

编译原理概述

魏恒峰

hfwei@nju.edu.cn

2022 年 10 月 31 日 (周一)



7 周 = 14 次课 < 8 周 = 16 次课



作业 (0 分): ≈ 7 次作业, 每周 1 ~ 2 题

作业 (0 分): ≈ 7 次作业, 每周 1 ~ 2 题

实验 (60 分): ≈ 8 次必做实验 + 1 次选做实验 (5 分)

作业 (0 分): ≈ 7 次作业, 每周 1 ~ 2 题

实验 (60 分): ≈ 8 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**

作业 (0 分): ≈ 7 次作业, 每周 1 ~ 2 题

实验 (60 分): ≈ 8 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 (5 分): 报告 + 录屏方式, 学习更现代的编译原理与技术



实验：当次实验计零分

附加项：不计分

期末测试：交与教务处

每周三晚上发布作业

下周三 23 : 55 前[提交作业](#)



邀请码: 5JP57Q5H

发布调查问卷

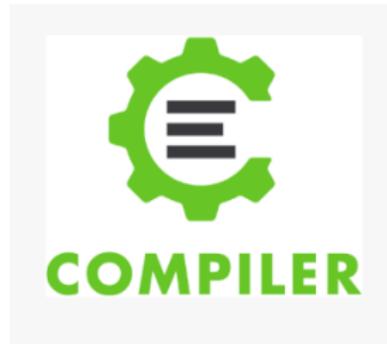
发布阅读材料

QQ 群号: **755783220**



助教: 夏宇、潘煜光、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

$$45 = \textcolor{red}{0} + 5 + 15 + 15 + 10 + \textcolor{red}{5}$$

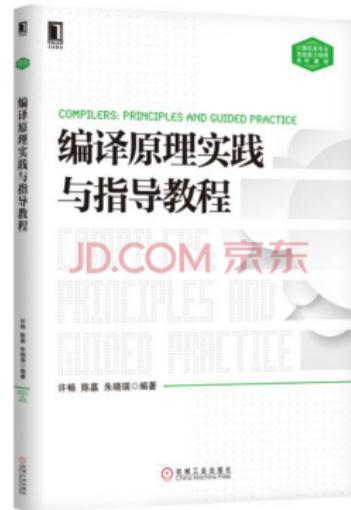
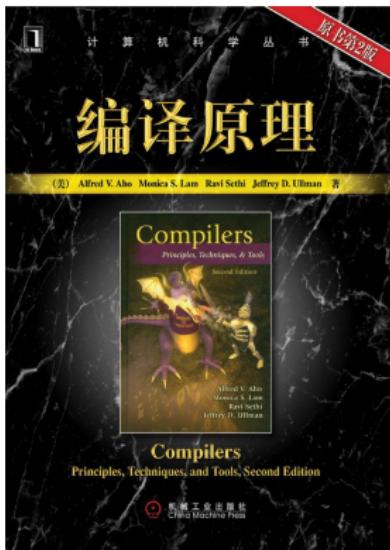
实验列表

- L0: 环境配置 (不占分, 11月10日-11月15日)
- L1: 词法分析 (5分, 11月12日-11月22日)
- L2: 语法分析 (15分, 11月24日-12月6号)
- L3: 语义分析 (15分, 12月8号-12月27号)
- L4: 中间代码生成 (10分, 12月17号-1月3号)
- L5: 目标代码生成 (5分, 待定)

