编译原理概述

魏恒峰

hfwei@nju.edu.cn

2020 年 11 月 1 日

8 周 = 7.5 周 (授课) + 0.5 周 (期末测试)

期末测试 (40 分): 2020 年 12 月 31 日; 时长 2 小时; 闭卷

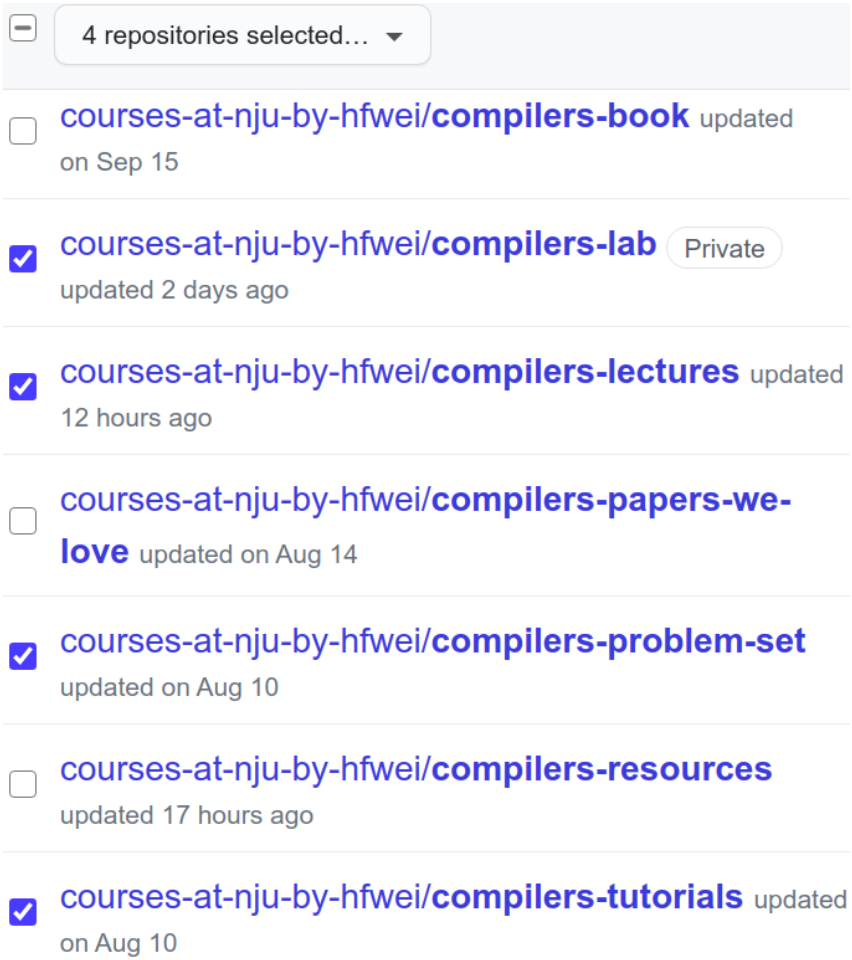
作业 (15 分): 7 次必做作业 + 1 次选做报告 (+3 分)

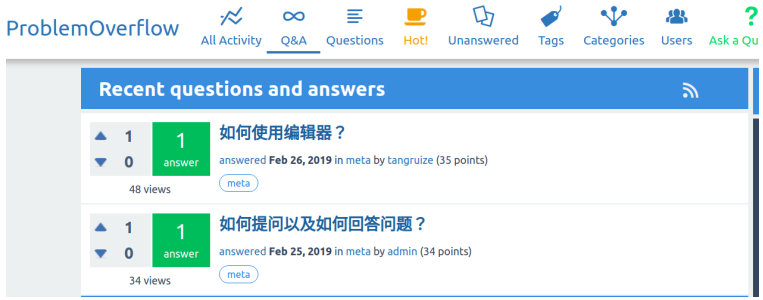
实验 (45 分): 4 次必做实验 + 1 次选做实验 (+5 分)

