Module 10

Compiler Basics

Module Ten

- Compiler Basics - Part Four
- In this presentation, we are going to talk about :
- Semantic Analysis

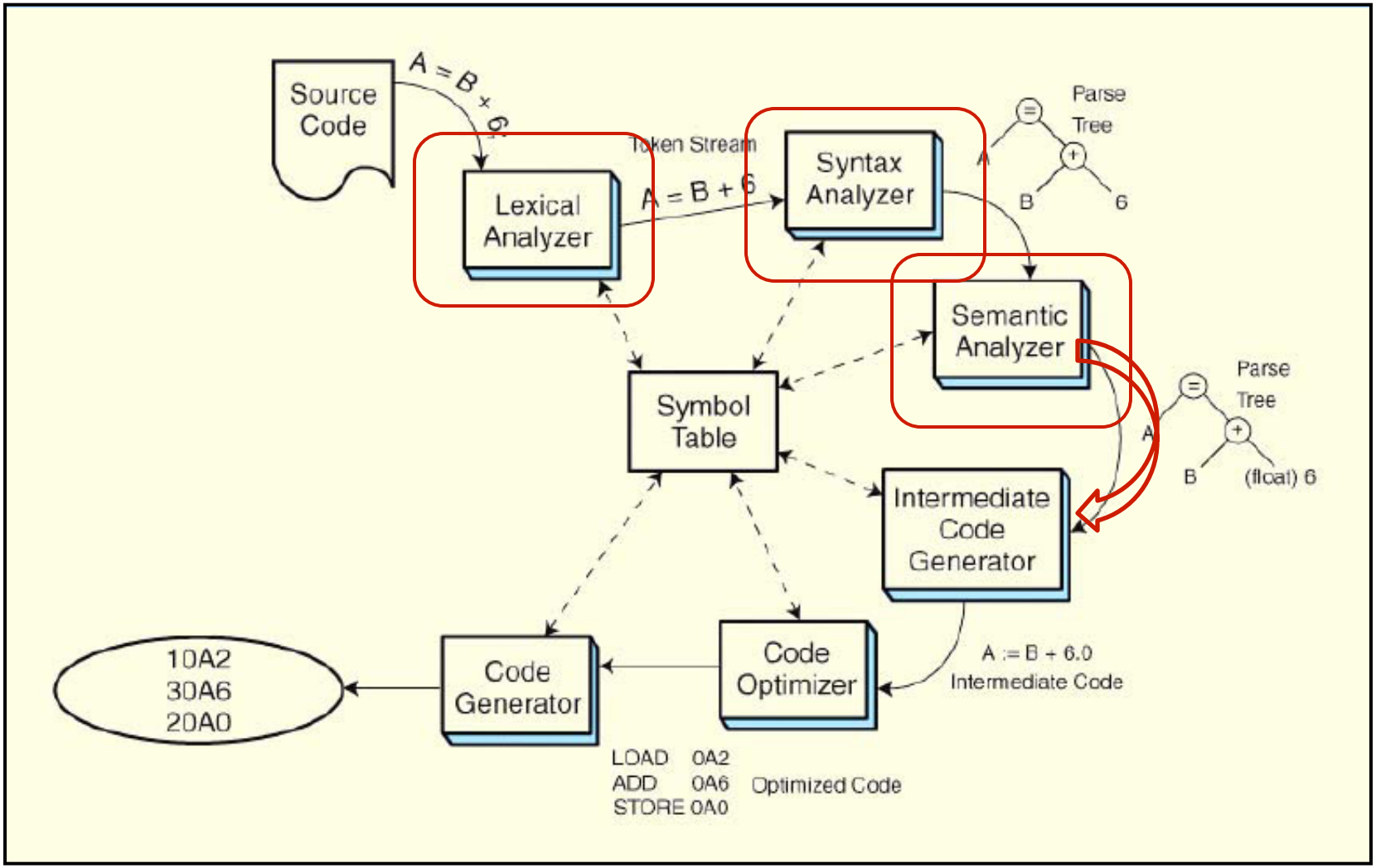
Overview

- Previously we talked about:

- Compiler Basic Functions
- Language Definition – Grammar
- Lexical Analysis
- Syntax Analysis

Now: Semantic Analysis

Where are we ?


Source Code
A = B x 6
Lexical Analyzer
Token Stream
A = B + 6
Syntax Analyzer
Parse Tree
=
A
+
B
6
Semantic Analyzer
Symbol Table
Intermediate Code Generator
Parse Tree
=
A
+
B
(float) 6
A := B + 6.0
Intermediate Code
Code Optimizer
Code Generator
10A2
30A6
20A0
LOAD 0A2
ADD 0A6
STORE 0A0
Optimized Code

Semantic Analysis

- Add the meaning and generate the code.
- Next task of the compiler.
- As each structure of the source code is recognized, call the Semantic routine to generate the actual code.
- Usually generate machine code directly, but often generate Assembly.
- Can generate an intermediate code form for optimization.
- Specific code is independent of the Parsing, but clearly dependent on the specific machine.

