

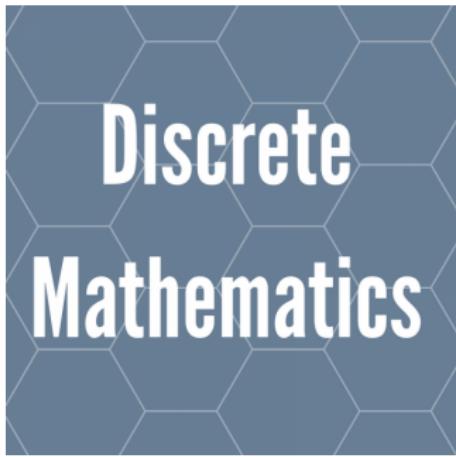
编译原理概述

魏恒峰

hfwei@nju.edu.cn

2022 年 11 月 07 日 (周一)



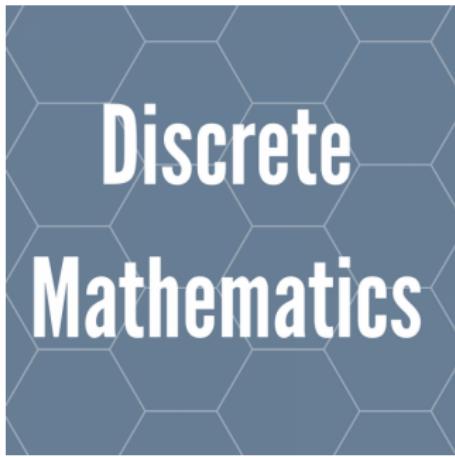


Discrete Mathematics



Compiler Construction

吾与离散数学，孰难？



吾与离散数学，孰难？

对于本学期课程，编译器实践比编译器理论更重要

7 周 = 14 次课 < 8 周 = 16 次课



本学期课程的考核方式做了较大调整

平时作业 (15 分): 7 + 1 次作业, 每周 1 ~ 2 题

期末测试 (40 分): 考试周统一安排; 2 小时; 开卷

课程实验 (45 分): 4 次必做实验 + 1 次选做实验 (5 分)

平时作业 (15 分): 7 + 1 次作业, 每周 1 ~ 2 题

期末测试 (40 分): 考试周统一安排; 2 小时; 开卷

课程实验 (45 分): 4 次必做实验 + 1 次选做实验 (5 分)

这是去年的啦 ...

平时作业 (00 分): 7 次作业, 每周 1 ~ 2 题

平时作业 (00 分): 7 次作业, 每周 1 ~ 2 题

课程实验 (60 分): 7 次必做实验 + 1 次选做实验 (5 分)

平时作业 (00 分): 7 次作业, 每周 1 ~ 2 题

课程实验 (60 分): 7 次必做实验 + 1 次选做实验 (5 分)

期末测试 (40 分): 考试周统一安排; 2 小时; **开卷**

平时作业 (00 分): 7 次作业, 每周 1 ~ 2 题

课程实验 (60 分): 7 次必做实验 + 1 次选做实验 (5 分)

期末测试 (40 分): 考试周统一安排; 2 小时; **开卷**

附加作业 (05 分): 报告 + 录屏方式

每周三晚上发布作业

下周三 23 : 55 前自愿提交作业



邀请码: 5JP57Q5H

发布阅读材料

发布调查问卷

发布作业答案

课程实验：开发 SysY 语言编译器



小步迭代、多步迭代、贴合课堂讲授进度

L0: 环境配置 今晚 18:00 发布

(20221107 ~ 20221113)



L0: 环境配置 今晚 18:00 发布

(20221107 ~ 20221113)



L1: 词法分析 本周三 18:00 发布

(20221109 ~ 20221116)

附加作业：报告 + 录屏方式，了解更现代的编译器原理与技术

The screenshot shows a GitHub repository page. At the top, the repository name is 'courses-at-nju-by-hfwei / compilers-papers-we-love' and it is labeled 'Public'. Below the repository name are navigation links for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', and 'Wiki'. Underneath these are buttons for 'master' (with a dropdown arrow), '1 branch', '0 tags', 'Go to file', and 'Add file'. The main content area lists various directory entries with their descriptions:

- hengxin +stringtemplate, +bison-ag
- attributed-grammars +stringtemplate, +bison-ag
- cfg +papers
- code-generation +stringtemplate, +bison-ag
- compilers +papers for parsers
- history +papers for parsers
- ir +papers for parsers
- optimization +papers for parsers
- parsing +papers for parsing
- programming-languages +paper: FunctionalStyle
- re-nfa-dfa +papers
- teaching +papers for parsers