$$45 = 0 + 5 + 15 + 15 + 10 + 5$$

http://n6.disalg.cn/labs/

一次：当次作业计零分　　两次：所有作业计零分　　三次：总评零分



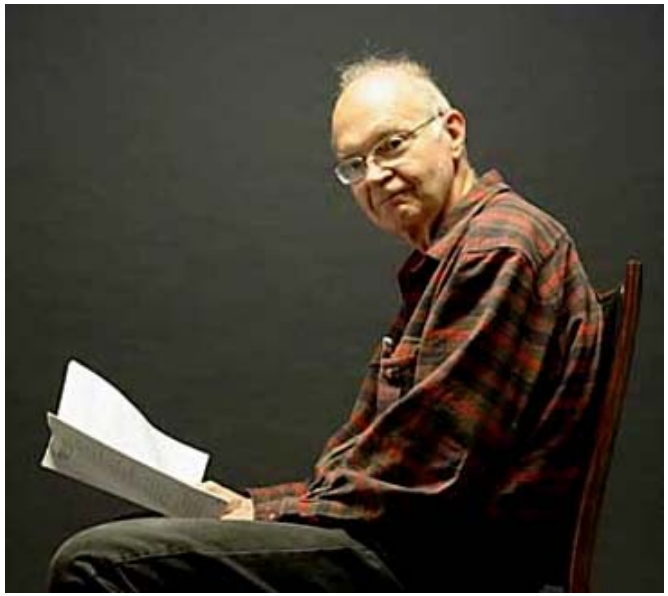


http://n6.disalg.cn/

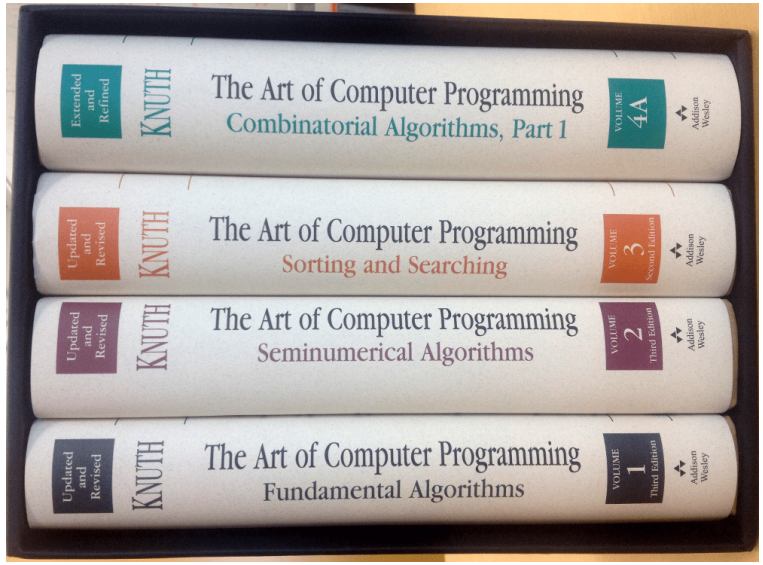
https://github.com/orgs/courses-at-nju-by-hfwei/teams/compilers-course-at-nju-software/repositories

