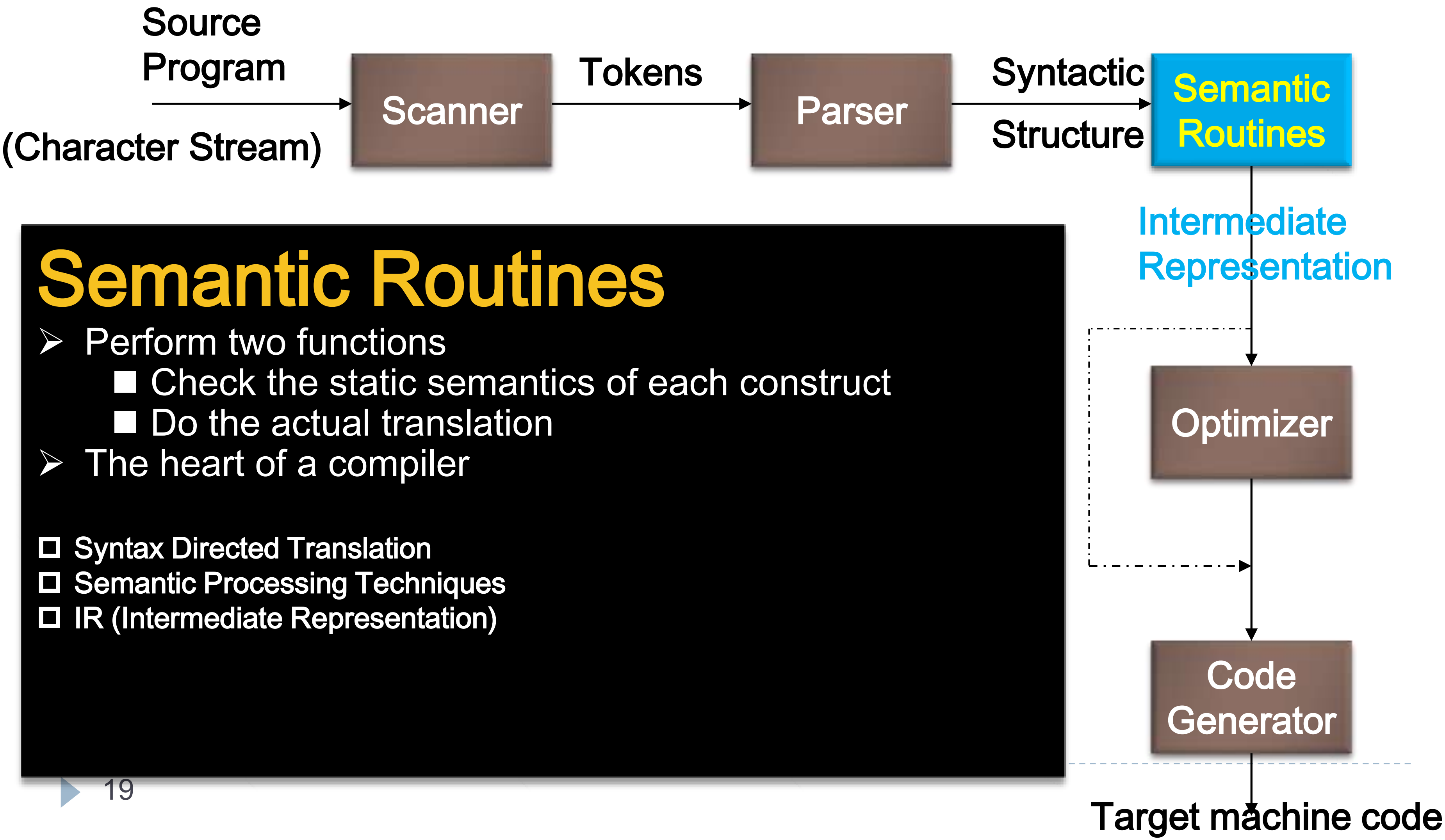


Parser

- ➢ Given a formal syntax specification (typically as a context-free grammar [CFG]), the parse reads tokens and groups them into units as specified by the productions of the CFG being used.
- ➢ As syntactic structure is recognized, the parser either calls corresponding semantic routines directly or builds a syntax tree.

