编译原理概述

魏恒峰

hfwei@nju.edu.cn

2023 年 02 月 15 日 (周三)

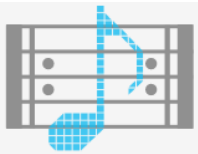
从**半学期**调整为**整学期**

从**选修课**调整为**必修课**

“那一天，人类终于回想起曾经一度被他们所支配的恐怖”

编译原理哦, 听起来怪怪的, 是不是很好玩 (好吃)?

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 repl
```

```
alda play -f .alda
```

alda @ 知乎

10 Demo: Alda @ youtube

Fractal Grower (Try It!)

Sierpinski Triangle @ wiki

“没有亲自动手写过编译器的，不足与谈编译器”

考勤 (00%)：非必要不点名

考勤 (00%)：非必要不点名

平时作业 (00%)：每周一次, 每次 1 ~ 2 题

考勤 (00%)： 非必要不点名

平时作业 (00%)： 每周一次, 每次 1 ~ 2 题

课程实验 (75%)： 10 ~ 12 次实验 + 1 次选做实验 (5 分)

考勤 (00%)： 非必要不点名

平时作业 (00%)： 每周一次, 每次 1 ~ 2 题

课程实验 (75%)： 10 ~ 12 次实验 + 1 次选做实验 (5 分)

期末测试 (25%)： 考试周统一安排; 2 小时; 开卷

每周五晚发布作业　　下周五 23 : 55 前自愿提交作业

邀请码: KPXHEQHD

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 下周五 18:00 发布

本科生期末和寒假安排时间表

周次	星期一	星期二	星期三	星期四	星期五	星期六	星期天
寒假	2月6日 寒假	2月7日 寒假	2月8日 寒假	2月9日 寒假	2月10日 寒假	2月11日 寒假	2月12日 寒假
第1周	2月13日 上课	2月14日 上课	2月15日 上课	2月16日 上课	2月17日 上课	2月18日 考试	2月19日 考试
第2周 停课考试	2月20日 考试	2月21日 考试	2月22日 考试	2月23日 考试	2月24日 考试	2月25日 考试	2月26日 考试
第3周	2月27日 上课	2月28日 上课	3月1日 上课	3月2日 上课	3月3日 上课	3月4日	3月5日
第4周	3月6日 上课	3月7日 上课	3月8日 上课	3月9日 上课	3月10日 上课	3月11日	3月12日
第5周	3月13日 上课	3月14日 上课	3月15日 上课	3月16日 上课	3月17日 上课	3月18日 补缓考	3月19日 补缓考

L1：词法分析 下下周三 18:00 发布