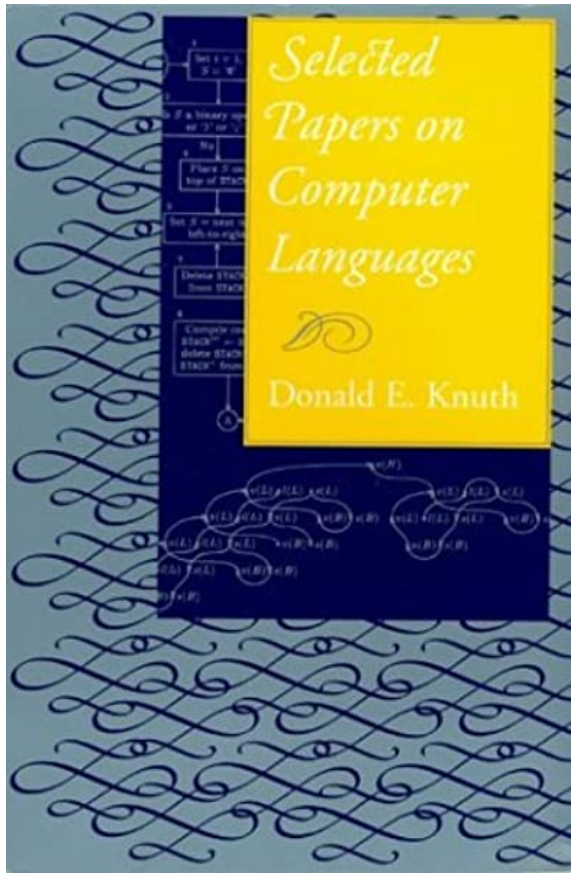

welcome
again

the “father of the analysis of algorithms”

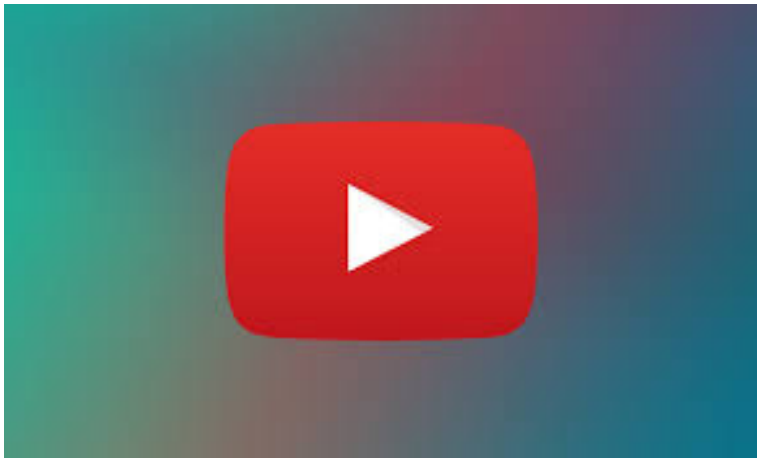


Donald E. Knuth (1938 $\sim$)





ALGOL 58 Compiler **LR Parser** **Attribute Grammar**



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

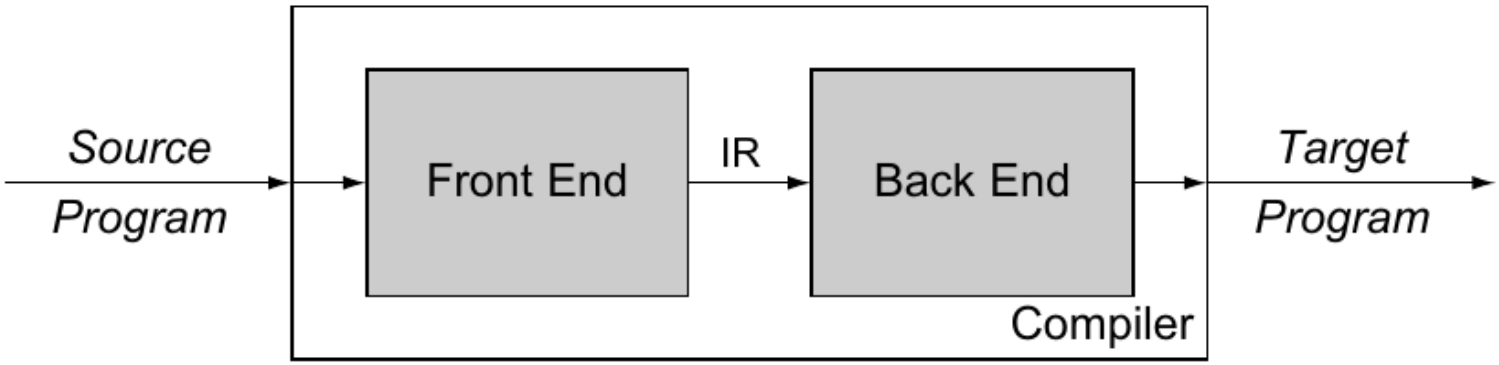
“高级”语言 $\Longrightarrow$ (通常)“低级”语言 (如, 汇编语言)

汇编语言经过**汇编器**生成机器语言

Q : 机器语言是如何跑起来的?