□ CFG (Context-Free Grammar)
□ BNF (Backus-Naur Form)
□ GAA (Grammar Analysis Algorithms)
□ LL, LR, SLR, LALR Parsers
□ YACC

The Structure of a Compiler (5)

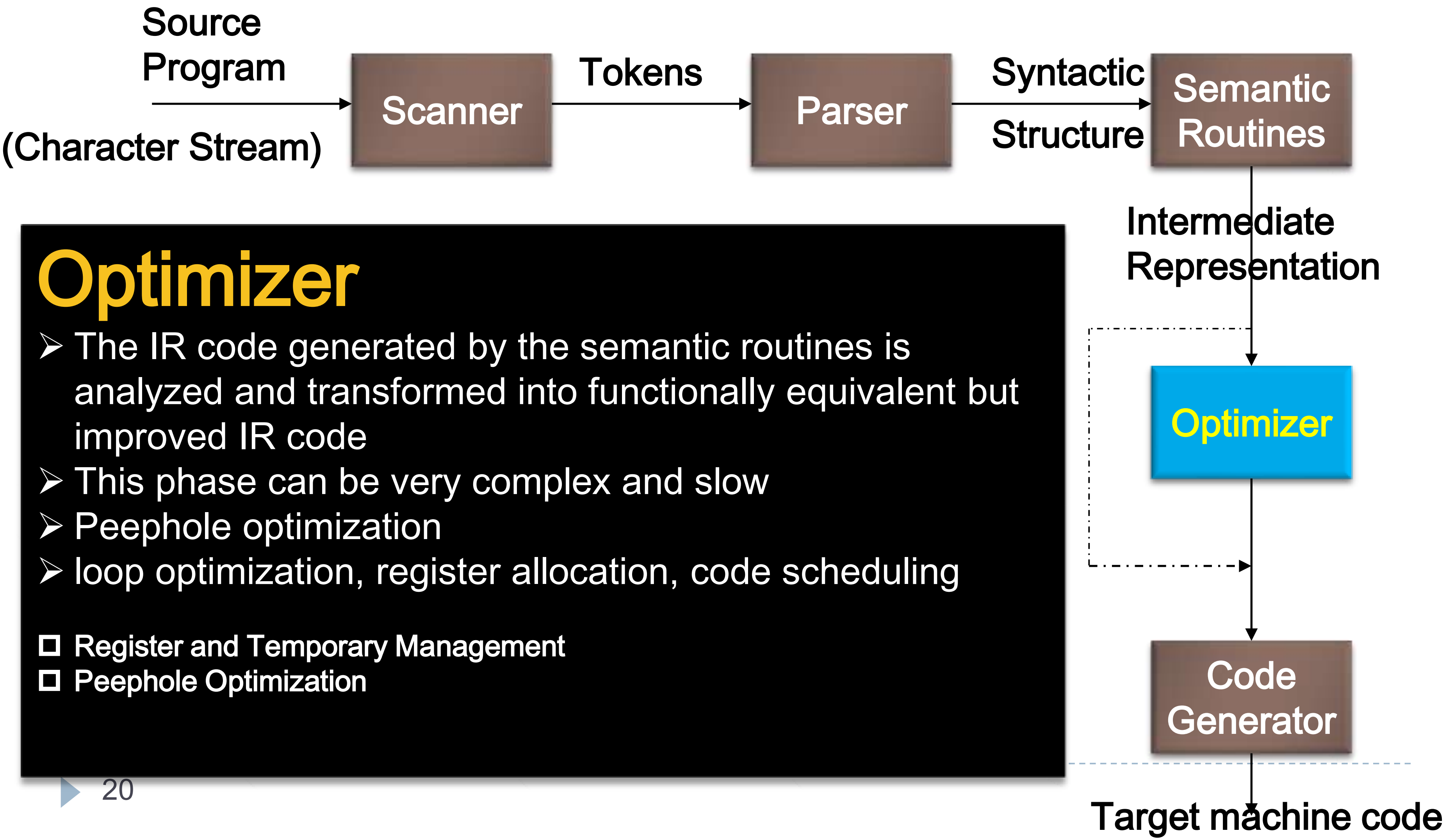


Semantic Routines

- ➢ Perform two functions
  - ■ Check the static semantics of each construct
  - ■ Do the actual translation
- ➢ The heart of a compiler

□ Syntax Directed Translation
□ Semantic Processing Techniques
□ IR (Intermediate Representation)

The Structure of a Compiler (6)