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

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

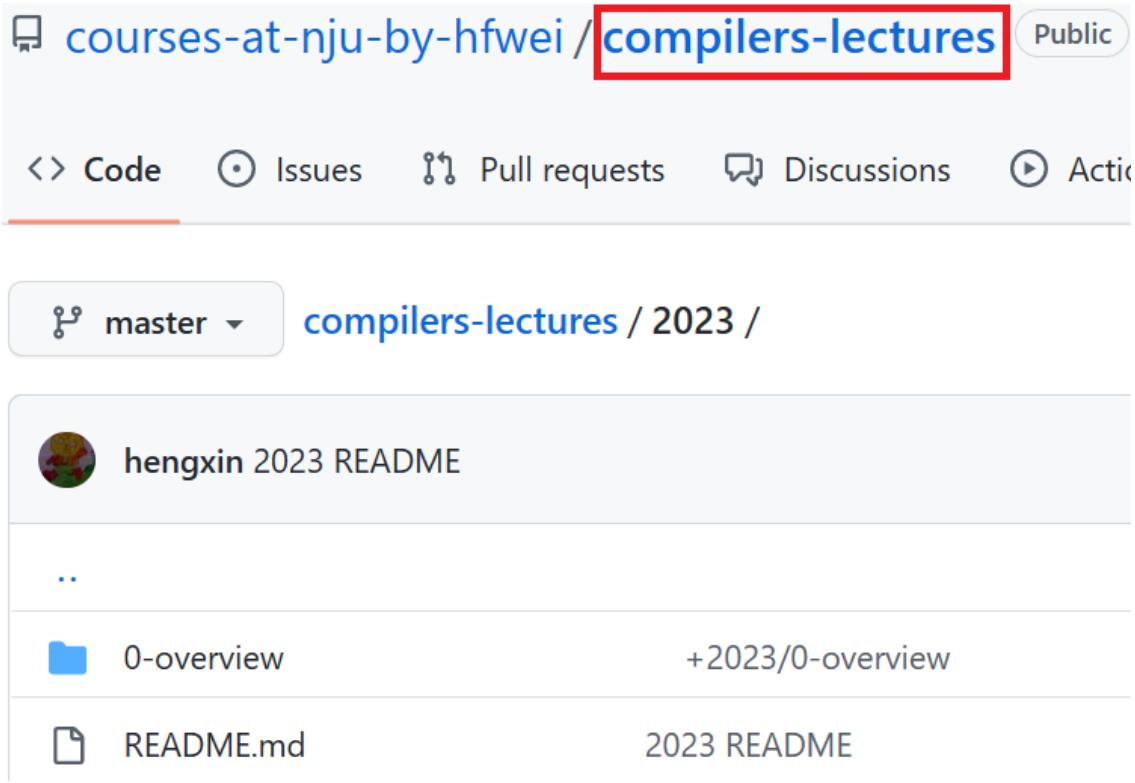
如果你不确定自己的行为是否构成抄袭, 请**事先**咨询助教。

QQ 群号：**455894178**

助教：潘昱光、顾龙、刘学卓

http://47.122.3.40:8081

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/2023

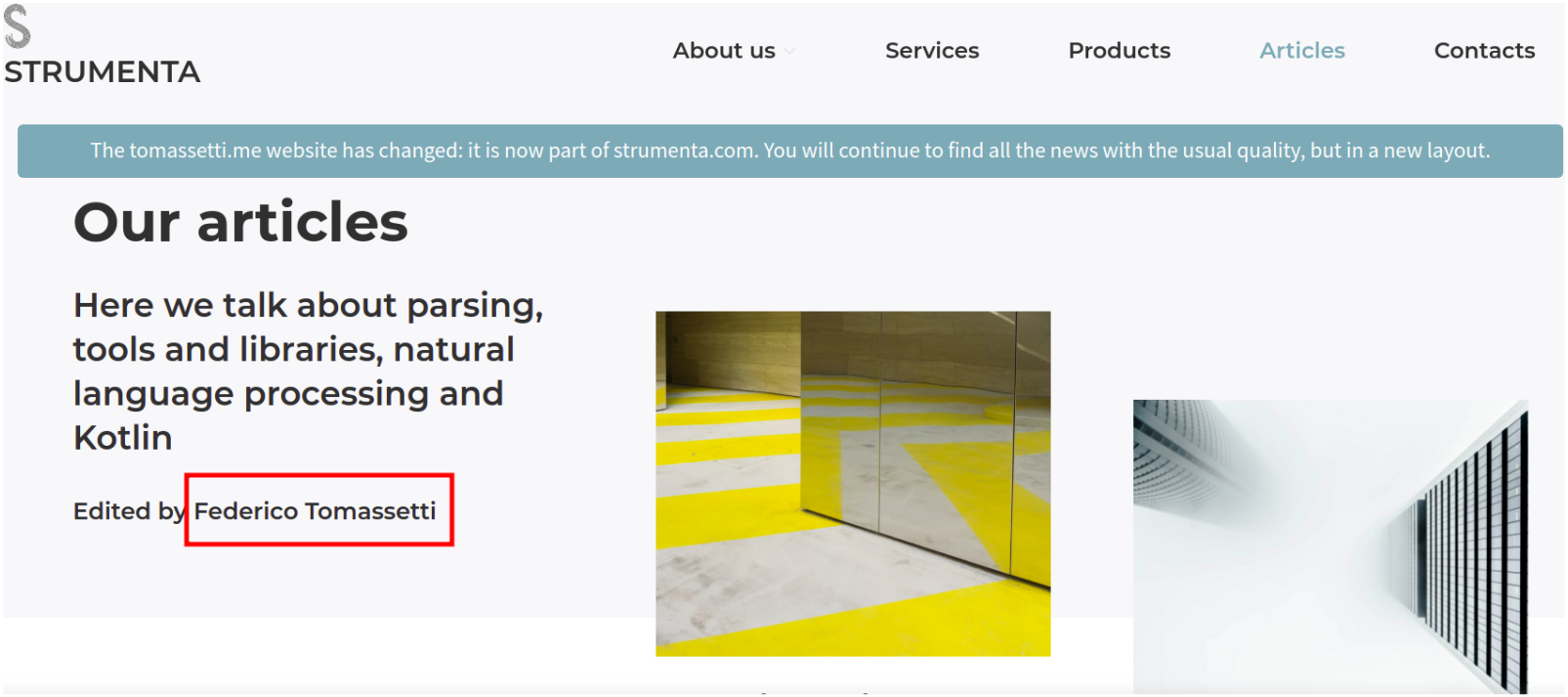
courses-at-nju-by-hfwei / **compilers-papers-we-love** Public

Code | Issues | Pull requests | Actions | Projects | Wiki

master | 1 branch | 0 tags | Go to file | Add fi

hengxin +stringtemplate, +bison-ag

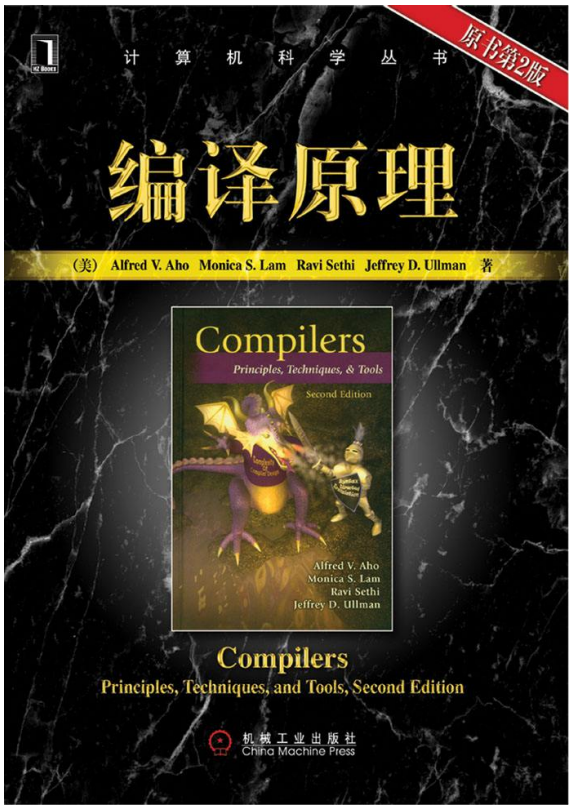
attributed-grammars	+stringtemplate, +bison-ag
cfg	+papers
code-generation	+stringtemplate, +bison-ag
compilers	+papers for parsers
history	+papers for parsers
ir	+papers for parsers
optimization	+papers for parsers
parsing	+papers for parsing
programming-languages	+paper: FunctionalStyle
re-nfa-dfa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

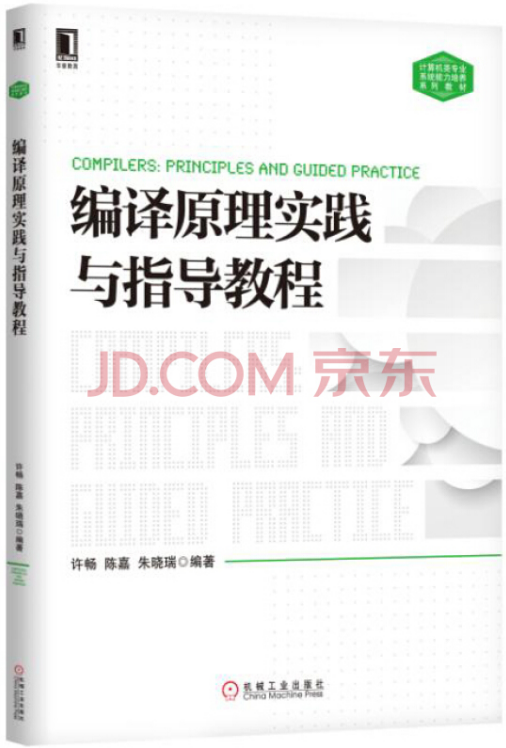


tomassetti.me

专业人士，专业网站，推荐**订阅**

(本书仅供参考)



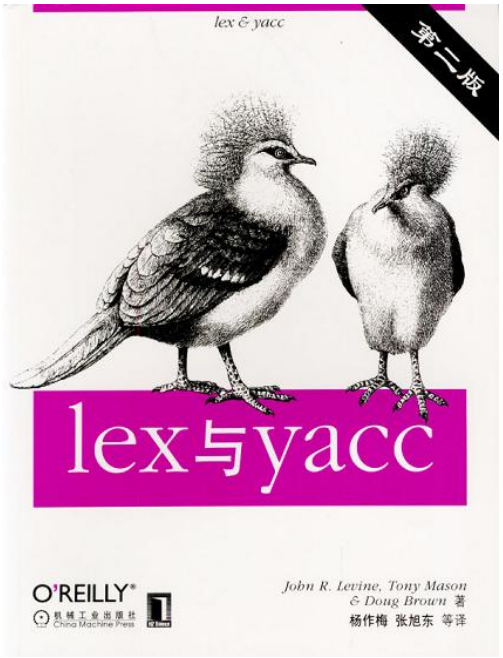


也可使用“**本科教学版**”

https://cs.nju.edu.cn/changxu/2_compiler/index.html

(已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

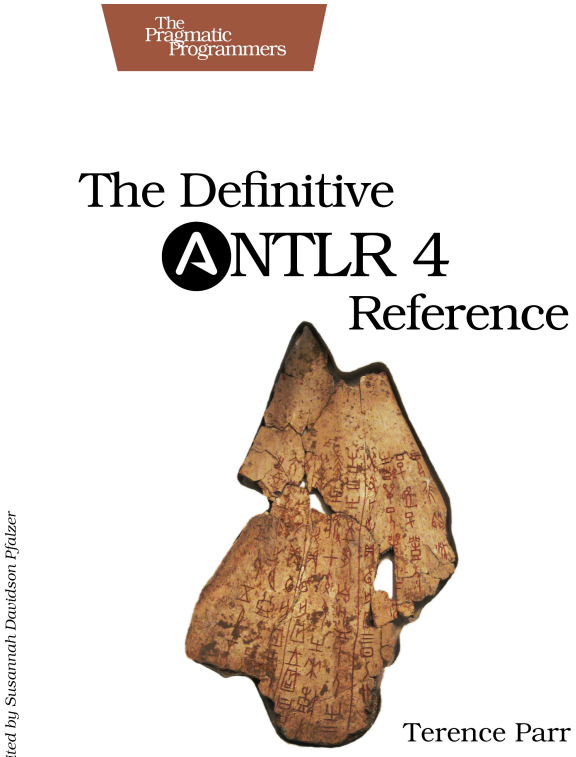
