CSc 553
Principles of Compilation

01. Background

Saumya Debray

The University of Arizona

Tucson, AZ 85721

Course Objectives

- Understand advanced concepts in compiler design and implementation, esp. program analysis and optimization.
- Understand how source language programs are implemented at the machine level.

Compilers

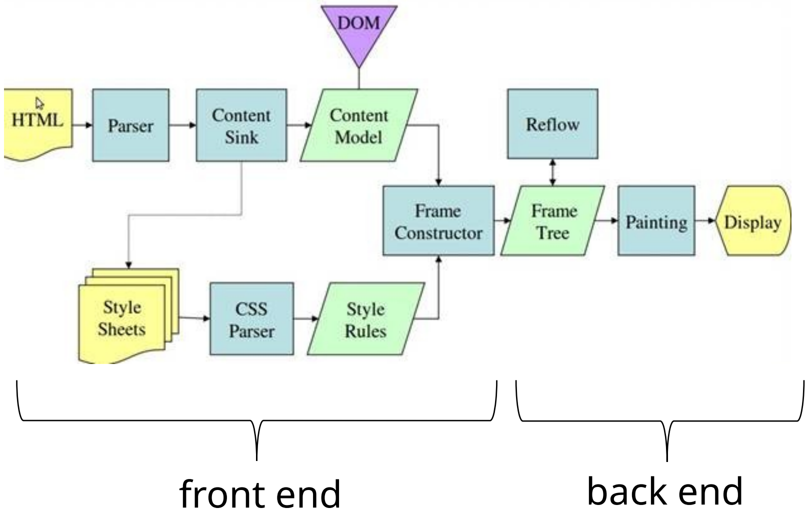
A *compiler* (more generally, *translator*) maps *source language strings* to "equivalent" *target language strings*. E.g.:

- gcc : C/C++ programs to assembly/machine code
- f2c : Fortran programs to C programs
- latex2html: Latex documents to HTML documents
- javac : Java programs to JVM byte code
- ps2pdf: PostScript files to PDF files

A very commonly used translator:
The Web Browser

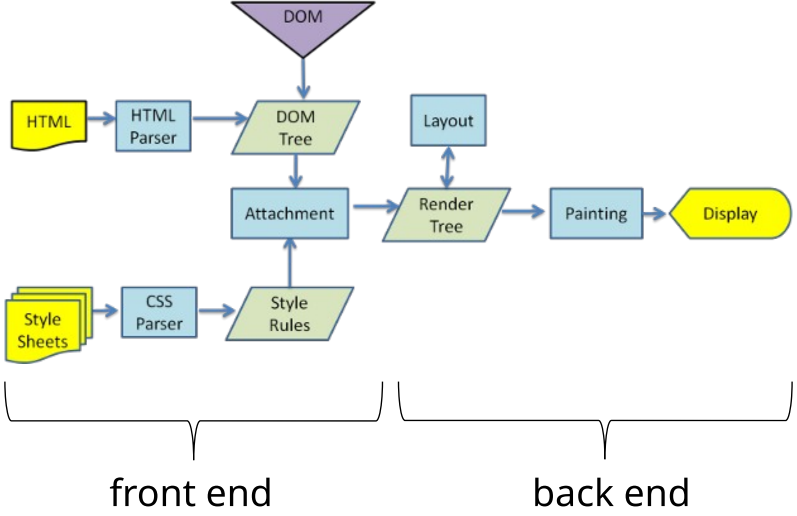
Gecko

(rendering engine for Firefox):



Webkit

(rendering engine for Safari and Chrome):

