Lecture 1.

Compiler constructions.

lecturer : Alexander.

assist : Daniel .

overspecify if you donte have time to finish the project.

Slides by Eva Rose
Krist Rose.

nyu/courses/spring19. / .

$$\begin{tikzcd}
\text{Source program} \arrow[d] \\
\text{Compiler} \arrow[d] \\
\text{target program.}
\end{tikzcd}$$

→. semantically equivalent programs.

→. Source programs : typically high level.

→. target programs : typically assembler or object / machine code

object code $\neq$ object-oriented.

1 Roles of compiler

→. allow programming at high level.
execute at low level.

→. write once, run everywhere.

→ help in verifying software.

→. early discovery of programming errors.

→ provide automatic code optimization.

ex. loop-fusion
loop-ficient.

History

→ · Grace Hopper .

She piorneered the concept of write in high level lang and compile and execute

Authors: Alfred. Aho, Jeffrey D. Ullman

textbook dragon book.

Language. Processor.

Compiler. a program (written in meta-language) that translates a program into a sematically equivalent. program.

Interpreter: a program (written in a. meta-lang) for executing another program.

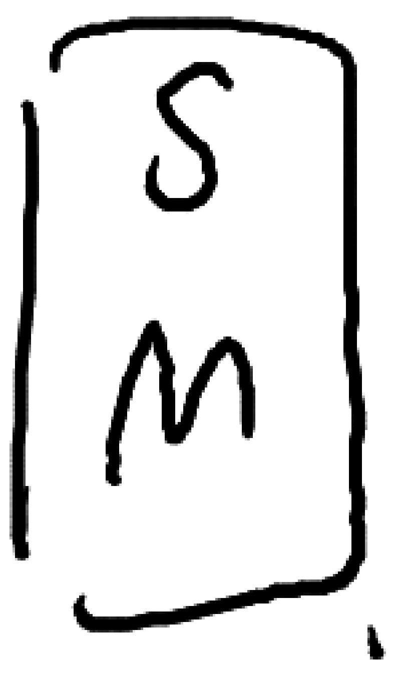
① $\rightarrow$ Usually compilers faster than interpretter.

② $\rightarrow$ Interpreters. usually better at error diagnostics.

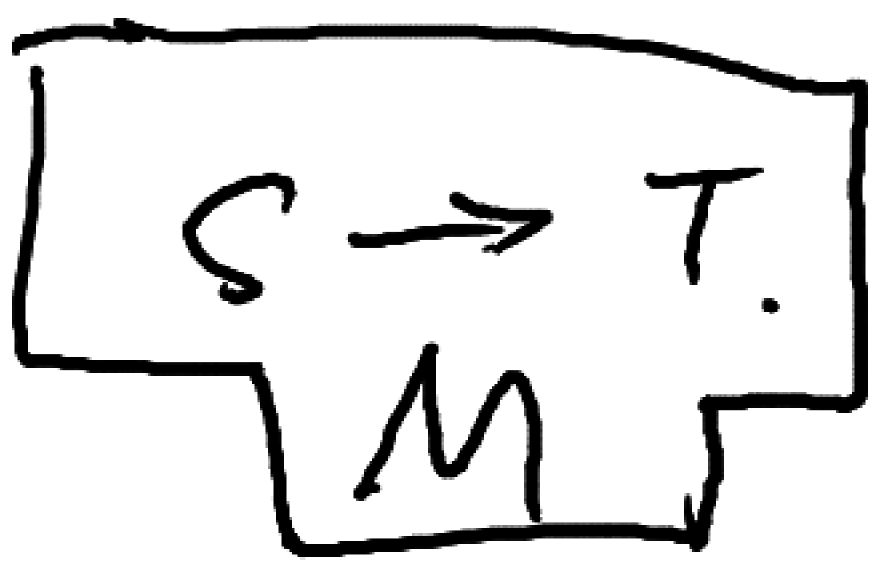
① main reason: interpretter cannot optimize with the limited context it sees.