(Since 1988)

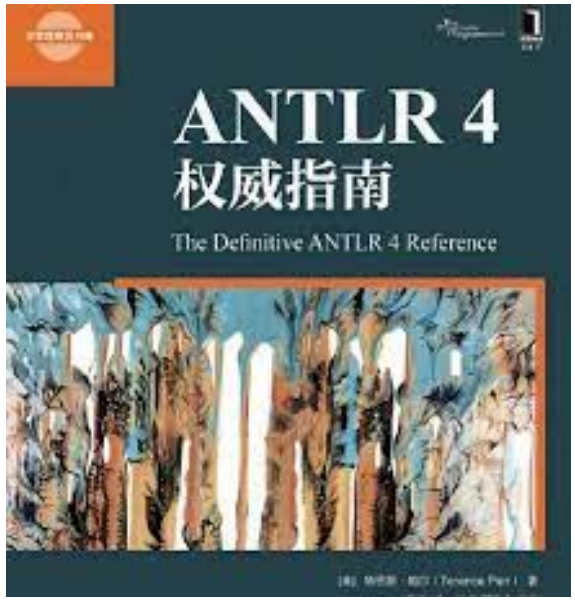
Terence Parr (University of San Francisco)

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

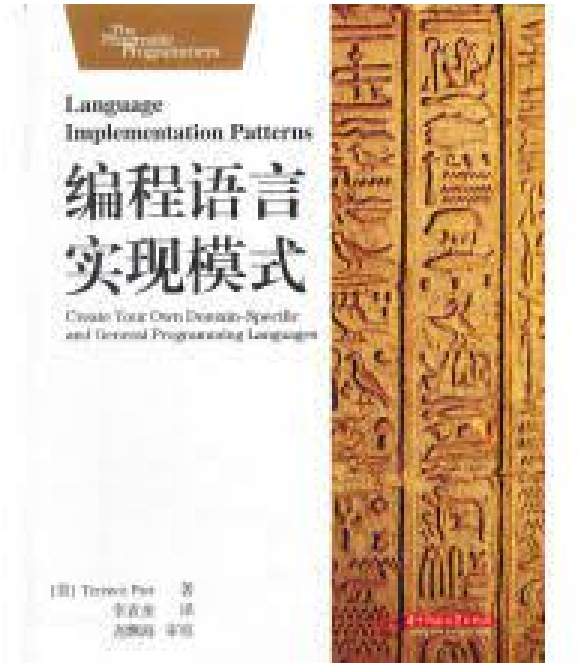
https://www.antlr.org/tools.html (IntelliJ Plugin)

http://lab.antlr.org/ (Online lab)



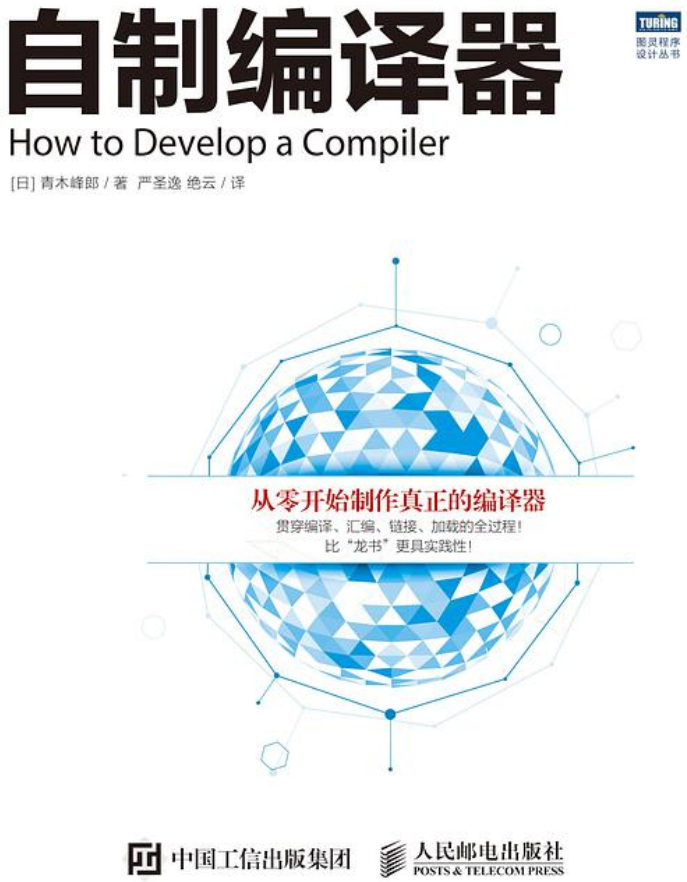


基于 ANTLR 4，是课程实验指导的**重要**参考资料

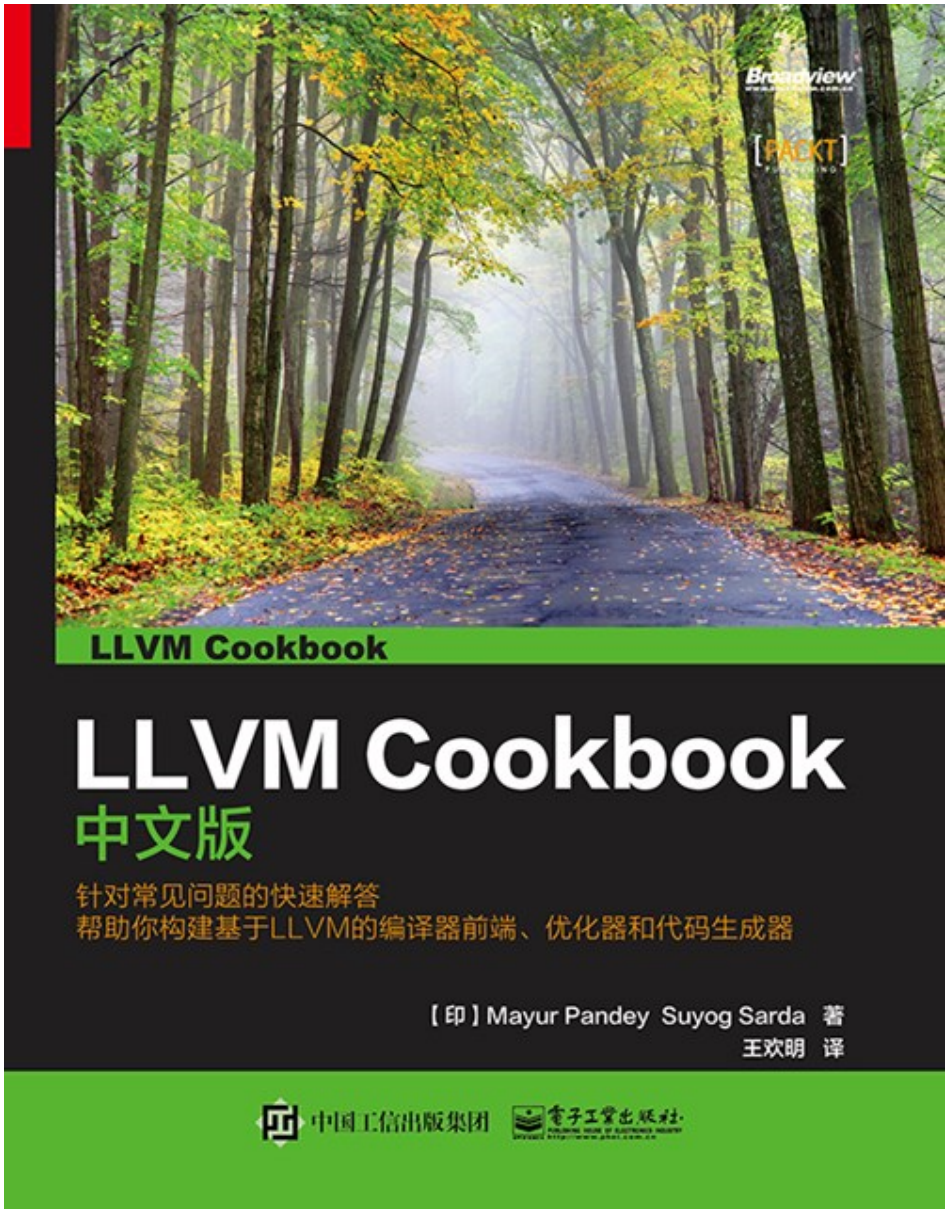


基于 ANTLR 3，与 ANTLR 4 相比，有些过时，

但可以看作理解 ANTLR 4 的基础



“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**



从某次实验开始, 你就会开始接触 LLVM (https://llvm.org/)

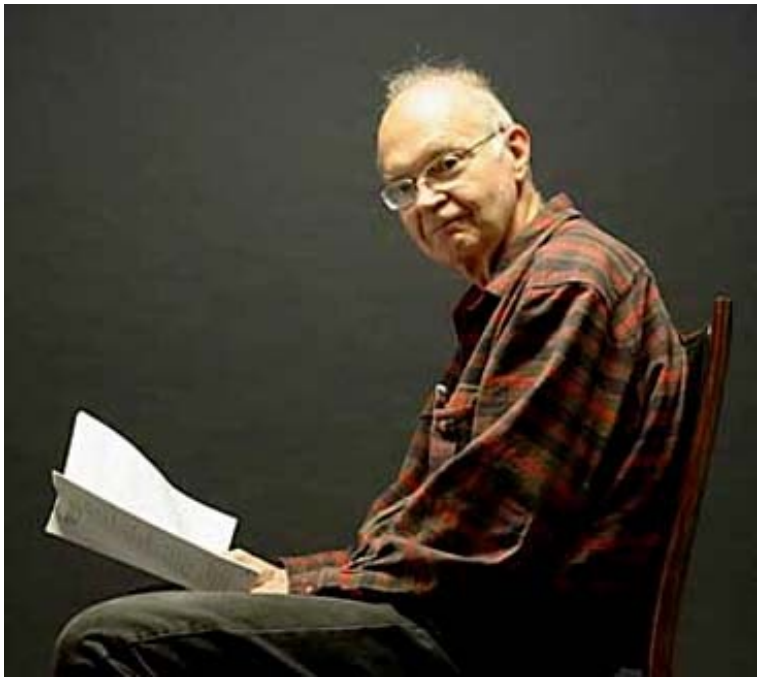
LLVM 简介 @ Bilibili

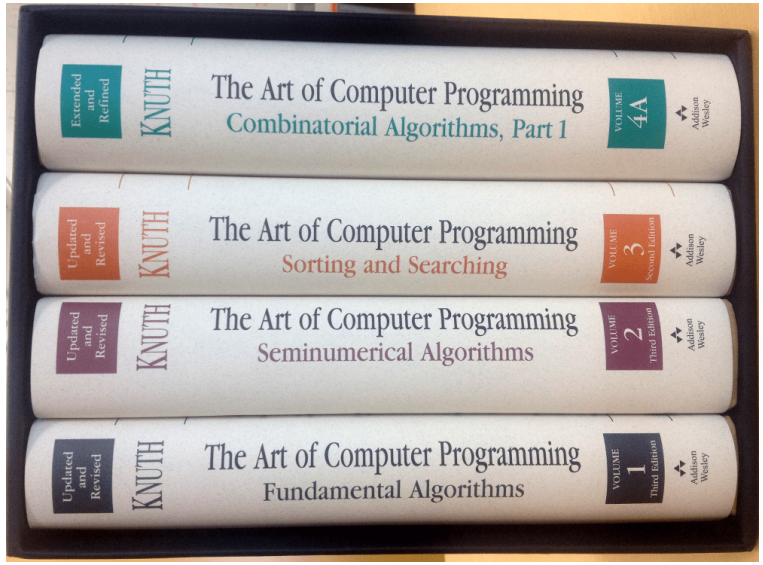
更多参考书, 随课程进展陆续发布

http://47.122.3.40:8081


LET'S GET
STARTED

the “father of the analysis of algorithms”

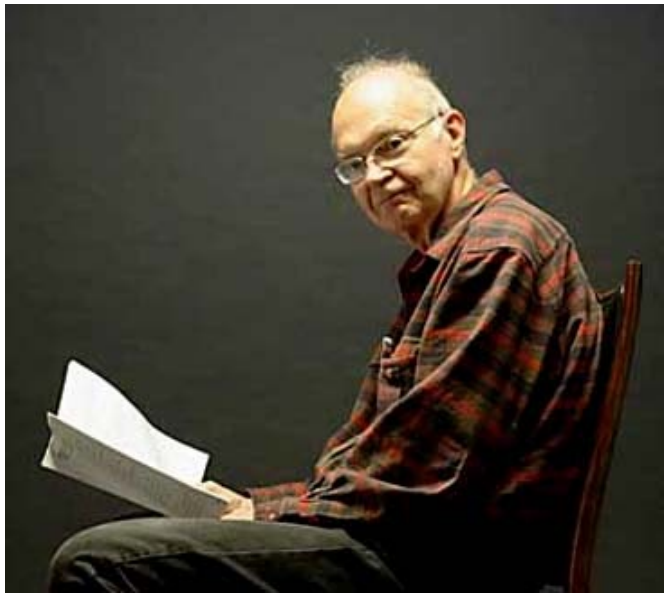




(Since 1962; 计划 7 卷)

Donald E. Knuth (1938 ∼)

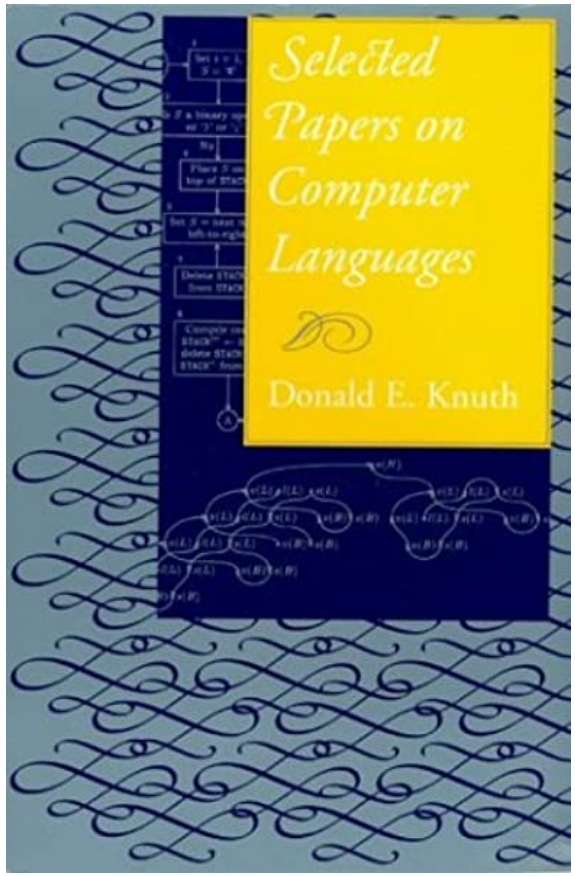
Turing Award, 1974



"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."

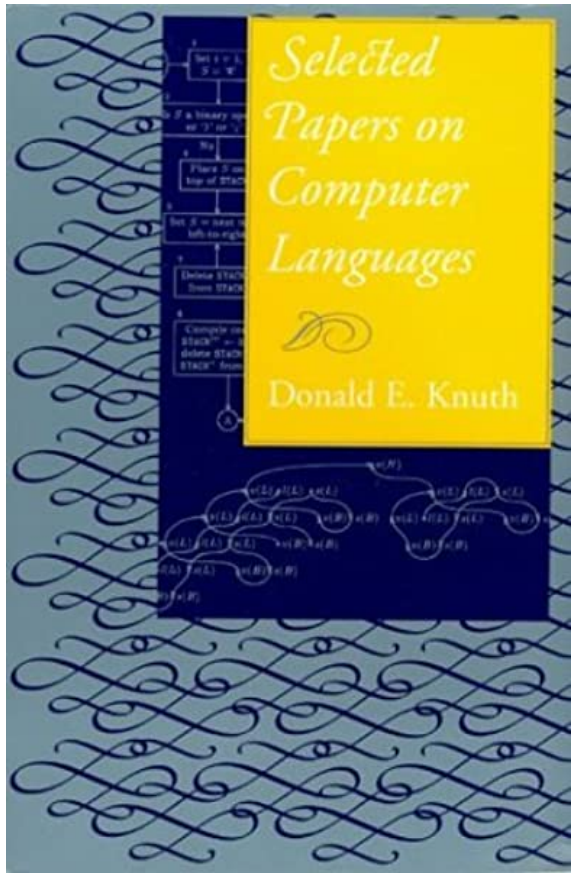
— Turing Award, 1974

"Selected Papers on Computer Languages"



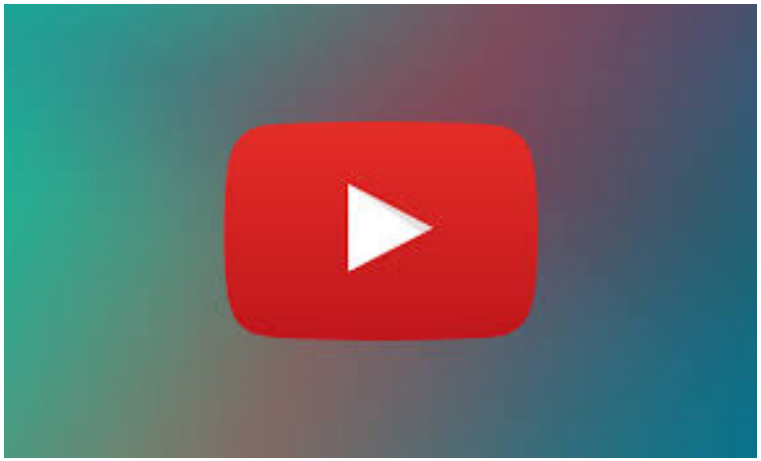
LR Parser (语法) **Attribute Grammar (语义)**

"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 (大四)
to write a compiler for Burroughs"