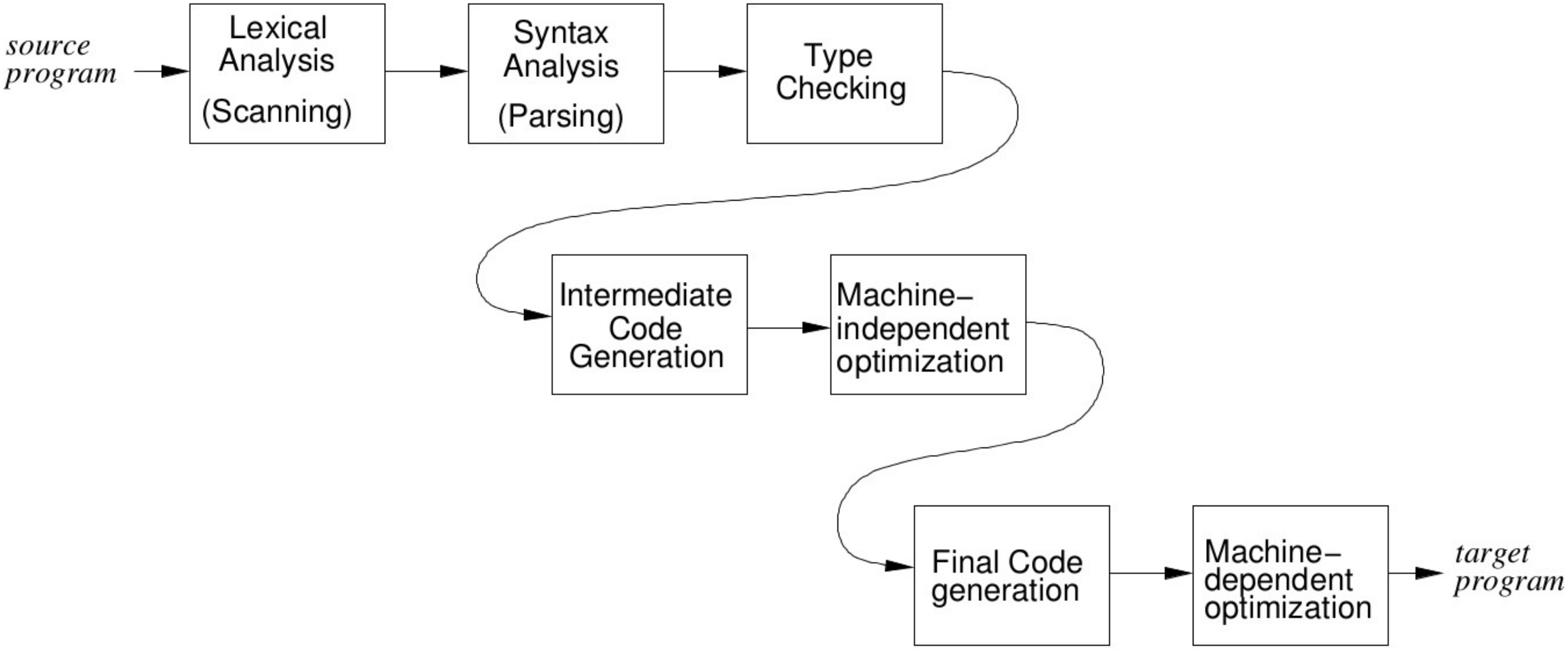


Source: *How Browsers Work: Behind the Scenes of Modern Web Browsers*, by Tali Garsiel & Paul Irish. http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