② main reason: error appears in source language rather than converted.

Interpreter diagrams. I-diag



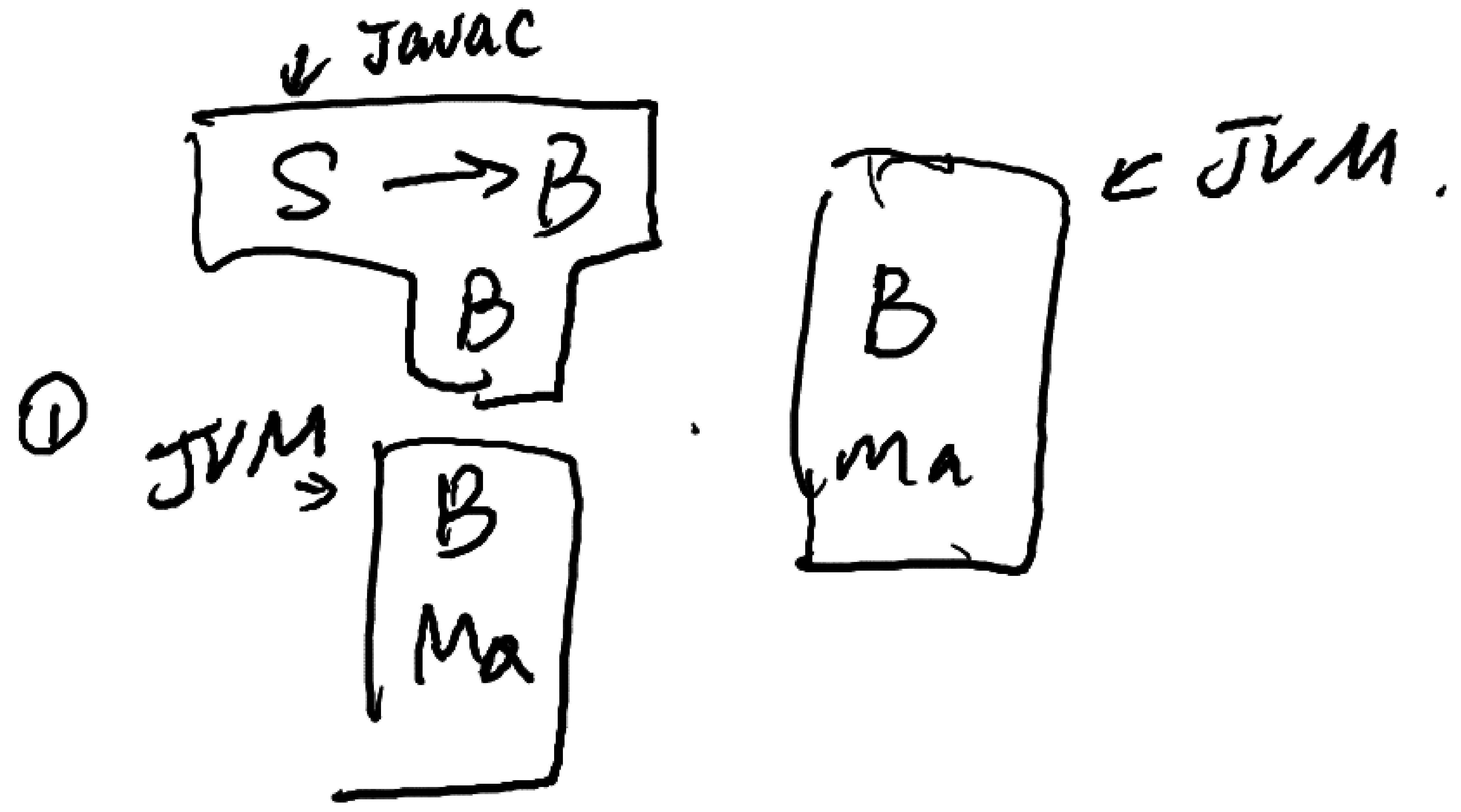
Compiler - T diag.