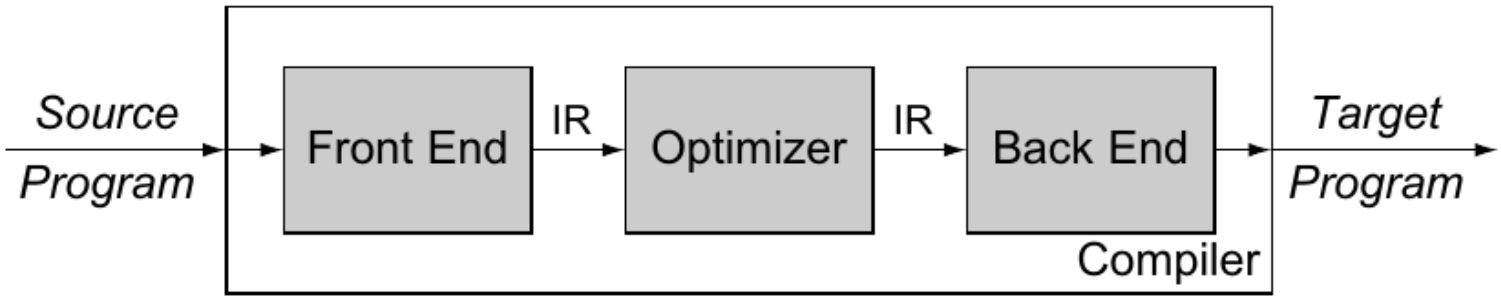
两个月的“编译器设计原理”之旅

IR: Intermediate Representation (中间表示)