鼓励讨论，但需独立编码完成课程实验



鼓励讨论，但需独立编码完成课程实验



课程实验：抄袭者当次实验计 0 分

附加作业：抄袭者当次作业计 0 分

请**经常**使用 Git/GitHub 保留代码编写记录 (OJ 系统原生支持)



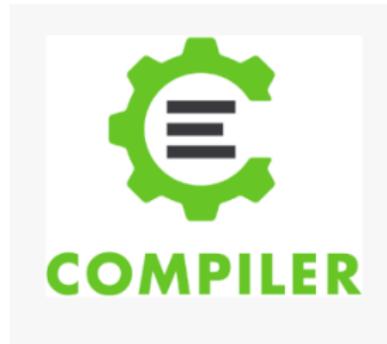
如果你不确定自己的行为是否构成抄袭, 请**事先**咨询助教。

QQ 群号: **755783220**



助教: 夏宇、潘昱光、顾龙、...

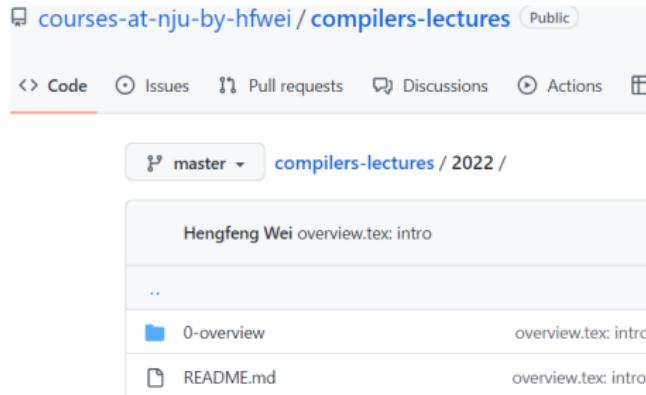
<http://47.96.123.231:8081>



搜索

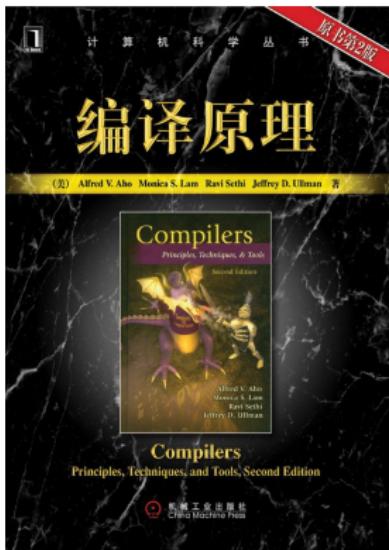
- > 编译原理课程简介
- > 课程要求
- > 教材选购
- > 环境配置
- > LLVM API使用手册
- > 实验

编译原理课程网站，请收藏并及时关注网站更新

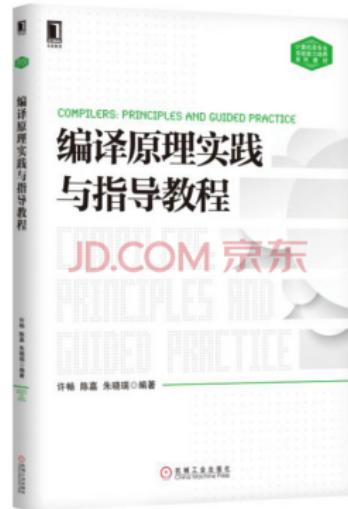


[https://github.com/courses-at-nju-by-hfwei/
compilers-lectures/tree/master/2022](https://github.com/courses-at-nju-by-hfwei/compilers-lectures/tree/master/2022)

(本书仅供参考)



也可使用“**本科教学版**”

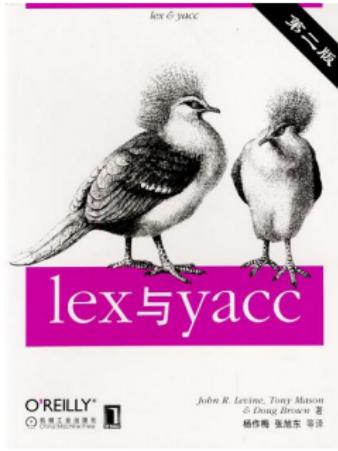


[https://cs.nju.edu.cn/
changxu/2_compiler/index.html](https://cs.nju.edu.cn/changxu/2_compiler/index.html)

(本学期对课程实验做了大幅改动)