L0: 环境配置 今晚 18:00 发布

[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两种工具



Terence Parr

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

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

<http://lab.antlr.org/>

<http://lab.antlr.org/>

arithmetic: number1.txt, simple2.txt

<http://lab.antlr.org/>

arithmetic: number1.txt, simple2.txt

fol: example1.txt

<http://lab.antlr.org/>

arithmetic: number1.txt, simple2.txt

fol: example1.txt

guitartab: example1.txt

<http://lab.antlr.org/>

arithmetic: number1.txt, simple2.txt

fol: example1.txt

guitartab: example1.txt

xyz: example1.txt

<http://lab.antlr.org/>

arithmetic: number1.txt, simple2.txt

fol: example1.txt

guitartab: example1.txt

xyz: example1.txt

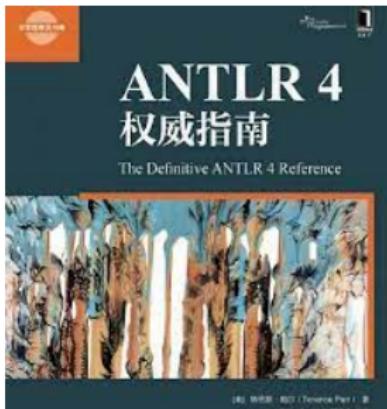
C: add.c

The Definitive
ANTLR 4
Reference



Terence Parr

The
Pragmatic
Programmers



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

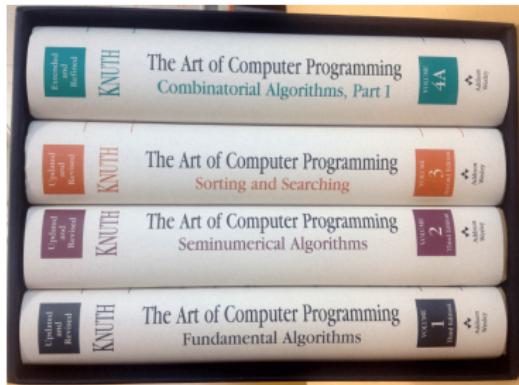
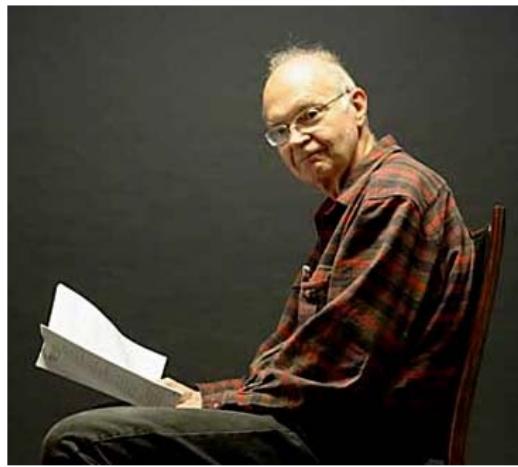
[译] Terence Parr 著
李吉东 译
高阳海 审校



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



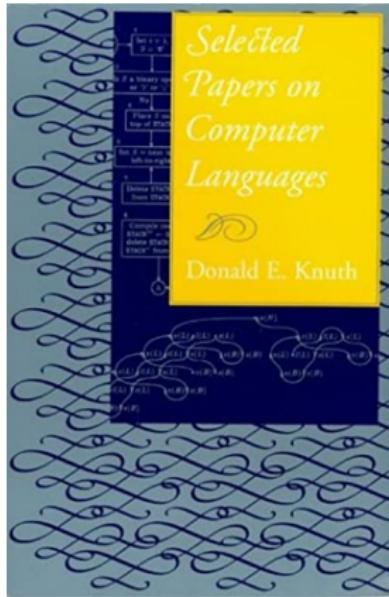
the “father of the analysis of algorithms”



Donald E. Knuth (1938 ~)

Turing Award, 1974

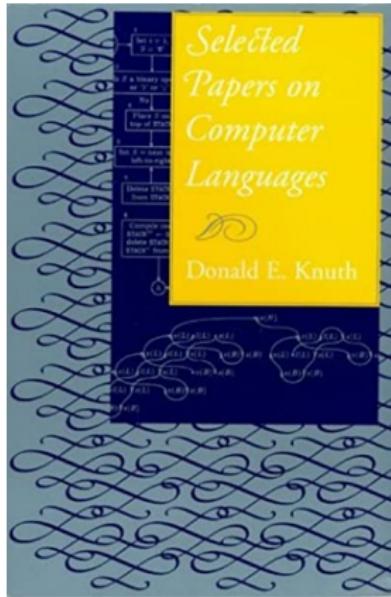
“Selected Papers on Computer Languages”



LR Parser (语法)

Attribute Grammar (语义)

“Selected Papers on Computer Languages”



LR Parser (语法)

Attribute Grammar (语义)

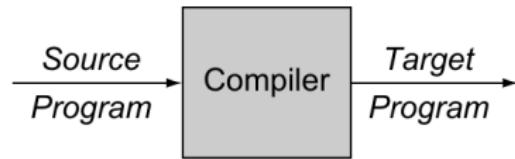
ALGOL 58 Compiler



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

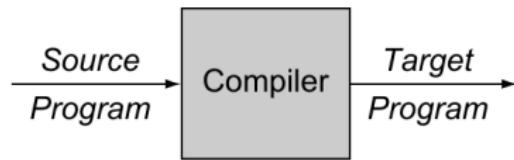
“高级”语言 \Rightarrow (通常) “低级”语言 (如, 汇编语言)

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 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 ~ 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