“高级”语言 $\Longrightarrow$ (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 $\Longrightarrow$ (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 | used by 1.2k projects | PASSED

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 ∼ P9): https://www.bilibili.com/video/BV1EW411u7th

(计算机科学速成课 40 集全 Crash Course Computer Science)

Q：机器语言是如何跑起来的?

作业 (P1 ∼ P9): https://www.bilibili.com/video/BV1EW411u7th

(计算机科学速成课 40 集全 Crash Course Computer Science)

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)

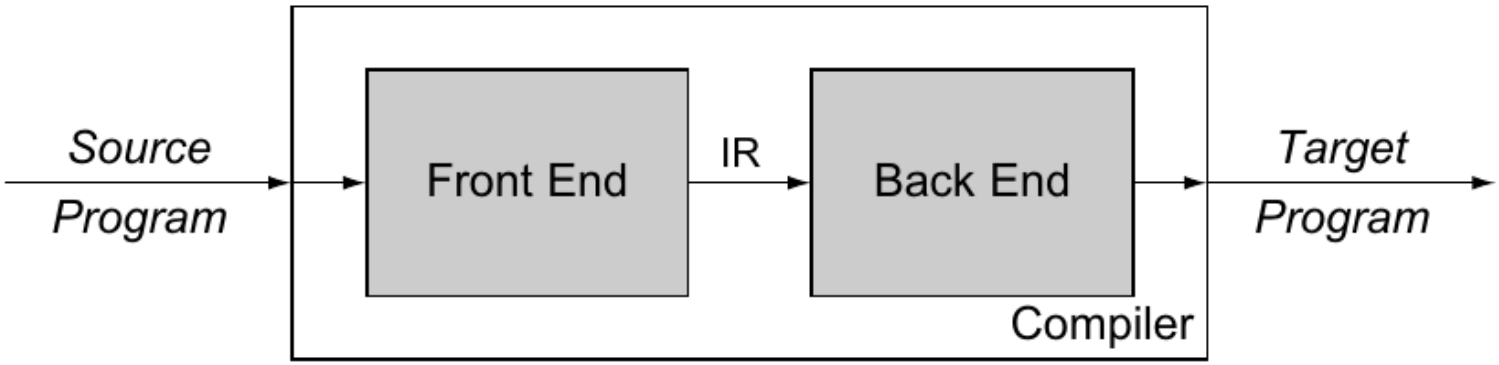
- SQL 引擎 (Structured Query Language)
- TLA$^+$/TLAPS (TPaxos.tla)
- (Java) 字节码解释器
- C/C++ 语言编译器

- 排版工具 (LaTeX)
- 绘图工具 (TikZ, Dot/Graphviz)
- L-System (Cantor Set)


Source
Program
Compiler
Target
Program

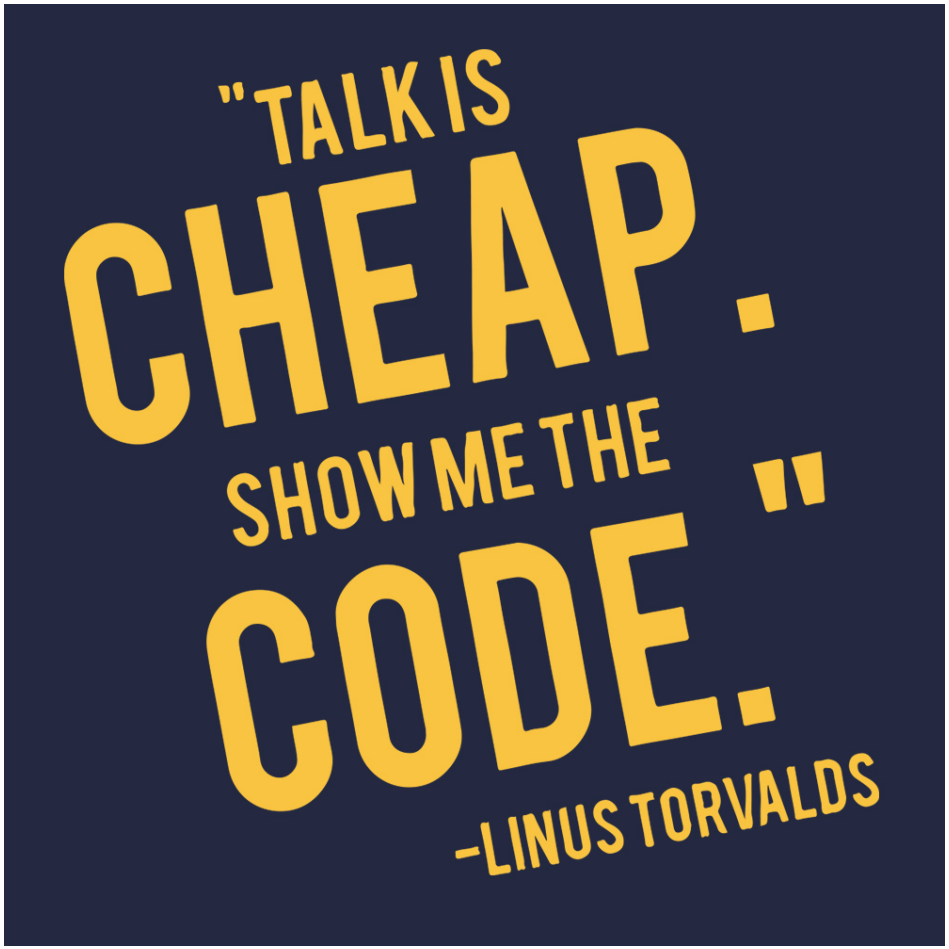

KEEP
CALM
AND
OPEN THIS
BOX

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

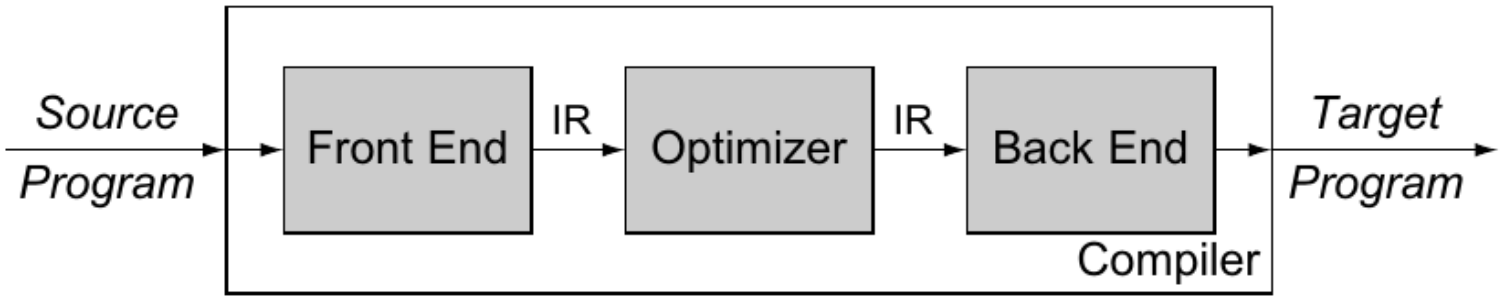

"TALK IS
CHEAP.
SHOW ME THE
CODE."
-LINUS TORVALDS

Clang: a C language family frontend for LLVM

The Clang project provides a language front-end and tooling infrastructure for languages in the C language family (C, C++, Objective C/C++, OpenCL, CUDA, and RenderScript) for the LLVM project. Both a GCC-compatible compiler driver (`clang`) and an MSVC-compatible compiler driver (`clang-cl.exe`) are provided. You can get and build the source today.