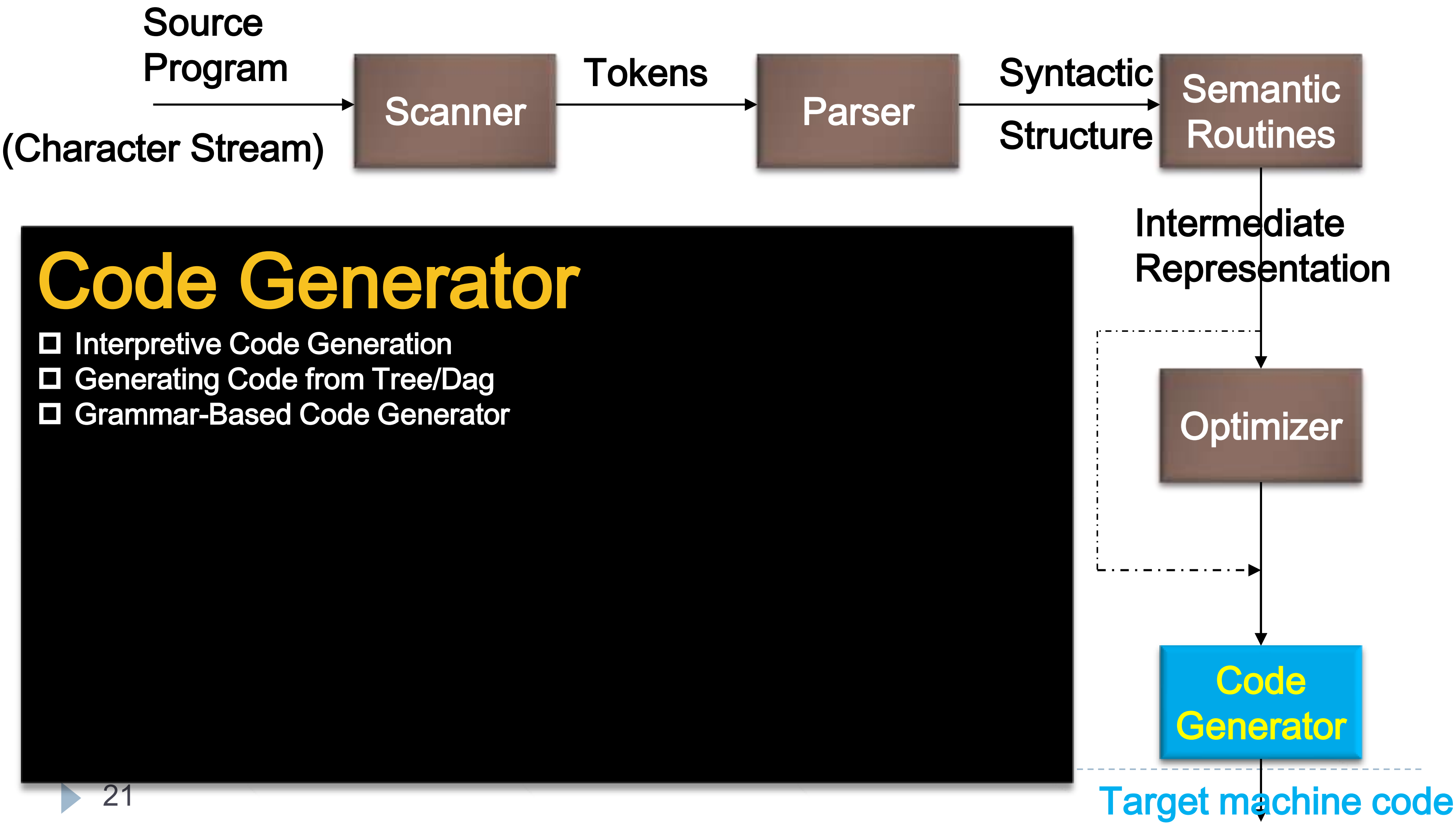


Optimizer

- ➢ The IR code generated by the semantic routines is analyzed and transformed into functionally equivalent but improved IR code
- ➢ This phase can be very complex and slow
- ➢ Peephole optimization
- ➢ loop optimization, register allocation, code scheduling

□ Register and Temporary Management
□ Peephole Optimization

The Structure of a Compiler (7)