M: meta lang.

T: targeted lang.

Hybrid: The Java Compiler.



B: Bytecode lang.

☆. JVM ① JVM used for Javac Java compiler itself.

(self-hosting language) ☆.

JVM ②. used for interpret the compiled Java bytecode program.

Lang-Processing System.

→ Preprocessor: macro & simple ops.

→ Compiler. : . . .. . .

→ Assembler: symbolic machine code.
→ relocatable binary code.

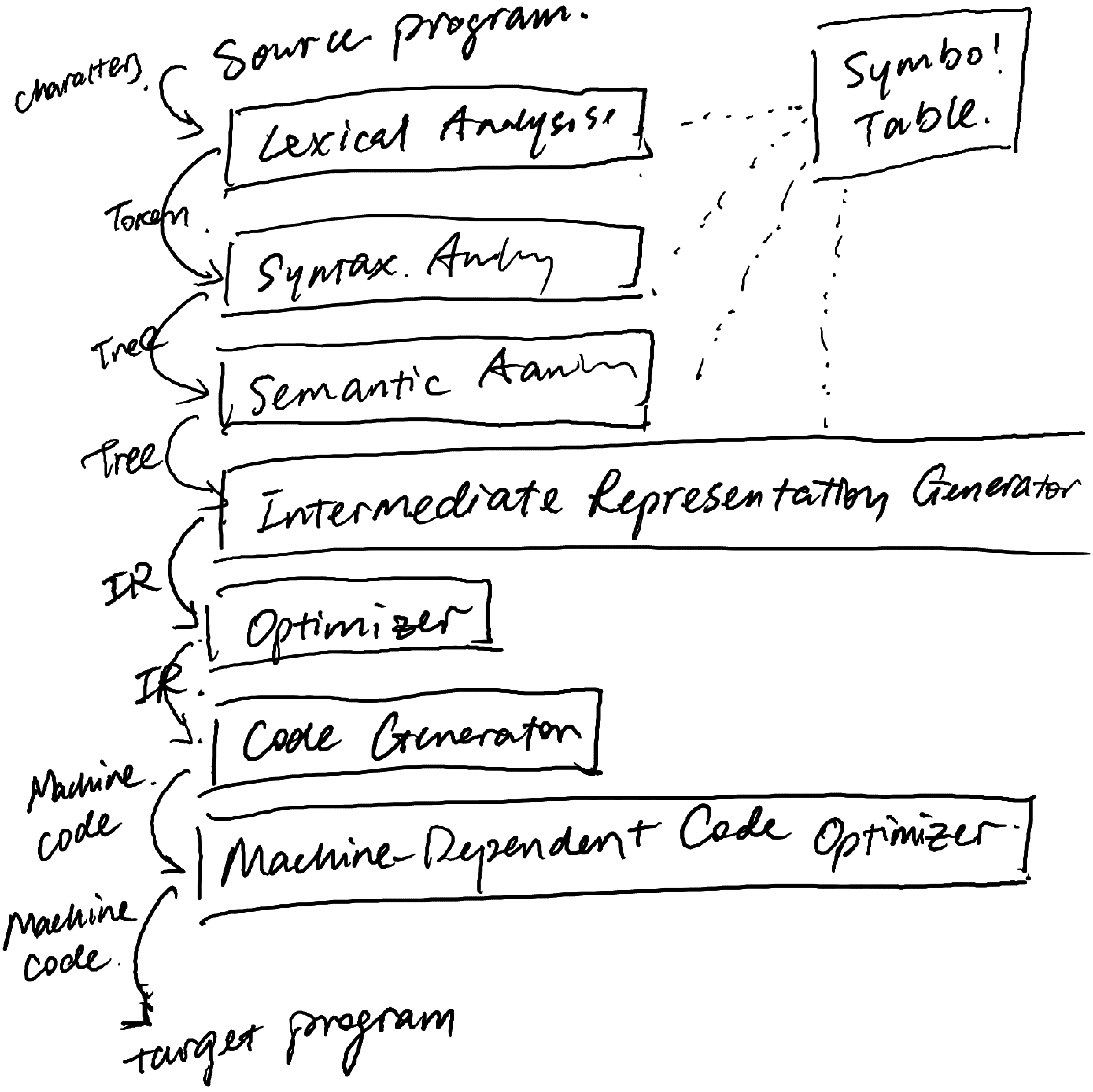
→ Linker: resolves links to library files and other relocatable object files.