Type checking

- Operator - Operand compatibility
  - Concatenate strings, not floating point numbers
- Flow control
  - Break when not in structure
- Uniqueness
  - Single definition of symbol name
- Name consistency
  - Where required, names match

Intermediate code generator

- Created as the Parser recognizes each construct of the program
- Create three address code

  x := y op z

- Quadruples

  op y z x

- Facilitates Optimization and
  Target-Code generation

Code Generator examples

- Parser finds:

```
sum  :=  sum  +  value

sumSQ := sumSQ + value * value

WRITE(mean,variance)
```

Code generated:

```
+     sum     value     t1
:=    t1                sum

*     value   value     t2
+     sumSQ   t2        t3
:=    t3                sumSQ

PARAM mean
PARAM variance
CALL  xWRITE
```

Semantic Analysis

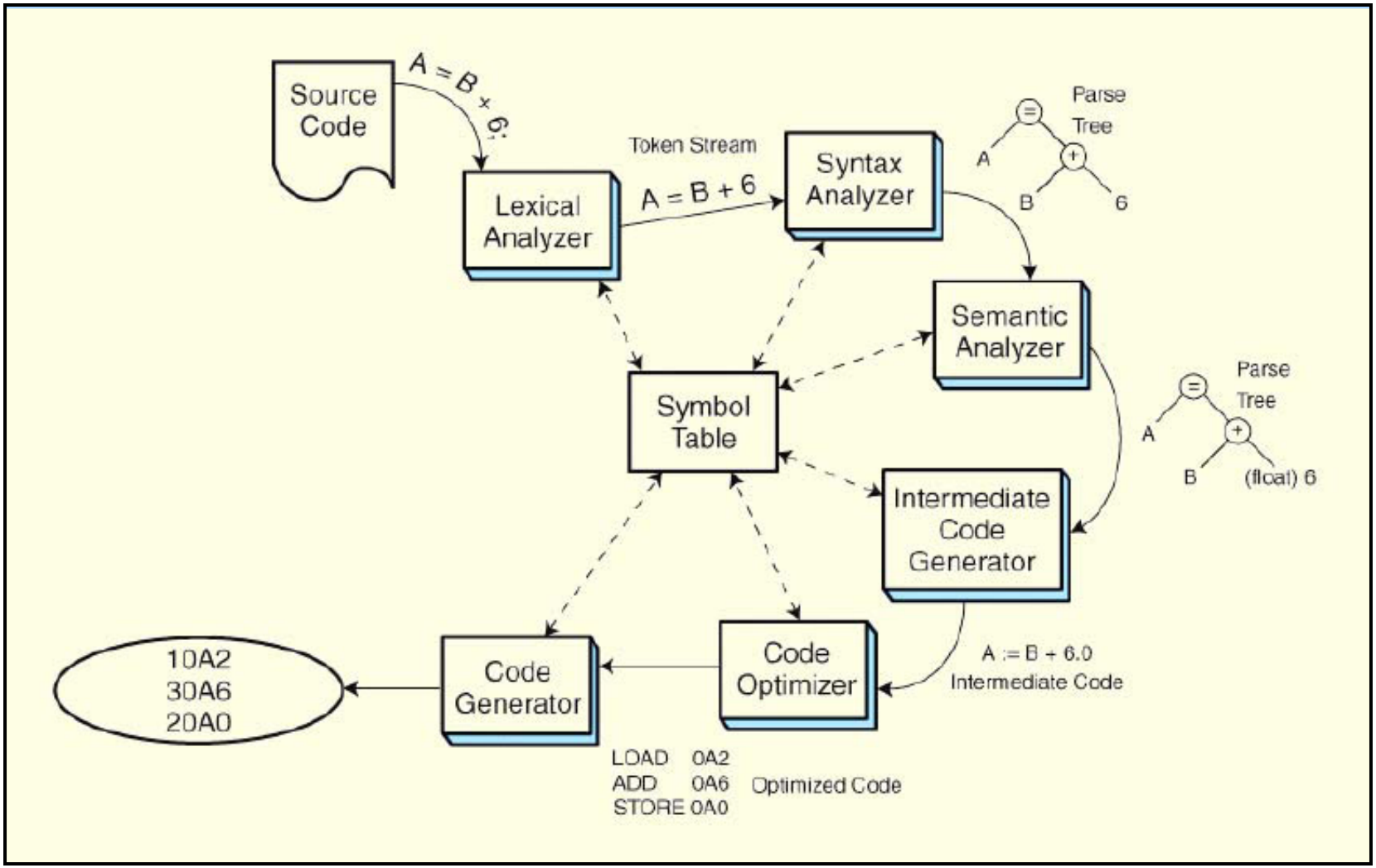
- Variable type conversions
- Reserving space for variables
- Addressing ARRAY objects
- Space for subroutine arguments and return values
- Structures for flow control
- Labels

Semantic Analysis

- Memory management
- Type of target program - Linker Loader concerns
- Instruction selection
- Register allocation
- Order of execution

- Use of subroutines for common code sequences.

Review


Source Code
A = B x 6;
Lexical Analyzer
Token Stream
A = B + 6
Syntax Analyzer
Parse Tree
A
B
6
Semantic Analyzer
Symbol Table
Intermediate Code Generator
Parse Tree
A
B
(float) 6
Code Optimizer
A := B + 6.0
Intermediate Code
Code Generator
10A2
30A6
20A0
LOAD 0A2
ADD 0A6
STORE 0A0
Optimized Code

Summary

- Basic Compiler Functions
    - Grammar
    - Lexical Analysis
    - Syntactical Analysis
        - Operator-Precedence Parsing
        - Recursive-Descent Parsing
    - Semantic Analysis
        - Code Generation