[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
```

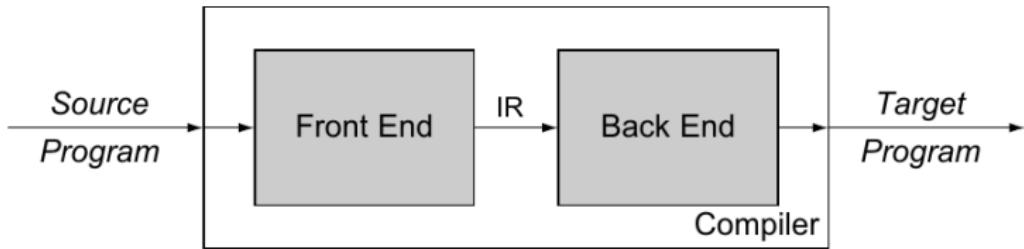
The language's design equally favors aesthetics, flexibility and ease of use.

[alda.io](#)

两个月的“编译器设计原理”之旅

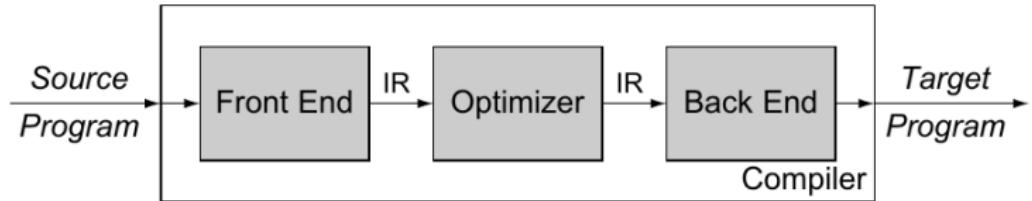


IR: Intermediate Representation (中间表示)



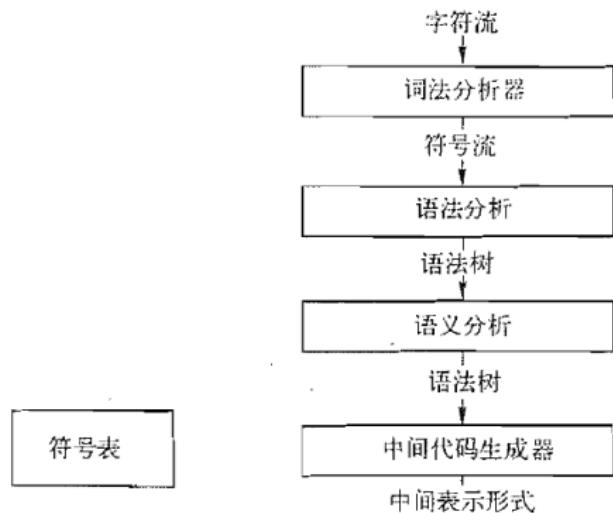
前端（分析阶段）: 分析源语言程序, 收集所有必要的信息

后端（综合阶段）: 利用收集到的信息, 生成目标语言程序



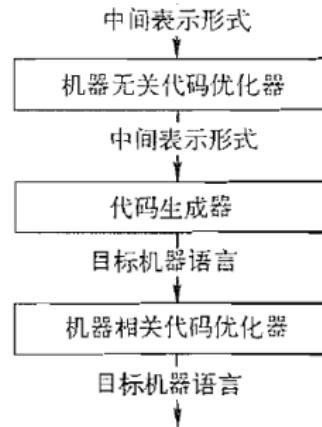
机器无关的中间表示优化

编译器前端：分析阶段



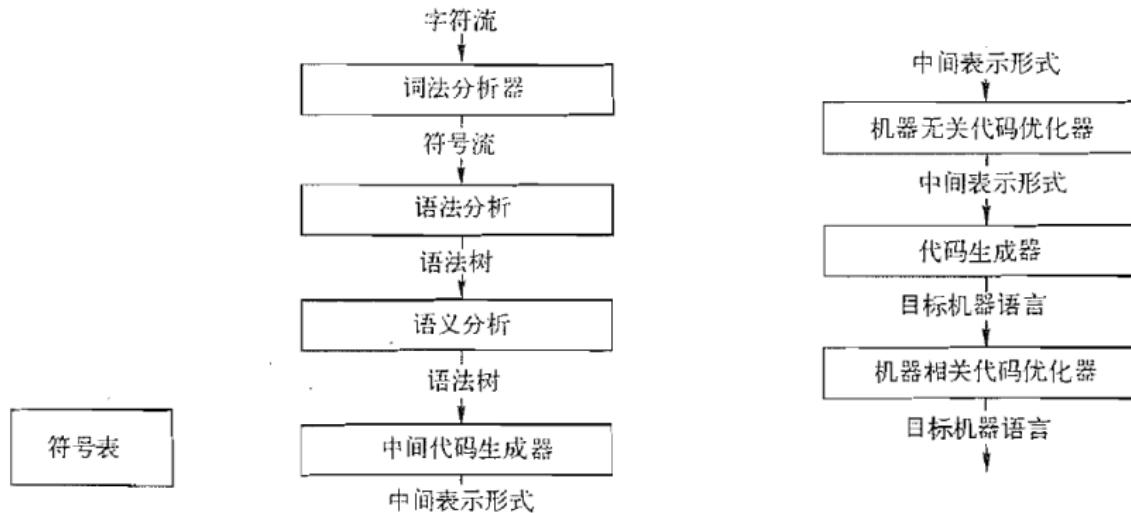
前四次必做实验

编译器后端：综合阶段



第五次选做实验

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



查看源码



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>

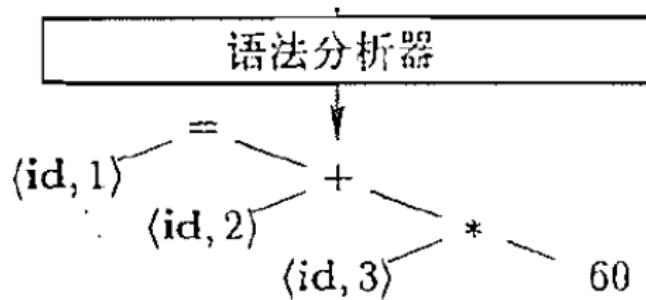
```
position = initial + rate * 60
```

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

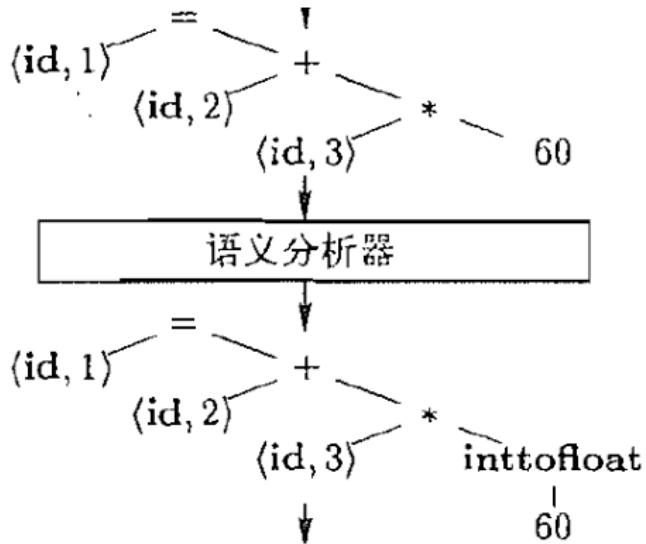
(此处, 1, 2, 3, 4 是指向**符号表**的指针)

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
```