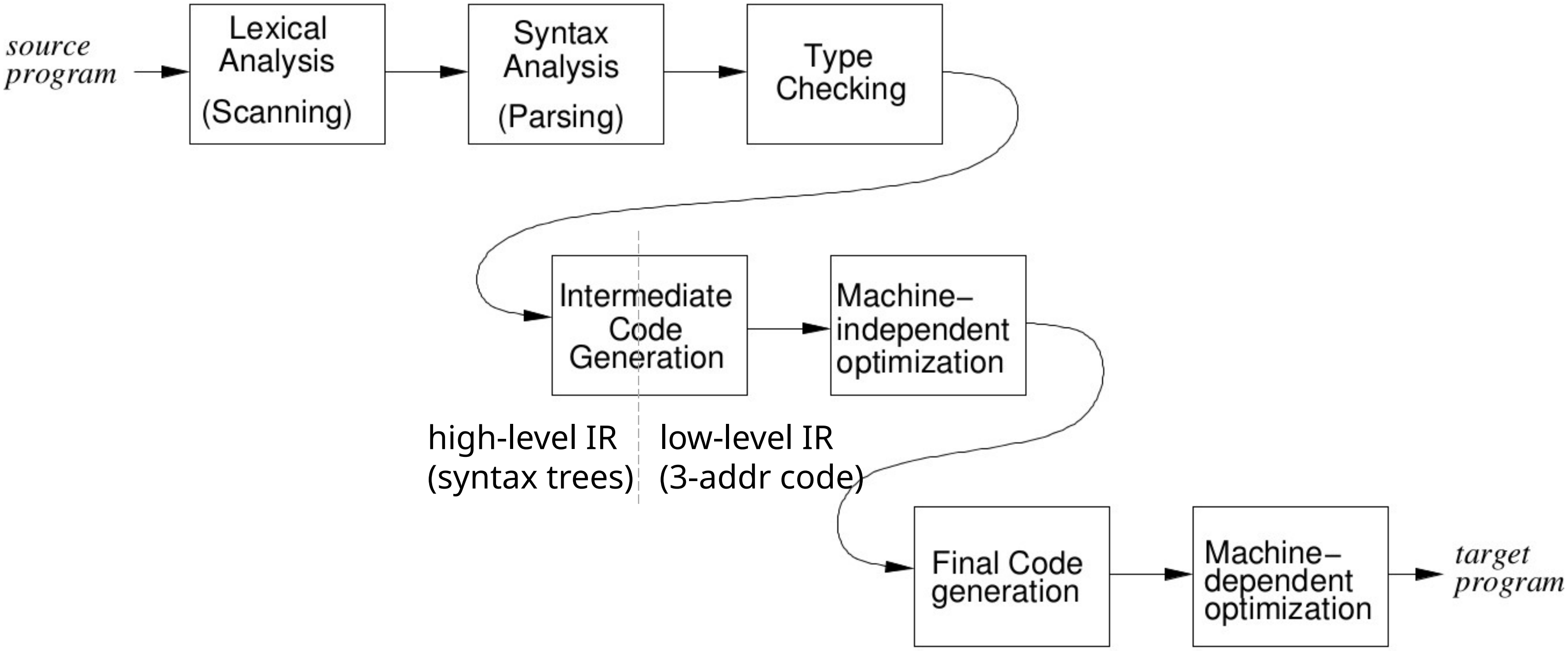
Languages

- *Syntax*:
  - "structural" aspects of program units.
  - specified by a grammar.
- *Semantics:*
  - the "meaning," i.e., behavior, of program units.
  - specified using ***actions*** associated with grammar rules.

Structure of a compiler (“phases”)


source program
Lexical Analysis (Scanning)
Syntax Analysis (Parsing)
Type Checking
Intermediate Code Generation
Machine–independent optimization
Final Code generation
Machine–dependent optimization
target program

Structure of a compiler ("phases")



source program

Lexical Analysis (Scanning)

Syntax Analysis (Parsing)

Type Checking

Intermediate Code Generation

Machine–independent optimization

high-level IR
(syntax trees)