https://clang.llvm.org/

(`factorial.c`)



机器无关的中间表示优化

Clang: a C language family frontend for LLVM

The Clang project provides a language front-end and tooling infrastructure for languages in the C language family (C, C++, Objective C/C++, OpenCL, CUDA, and RenderScript) for the LLVM project. Both a GCC-compatible compiler driver (`clang`) and an MSVC-compatible compiler driver (`clang-cl.exe`) are provided. You can get and build the source today.

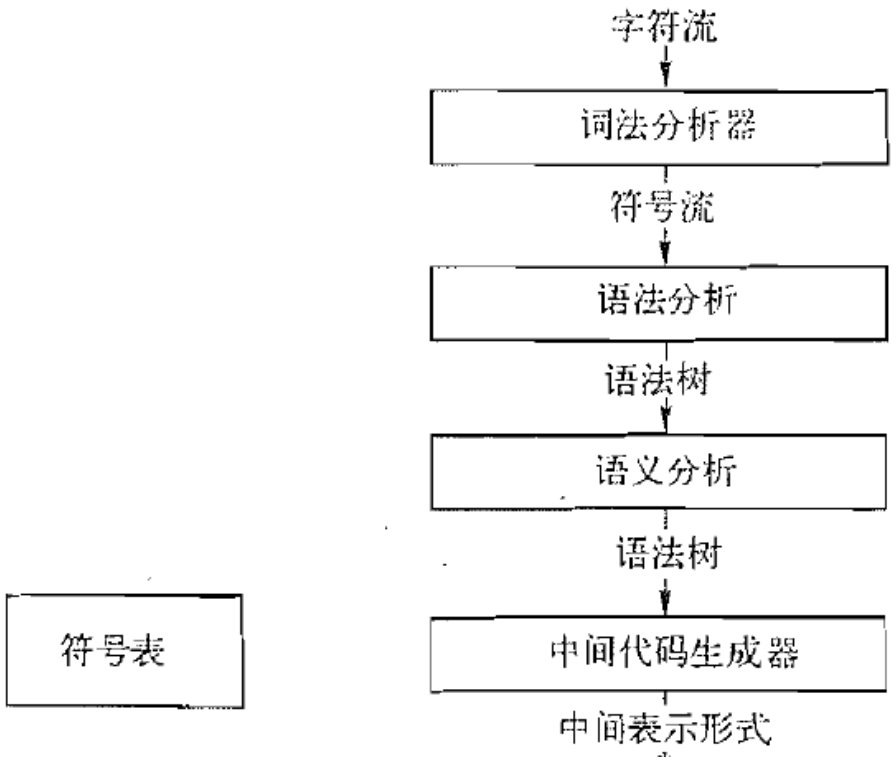
https://clang.llvm.org/

(factorial.c)

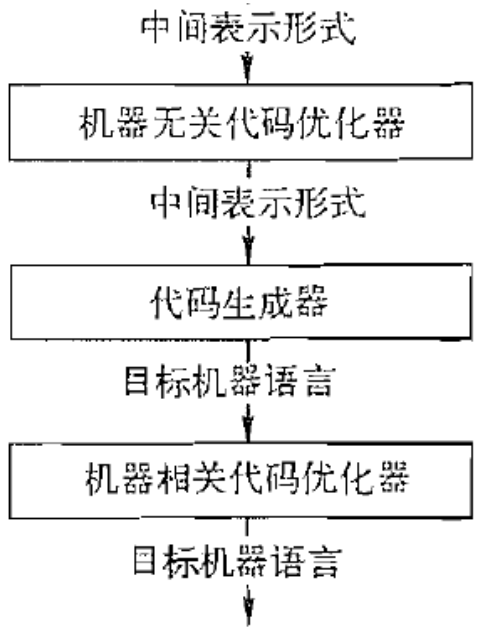
在设计实际生产环境中的编译器时，**优化**通常占用了大多数时间

Maple IR 及其各种优化技术

编译器前端：分析阶段



编译器后端：综合阶段



Clang: a C language family frontend for LLVM

The Clang project provides a language front-end and tooling infrastructure for languages in the C language family (C, C++, Objective C/C++, OpenCL, CUDA, and RenderScript) for the LLVM project. Both a GCC-compatible compiler driver (`clang`) and an MSVC-compatible compiler driver (`clang-cl.exe`) are provided. You can get and build the source today.

https://clang.llvm.org/

(naming.c)