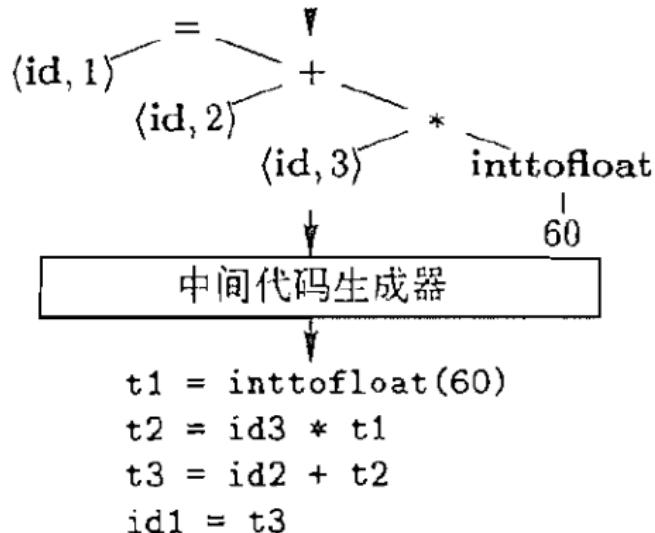


语义分析器：语义检查，如类型检查、“先声明后使用”约束检查



通过语法树上的遍历来完成

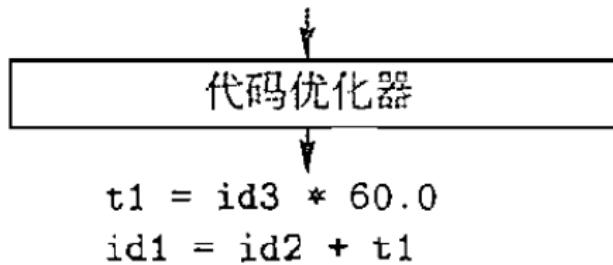
中间代码生成器：生成中间代码，如“三地址代码”