→ Loader: combines executable object files in memory

ex. .COM file memory loaders.

Structure of a compiler.

Transformation phases.



Why Two Optimizors ?

→ . Different stage create different optimizing opportunities.

Ex.

Source came as <u>stream of chars</u>.

<u>Lexemes</u>.

<u>Lexical Analysis</u>.

scanned into table.

<u>Syntax Analysis</u>.

precedence appear at this stage.

$$\begin{tikzcd}
 & = \arrow[dl] \arrow[dr] & & & \\
\langle id,1\rangle & & + \arrow[dl] \arrow[dr] & & \\
 & \langle id,2\rangle & & * \arrow[dl] \arrow[dr] & \\
 & & \langle id,3\rangle & & \langle num,60\rangle
\end{tikzcd}$$

Semantic Analysis

enriched with semantic info.
(explicit type conversion).

AST CST.

Abstract syntax tree.
Concrete syntax tree.

IR Gen

$$\begin{tikzcd}
 & = \arrow[dl] \arrow[dr] & & & \\
{} & & + \arrow[dr] & & \\
 & & & * \arrow[r] & \text{inttofloat.} \\
 & & & & \langle num, 60 \rangle .
\end{tikzcd}$$

t1 = inttofloat(60).
t2 = id3 * t1.
t3 = id2 + t2.
id1 = t3.

optimize into.
t1 = id3 * 60.0.
id1 = id2 + t1.

optimize criteria:
faster,
shorter,
less power $\searrow$
app.

Code Gen.

actually generate machine code.

```
LDF  R2, id3.
MULF R2, R2, #60.0
LDF  R1, id2.
```

Symbol table.

records are < var name, attributes>.

Attributes:

- → storage info.
- → type.
- → scope.
- → procedure. names; number & types of arguments.
- → argument passing. (by val or by ref)
- → return type.

Design principle: find, store, retrieve records quickly.

Compiler-construction Tools.

Commonly used tools:

- → scanner generators: token → lexical analys
- → parser generators: grammar → syntax analyz
- → syntax-directed translation engines: syntax tree ↓ IR.
- → code-generator generators: rules → code gen
- → data-flow engines: data-flow info analyzers.

Compiler generators: integrated set of the above.

Course Description.

textbook : the Dragon book.

11 lectures and homework assignments.

1 special topic.

2 exams.

Semester-long programming project.

Grading

15% . hw.

15% midterm.

25% final.

45% project — Implement fully functional compiler.

Source : ChocoPy

Target : RISC-V assembly

Implem Lang : Java.

Team-up. 3 parts.

ChocoPy

A dialect of Python designed UCB
for teaching compilers.

chocopy.org

→ Familiar : runnable in Python.
→ statically typed .
→ Expressive .

RISC-V. ("risk five" pronounce).
→ Reduced instruction set computers (RISC)
use a small set of general instructions.
→ RISC-V is an open-source architecture
based on RISC.
→ Has online and offline simulators.

Implementation Lang.

→ Java. ~ 5 KLoC given.
 another ~5 K LoC to write.

→ Will use lexer and parser generaters. (JFlex and CUP). (anteler).

→. Only use another language if you seek challenge.

Team:

→. Working in 3-4 person teams.

→ 3 milestones: parser, type checker, code generator

→ Submit code and write-up.