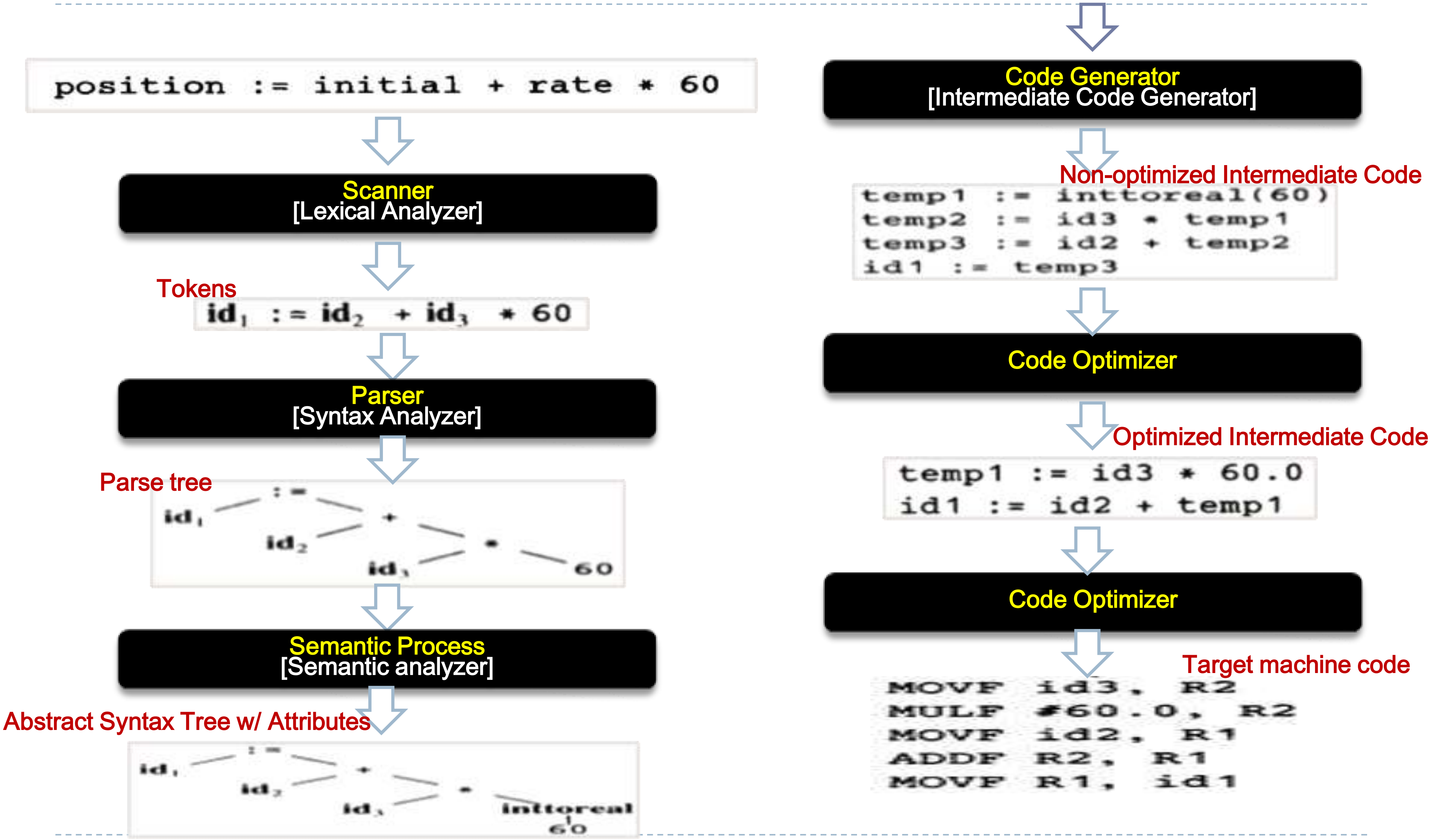
Code Generator

- □ Interpretive Code Generation
- □ Generating Code from Tree/Dag
- □ Grammar-Based Code Generator

The Structure of a Compiler (8)

SYMBOL TABLE

1	position	...
2	initial	...
3	rate	...
4		



```
position := initial + rate * 60
```

Scanner
[Lexical Analyzer]

Tokens

$id_1 := id_2 + id_3 * 60$

Parser
[Syntax Analyzer]

Parse tree

Semantic Process
[Semantic analyzer]

Abstract Syntax Tree w/ Attributes

Code Generator
[Intermediate Code Generator]

Non-optimized Intermediate Code

```
temp1 := inttoreal(60)
temp2 := id3 * temp1
temp3 := id2 + temp2
id1 := temp3
```

Code Optimizer

Optimized Intermediate Code

```
temp1 := id3 * 60.0
id1 := id2 + temp1
```

Code Optimizer

Target machine code

```
MOVF id3, R2
MULF #60.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1
```

Video on Compilers

1. Introduction to Compiler
2. Application of Phases of Compiler

Questions..

1. Define Compiler?
2. List few applications of Compiler.
3. Explain the phases of compiler?
4. What is mean by token?

Homework..

1. What is parser?

2What is mean by analysis and synthesis.

3. Describe the following example.

area=pi * r * r + 45