Flex: 词法分析器生成器



Bison: 语法分析器生成器

不够现代, 本学期课程实验**不再支持**这些工具



(Since 1988)



Terence Parr (University of San Francisco)

<https://www.antlr.org/index.html>

<https://www.antlr.org/tools.html> (IntelliJ Plugin)

在线玩: <http://lab.antlr.org/>

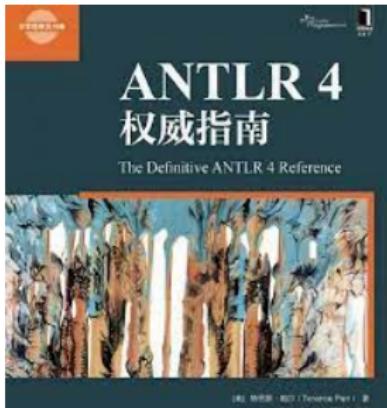
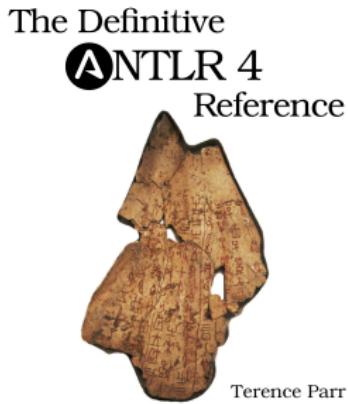
arithmetic: number1.txt, simple2.txt

fol: example1.txt

guitartab: example1.txt

xyz: example1.txt

C: add.c



基于 ANTLR 4，是课程实验指导的**重要**参考资料

Language Implementation Patterns

Create Your Own Domain-Specific and General Programming Languages

Terence Parr



Language
Implementation Patterns

编程语言 实现模式

Create Your Own Domain-Specific
and General Programming Languages

[译] Terence Parr 著
李直来 译
高鹏翔 审校



基于 ANTLR 3，与 ANTLR 4 相比，有些过时，
但可以看作理解 ANTLR 4 的基础

自制编译器

How to Develop a Compiler

[日] 青木峰郎 / 著 严基连 焦云 / 译

TING
图灵
图灵原书
设计丛书

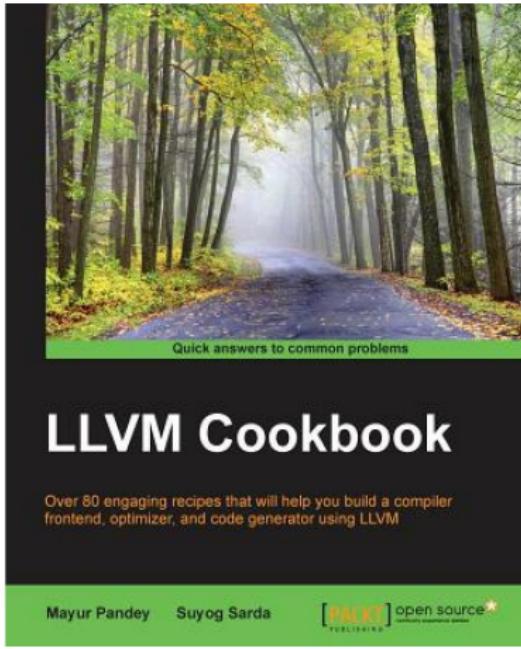


中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS

“从零开始制作真正的编译器”，对课程实验很有帮助，**强烈推荐**



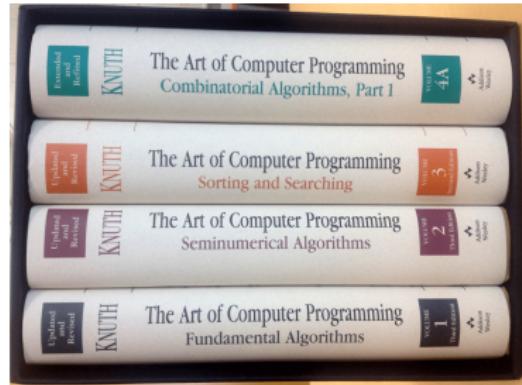
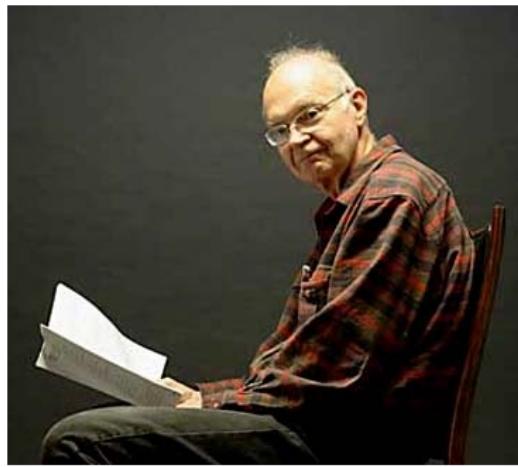
本学期课程实验引入了 LLVM (<https://llvm.org/>)