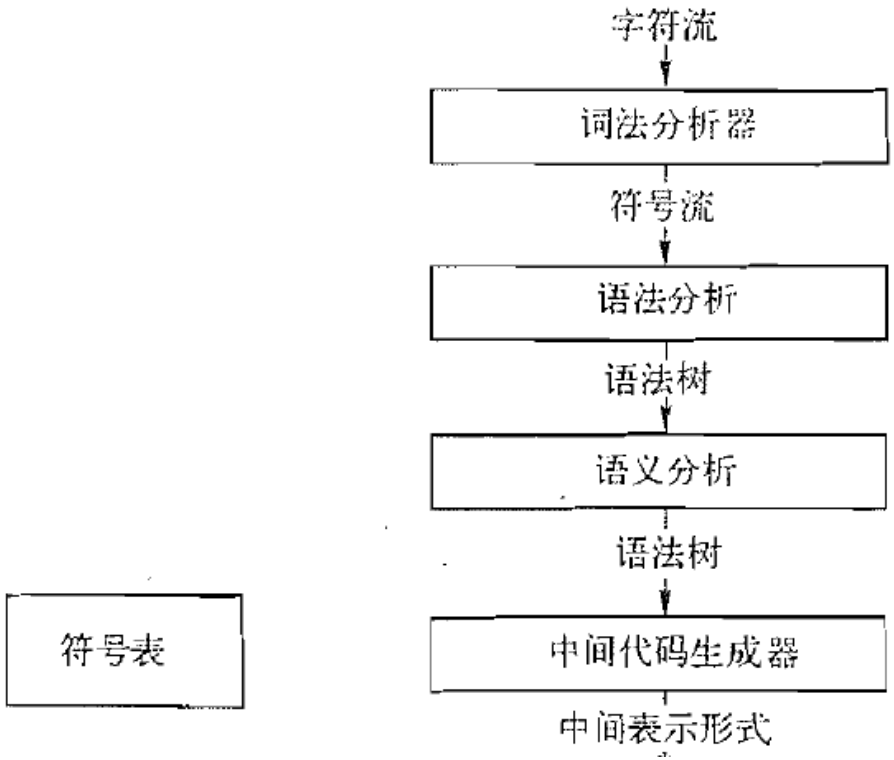
前端 **(分析阶段)**: 分析源语言程序, 收集所有必要的信息

后端 **(综合阶段)**: 利用收集到的信息, 生成目标语言程序



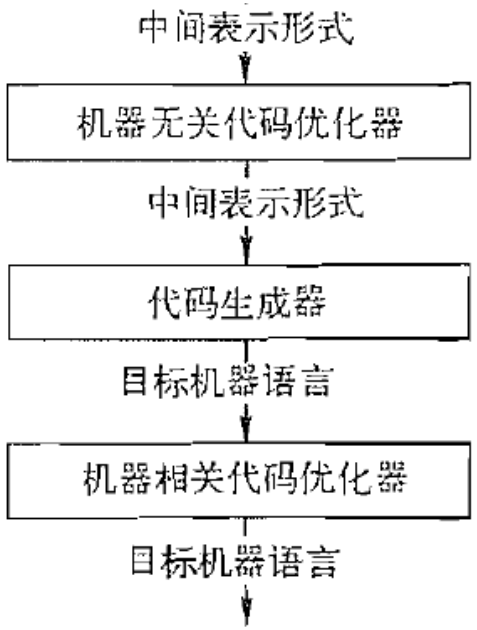
机器无关的中间表示优化

编译器前端：分析阶段



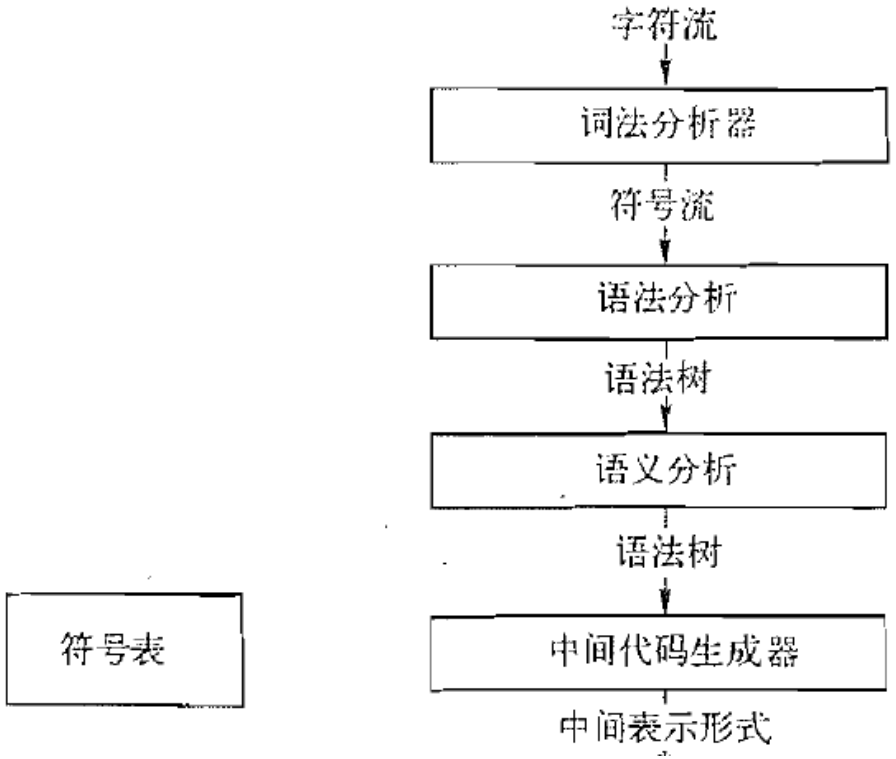
前四次必做实验

编译器后端：综合阶段



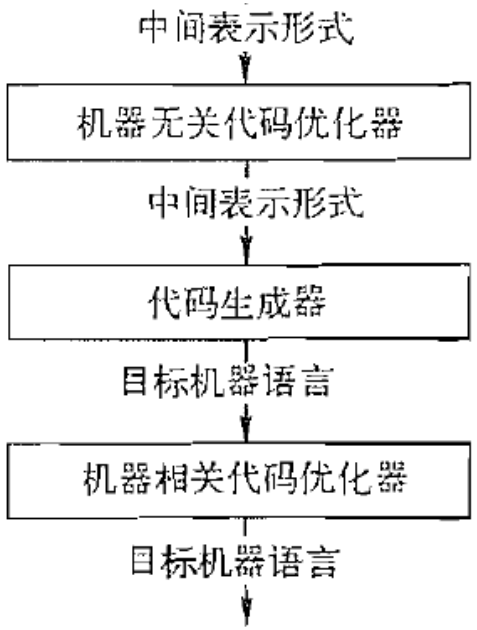
第五次选做实验

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

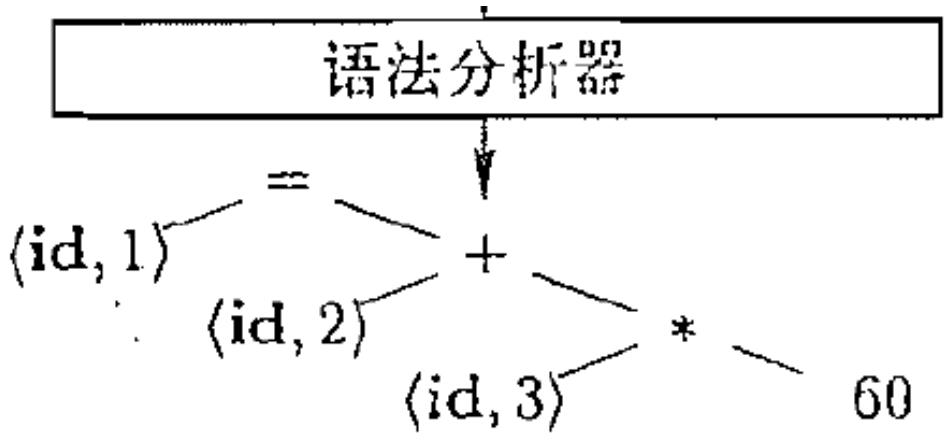