```
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 : $\langle$token-class, attribute-value$\rangle$

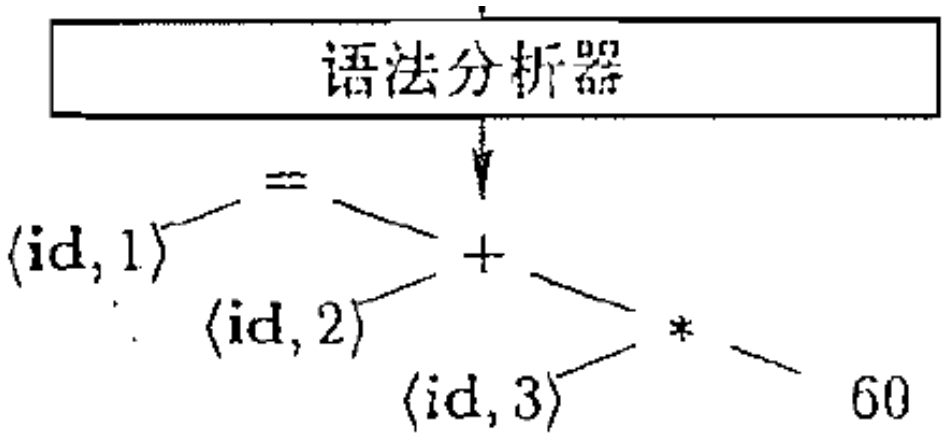
```
position = initial + rate * 60
```

$\langle$**id**, 1$\rangle$ $\langle$ws$\rangle$ $\langle$**assign**$\rangle$ $\langle$ws$\rangle$ $\langle$**id**, 2$\rangle$ $\langle$ws$\rangle$
$\langle+\rangle$ $\langle$ws$\rangle$ $\langle$**id**, 3$\rangle$ $\langle$ws$\rangle$ $\langle*\rangle$ $\langle$ws$\rangle$ $\langle$**num**, 4$\rangle$