LLVM 简介 @ Bilibili



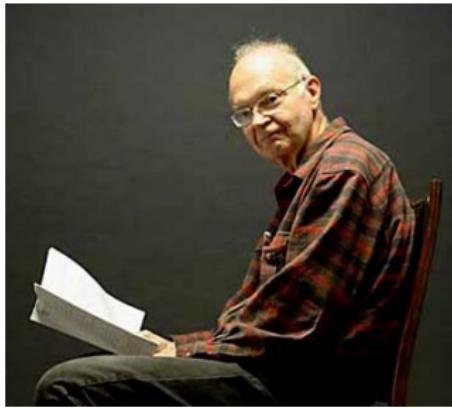
the “father of the analysis of algorithms”



(Since 1962; 计划 7 卷)

Donald E. Knuth (1938 ~)

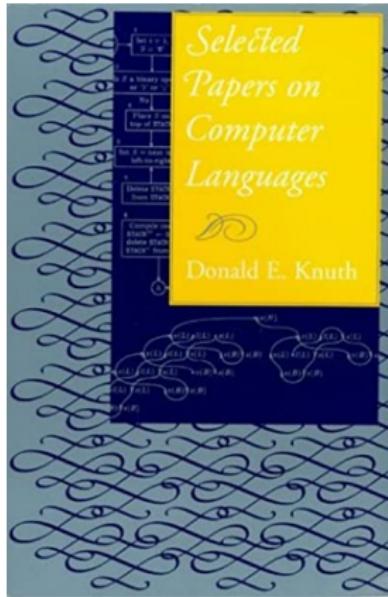
Turing Award, 1974



*“For his major contributions to **the analysis of algorithms** and **the design of programming languages**, and in particular for his contributions to “*The Art of Computer Programming*” through his well-known books in a continuous series by this title.”*

— Turing Award, 1974

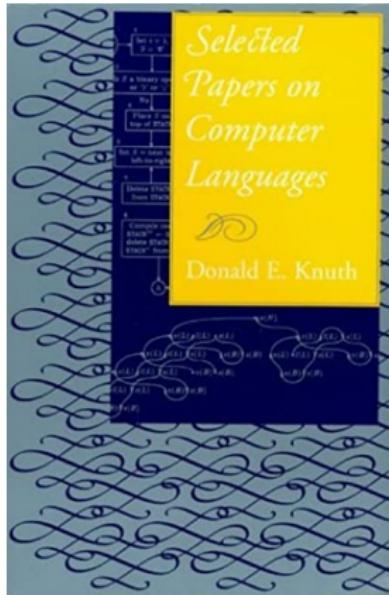
“Selected Papers on Computer Languages”



LR Parser (语法)

Attribute Grammar (语义)

“Selected Papers on Computer Languages”



LR Parser (语法)

Attribute Grammar (语义)

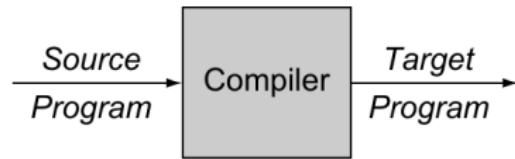
ALGOL 58 Compiler



“I got a job at the end of my senior year
to write a compiler for Burroughs”

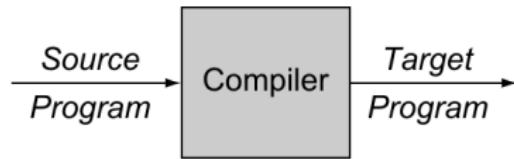
“高级”语言 \Rightarrow (通常) “低级”语言 (如, 汇编语言)

汇编语言经过**汇编器**生成机器语言



“高级”语言 \Rightarrow (通常) “低级”语言 (如, 汇编语言)

汇编语言经过汇编器生成机器语言



GopherJS - A compiler from Go to JavaScript

[godoc](#) [reference](#)

GopherJS compiles Go code ([golang.org](#)) to pure JavaScript code. Its main purpose is to give you the opportunity to write front-end code in Go which will still run in all browsers.

Q : 机器语言是如何跑起来的?

Q : 机器语言是如何跑起来的?