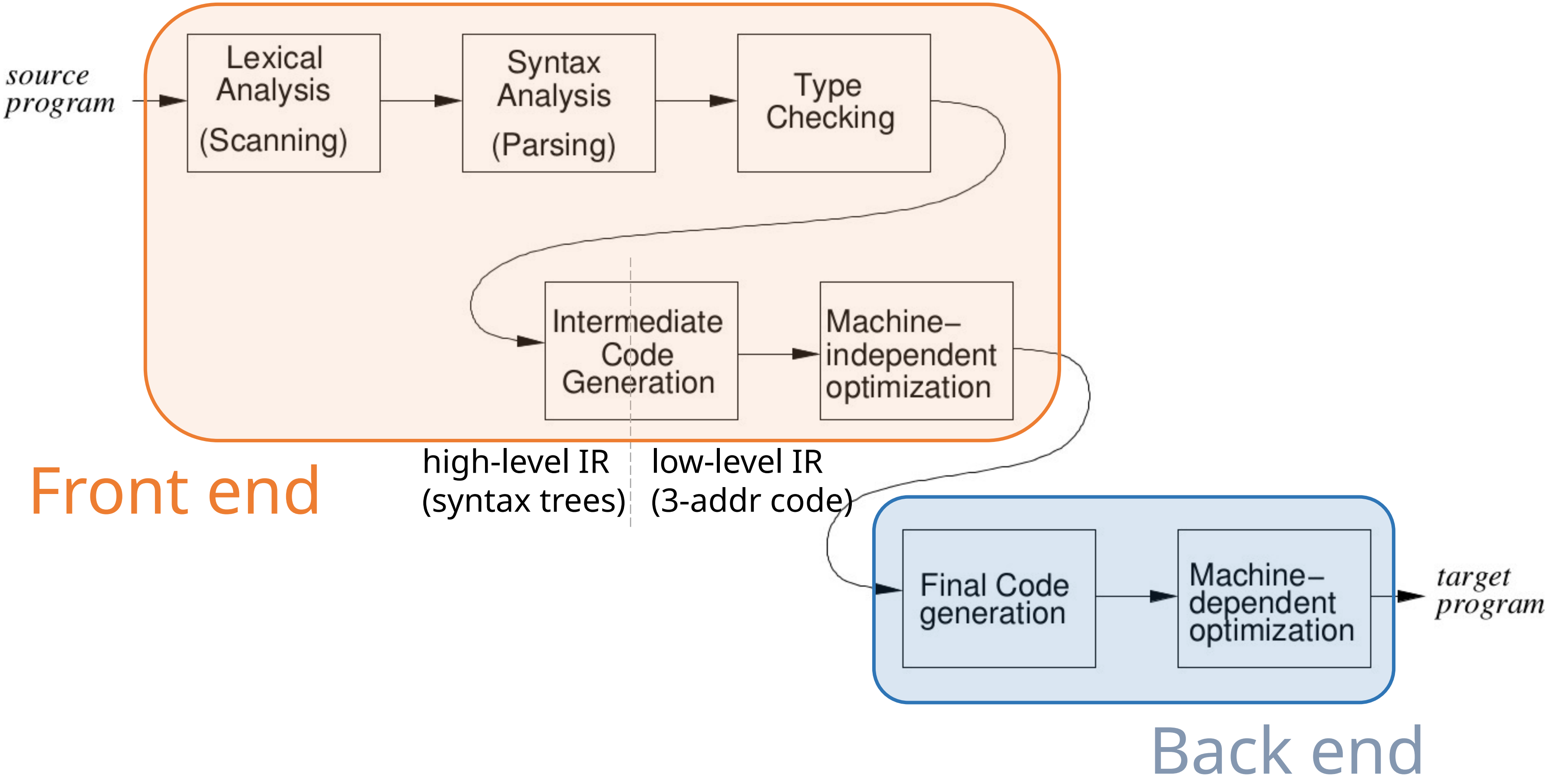
low-level IR
(3-addr code)

Final Code generation

Machine–dependent optimization

target program

Structure of a compiler ("phases")


source program
Lexical Analysis (Scanning)
Syntax Analysis (Parsing)
Type Checking
Intermediate Code Generation
Machine–independent optimization
high-level IR (syntax trees)
low-level IR (3-addr code)
Front end
Final Code generation
Machine–dependent optimization
target program
Back end

Phases of a Compiler

1. Lexical analysis (“scanning”)
   - Reads in program, groups characters into “tokens”
2. Syntax analysis (“parsing”)
   - Structures token sequence according to grammar rules of the language.
3. Semantic analysis
   - Checks semantic constraints of the language.
4. Intermediate code generation
   - Translates to “lower level” representation.
5. Program analysis and code optimization
   - Improves code quality.
6. Final code generation.

Grouping of Phases

Front end: machine-independent phases:

Lexical analysis

Syntax analysis

Semantic analysis

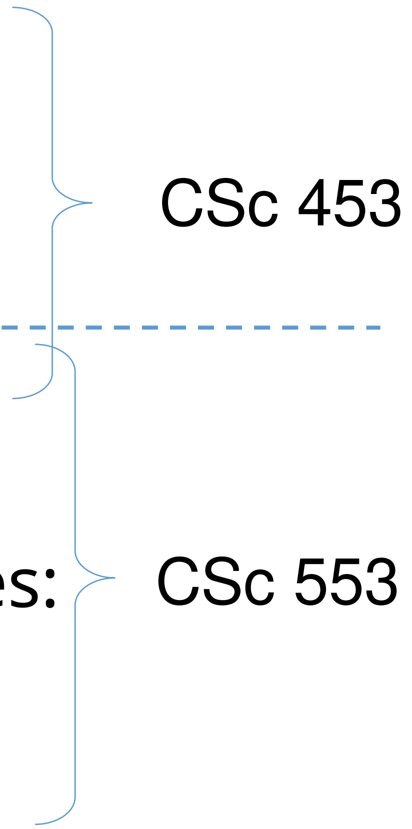
Intermediate code generation

some code optimization

Back end: machine-dependent phases:

Final code generation

machine-dependent optimizations


CSc 453
CSc 553