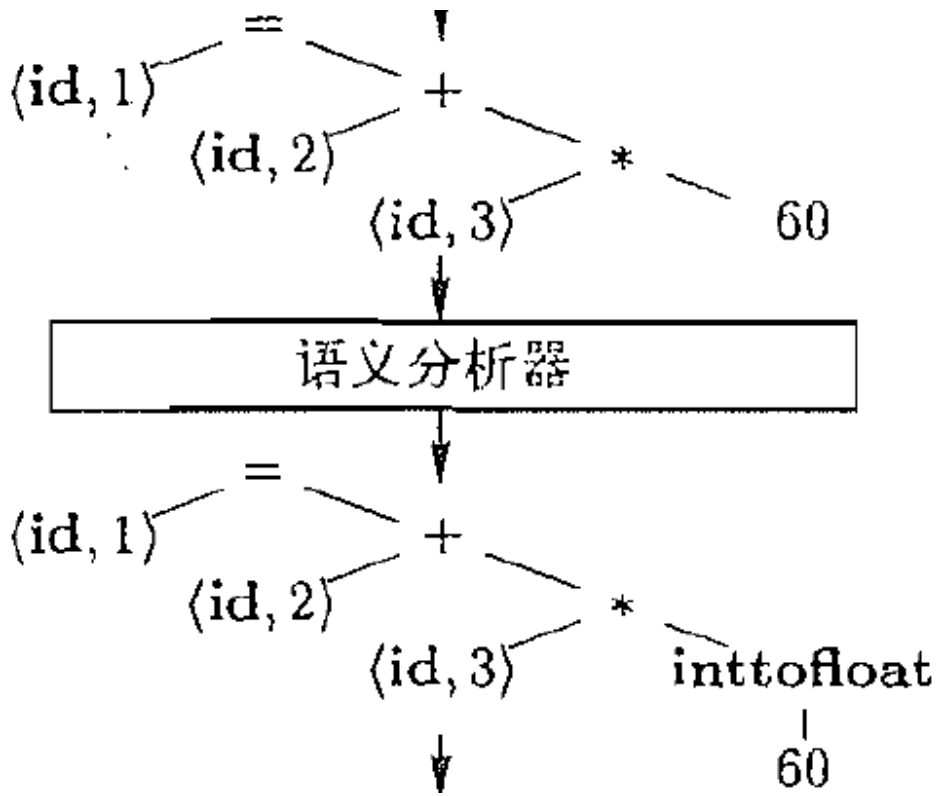
(此处, 1, 2, 3, 4 是指向**符号表**的指针)