作业 (P1 ~ P8): <https://www.bilibili.com/video/BV1EW411u7th>

语言类应用程序

- ▶ 配置文件解析 (.properties)
- ▶ CSV 文件 (Comma-Separated Values)
- ▶ JSON 文件 (JavaScript Object Notation)

语言类应用程序

- ▶ 配置文件解析 (.properties)
- ▶ CSV 文件 (Comma-Separated Values)
- ▶ JSON 文件 (JavaScript Object Notation)
- ▶ SQL 引擎 (Structured Query Language)
- ▶ TLA⁺/TLAPS (TPaxos.tla)
- ▶ (Java) 字节码解释器
- ▶ C/C++ 语言编译器

语言类应用程序

- ▶ 配置文件解析 (.properties)
- ▶ CSV 文件 (Comma-Separated Values)
- ▶ JSON 文件 (JavaScript Object Notation)
- ▶ SQL 引擎 (Structured Query Language)
- ▶ TLA⁺/TLAPS (TPaxos.tla)
- ▶ (Java) 字节码解释器
- ▶ C/C++ 语言编译器
- ▶ 排版工具 (LATEX)
- ▶ 绘图工具 (TikZ, Dot/Graphviz)

语言类应用程序

- ▶ 配置文件解析 (.properties)
- ▶ CSV 文件 (Comma-Separated Values)
- ▶ JSON 文件 (JavaScript Object Notation)
- ▶ SQL 引擎 (Structured Query Language)
- ▶ TLA⁺/TLAPS (TPaxos.tla)
- ▶ (Java) 字节码解释器
- ▶ C/C++ 语言编译器
- ▶ 排版工具 (LATEX)
- ▶ 绘图工具 (TikZ, Dot/Graphviz)
- ▶ L-System (Cantor Set)

The tomassetti.me website has changed: it is now part of strumenta.com. You will continue to find all the news with the usual quality, but in a new layout.

Our articles

Here we talk about parsing,
tools and libraries, natural
language processing and
Kotlin

Edited by [Federico Tomassetti](#)



[tomassetti.me](#)

alda.io



alda

[install](#) [tutorial](#) [cheat sheet](#) [docs](#) [community](#)

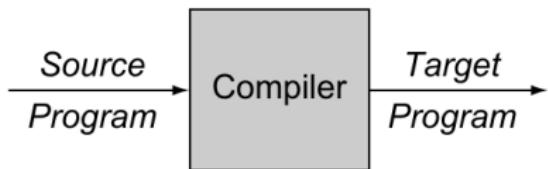
Alda is a text-based programming language for music composition. It allows you to write and play back music using only a text editor and the command line.

```
piano:  
o3  
g8 a b > c d e f+ g | a b > c d e f+ g4  
g8 f+ e d c < b a g | f+ e d c < b a g4  
<< g1/>g/>g/b/>d/g
```

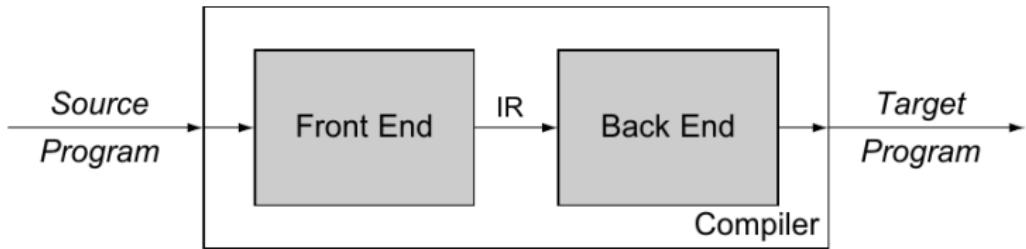
The language's design equally favors aesthetics, flexibility and ease of use.

alda @ 知乎

两个月的“编译器设计原理”之旅

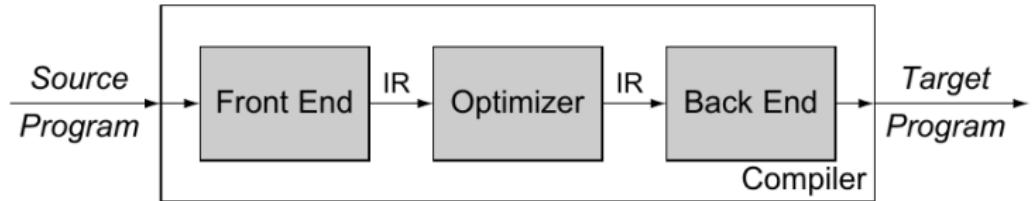


IR: Intermediate Representation (中间表示)