```
position = initial + rate * 60
```

词法分析器 (Lexer/Scanner): 将**字符**流转化为**词法单元** (token) 流。

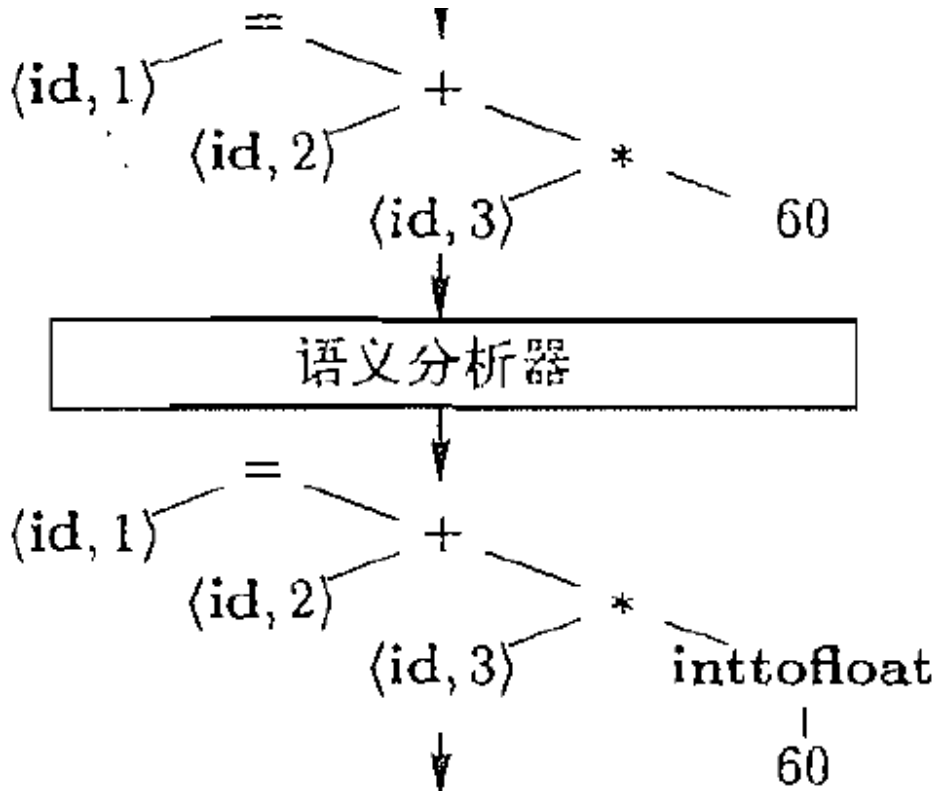
$$\boxed{\text{token} : \langle \text{token-class}, \text{attribute-value} \rangle}$$

$$\langle \mathbf{id}, 1 \rangle \quad \langle \mathbf{ws} \rangle \quad \langle \mathbf{assign} \rangle \quad \langle \mathbf{ws} \rangle \quad \langle \mathbf{id}, 2 \rangle \quad \langle \mathbf{ws} \rangle$$
$$\langle + \rangle \quad \langle \mathbf{ws} \rangle \quad \langle \mathbf{id}, 3 \rangle \quad \langle \mathbf{ws} \rangle \quad \langle * \rangle \quad \langle \mathbf{ws} \rangle \quad \langle \mathbf{num}, 4 \rangle$$

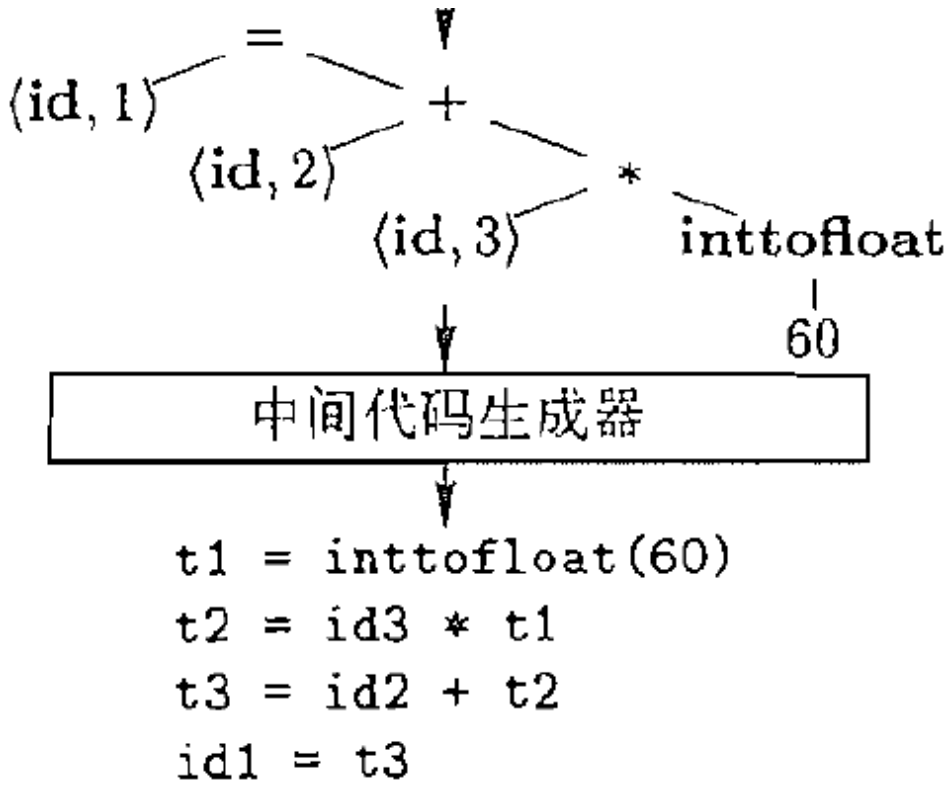
语法分析器 (Parser): 构建**词法单元**之间的语法结构, 生成**语法树**