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）

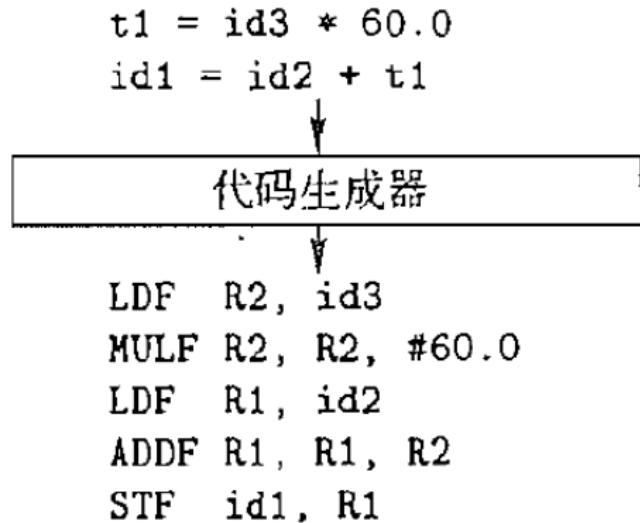
中间代码优化器

```
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:

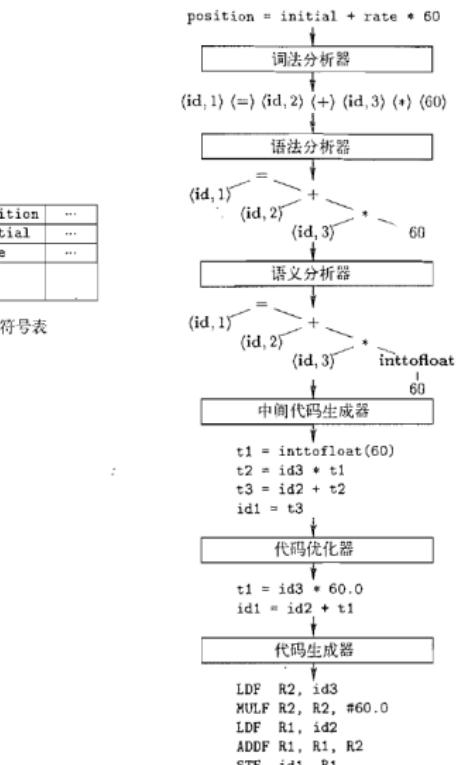
类型、寄存器、内存地址、行号

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

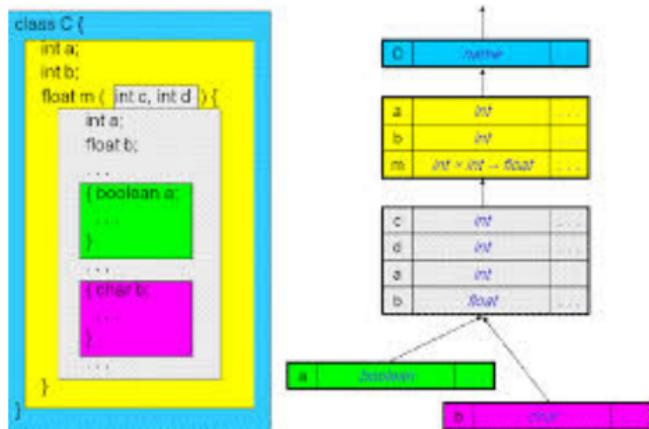


```
public class ST<Key extends Comparable<Key>, Value>
```

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

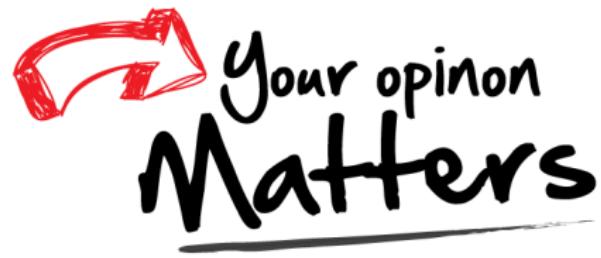
Hello.g4

SimpleExpr.g4

Expr.g4

grammars-v4 @ github

Thank You!



Office 926

hfwei@nju.edu.cn