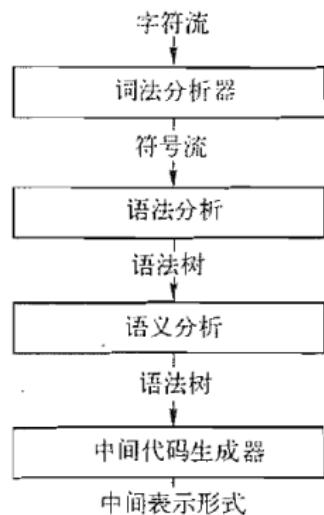
前端（分析阶段）: 分析源语言程序, 收集所有必要的信息

后端（综合阶段）: 利用收集到的信息, 生成目标语言程序

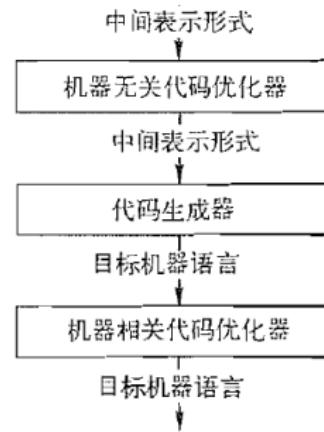


机器无关的中间表示优化

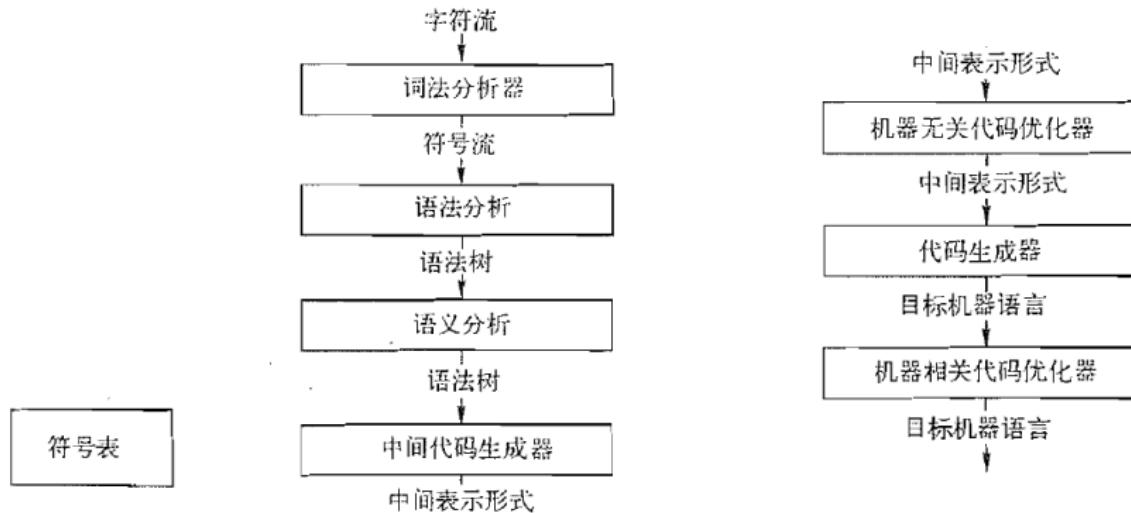
编译器前端：分析阶段



编译器后端：综合阶段



时间苦短，来不及优化



在设计实际生产环境中的编译器时，**优化**通常占用了大多数时间



方舟编译器

多端多语言，轻量低开销

立即下载

快速入门



Maple IR 及其各种优化技术



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

作为一名**程序员**, 你看到了什么?

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

作为一名**程序员**, 你看到了什么?

词法: 标识符、数字、运算符

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

作为一名**程序员**, 你看到了什么?

词法: 标识符、数字、运算符

语法: 包含算术运算的赋值语句

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

作为一名**程序员**, 你看到了什么?

词法: 标识符、数字、运算符

语法: 包含算术运算的赋值语句

语义: position, initial, rate 是数值类型

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

作为一名**程序员**, 你看到了什么?

词法: 标识符、数字、运算符

语法: 包含算术运算的赋值语句

语义: position, initial, rate 是数值类型

物理定律: 当前位置 = 初始位置 + 速度 × 时间

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

作为一名**程序员**, 你看到了什么?