语法分析器 (Parser): 构建**词法单元**之间的语法结构, 生成**语法树**

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

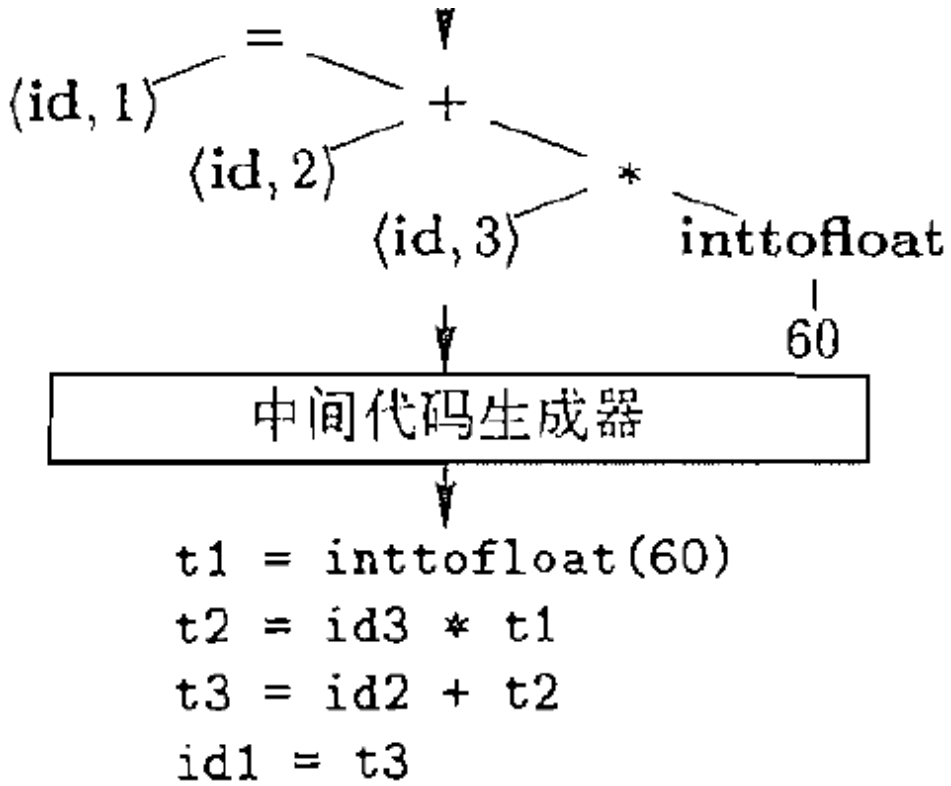


语义分析器: 语义检查, 如**类型检查、“先声明后使用”约束检查**



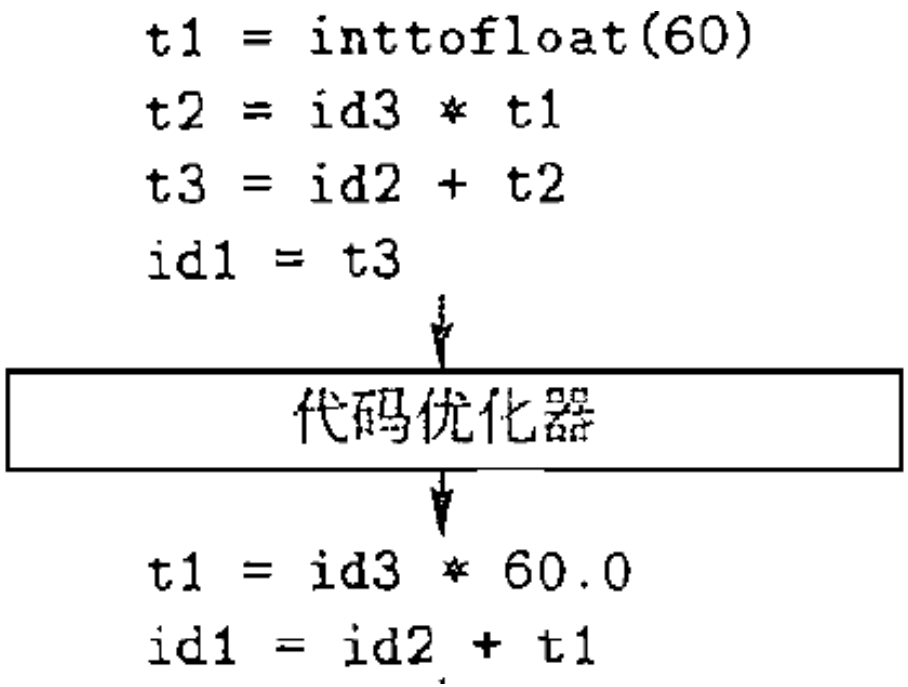
通过语法树上的遍历来完成

中间代码生成器: 生成中间代码, 如 **“三地址代码”**



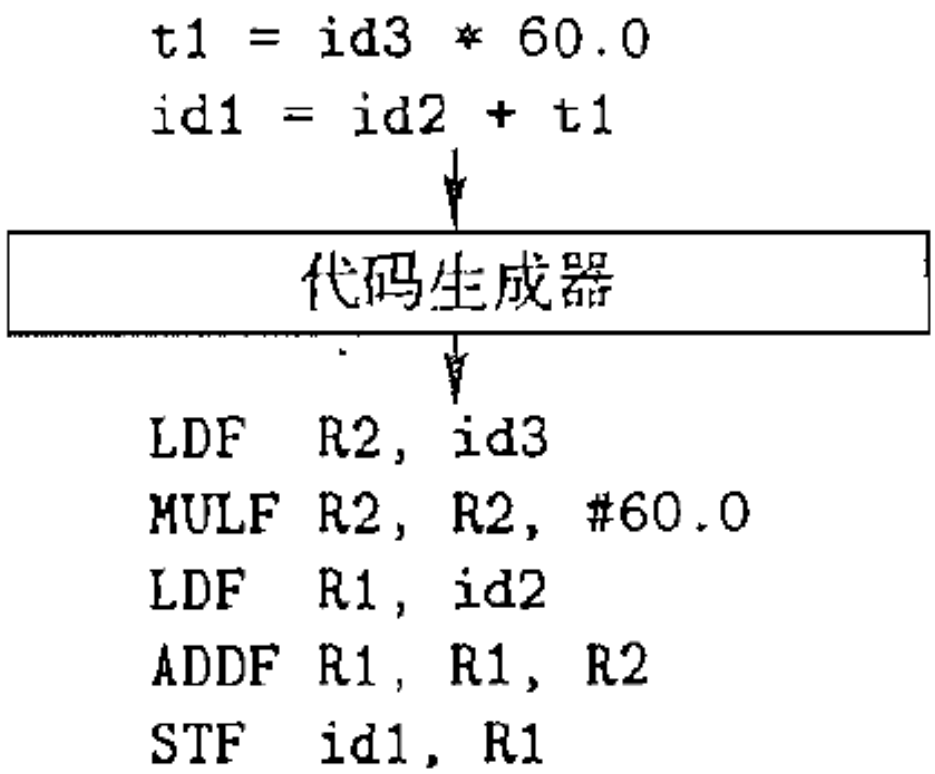
中间代码类似目标代码, 但不含有机器相关信息 (如寄存器、指令格式)

中间代码优化器



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:

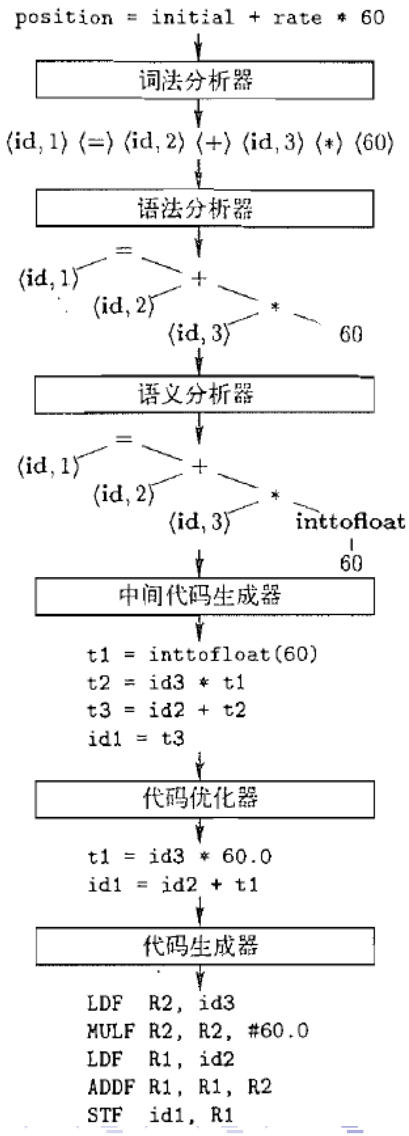
类型、寄存器、内存地址、行号

函数名:

参数个数、参数类型、返回值类型

1	position	···
2	initial	···
3	rate	···

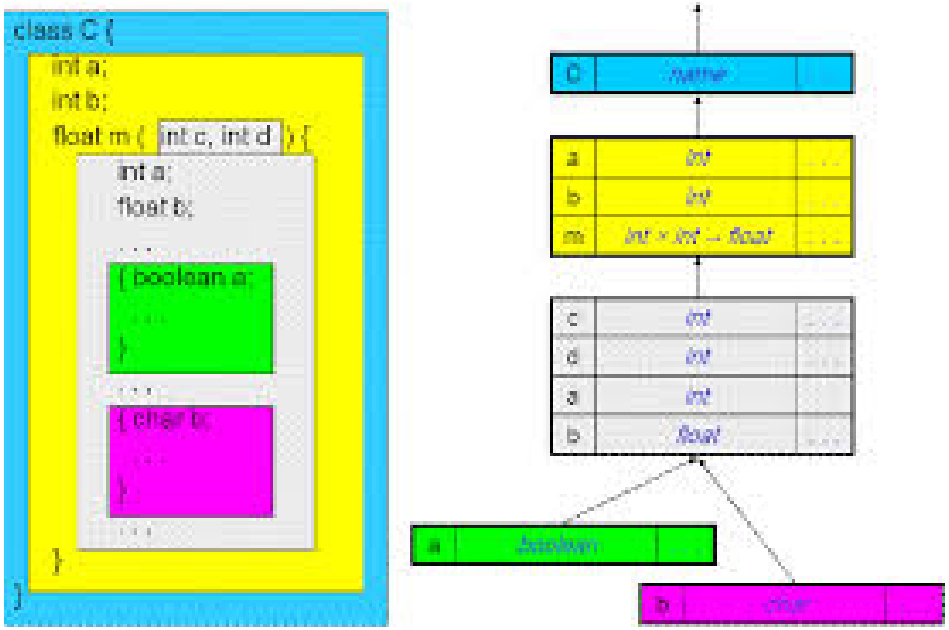
符号表



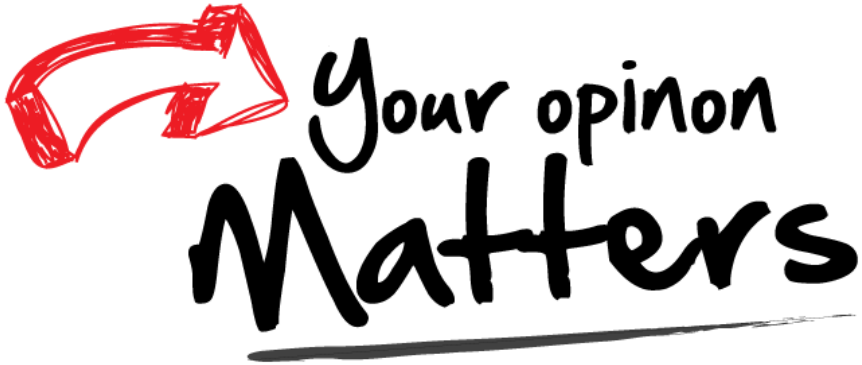
public class ST<Key extends Comparable<Key>, Value>		
	ST()	*create an empty symbol table*
void	put(Key key, Value val)	*associate* `val` *with* `key`
Value	get(Key key)	*value associated with* `key`
void	remove(Key key)	*remove* `key` *(and its associated value)*
boolean	contains(Key key)	*is there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

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




Thank
You!



Office 926

hfwei@nju.edu.cn