词法: 标识符、数字、运算符

语法: 包含算术运算的赋值语句

语义: position, initial, rate 是数值类型

物理定律: 当前位置 = 初始位置 + 速度 × 时间

但是, 作为**编译器**, 它仅仅看到了一个**字符串**

词法分析器 (Lexer/Scanner): 将字符流转化为词法单元 (token) 流。

token : <token-class, attribute-value>

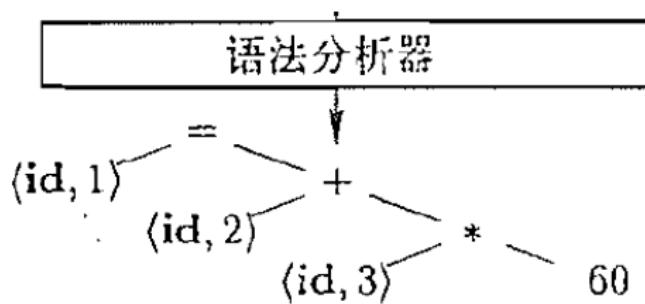
```
position = initial + rate * 60
```

<**id**, 1> <ws> <**assign**> <ws> <**id**, 2> <ws>
<+> <ws> <**id**, 3> <ws> <*> <ws> <**num**, 4>

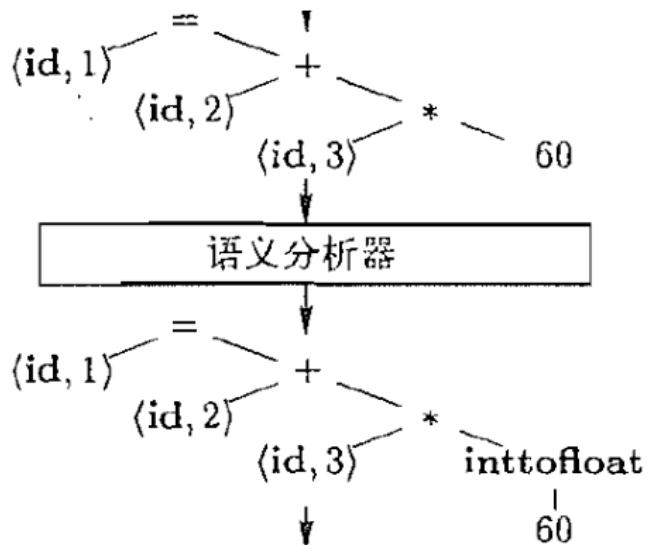
(此处, 1, 2, 3, 4 是指向**符号表**的指针)

语义分析器 (Parser): 构建词法单元之间的语义结构, 生成**语义树**

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

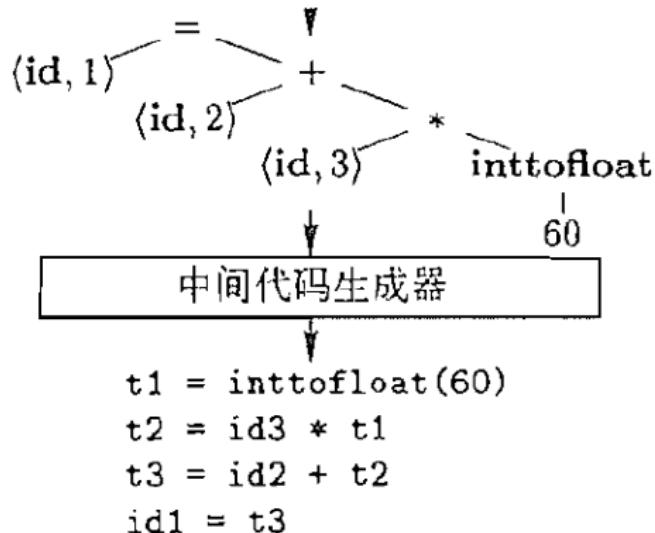


语义分析器：语义检查，如类型检查、“先声明后使用”约束检查



通过语法树上的遍历来完成

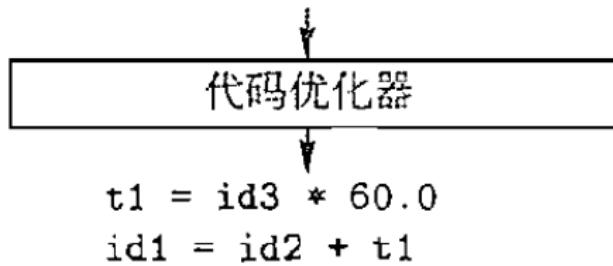
中间代码生成器：生成中间代码，如“三地址代码”



中间代码类似目标代码，但不含有机器相关信息（如寄存器、指令格式）

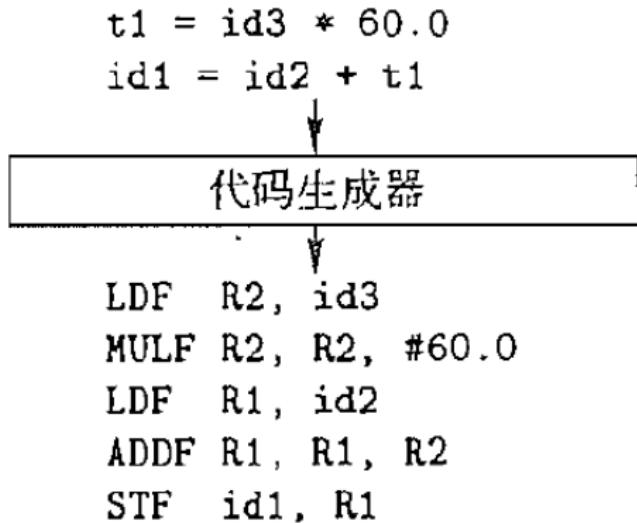
中间代码优化器

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



编译时计算、消除冗余临时变量

代码生成器: 生成目标代码, 主要任务包括**指令选择、寄存器分配**



符号表: 收集并管理变量名/函数名相关的信息

变量名:

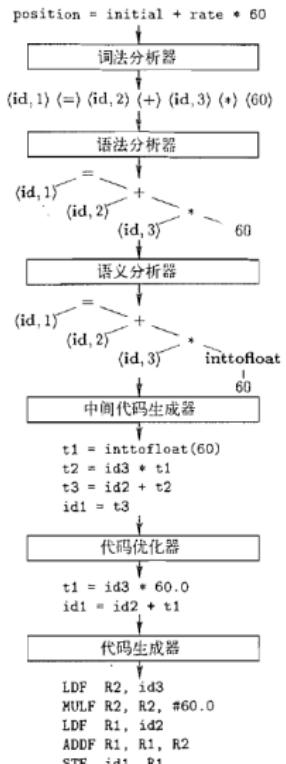
类型、寄存器、内存地址、行号

函数名:

参数个数、参数类型、返回值类型

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

符号表

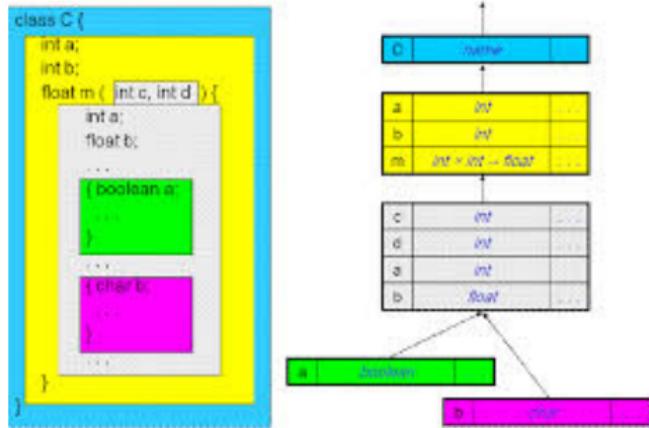


```
public class ST<Key extends Comparable<Key>, Value>
```

ST()	<i>create an empty symbol table</i>
void put(Key key, Value val)	<i>associate val with key</i>
Value get(Key key)	<i>value associated with key</i>
void remove(Key key)	<i>remove key (and its associated value)</i>
boolean contains(Key key)	<i>is there a value associated with key?</i>
int size()	<i>number of key-value pairs</i>
Iterable<Key> keys()	<i>all keys in the symbol table</i>

红黑树 (RB-Tree)、哈希表 (Hashtable)

为了方便表达嵌套结构与作用域, 可能需要维护多个符号表





Hello.g4

SimpleExpr.g4

Expr.g4

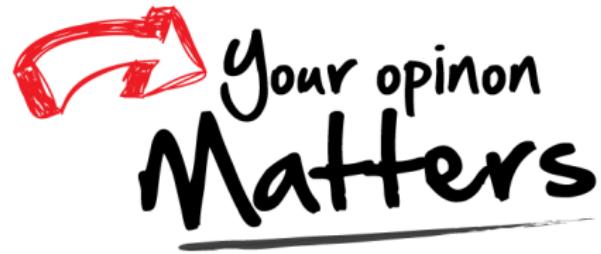
Hello.g4

SimpleExpr.g4

Expr.g4

grammars-v4 @ github

Thank You!



Office 926

hfwei@nju.edu.cn