语义分析器：语义检查，如**类型检查、“先声明后使用”约束检查**

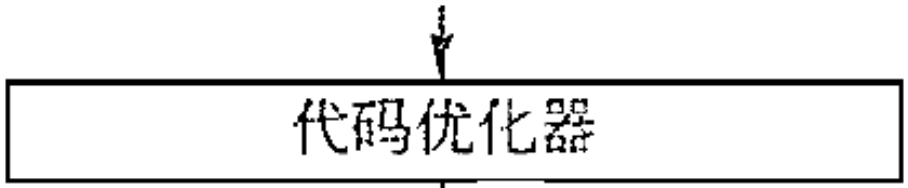


中间代码生成器: 生成中间代码, 如 “**三地址代码**”



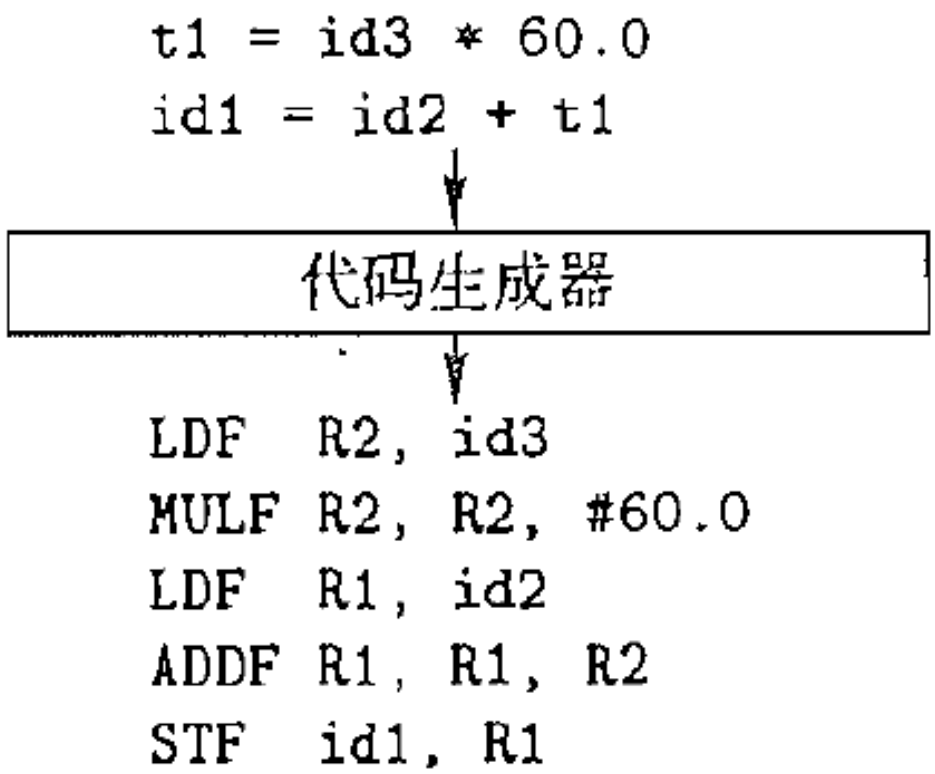
中间代码优化器

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




```
t1 = id3 * 60.0
id1 = id2 + t1
```

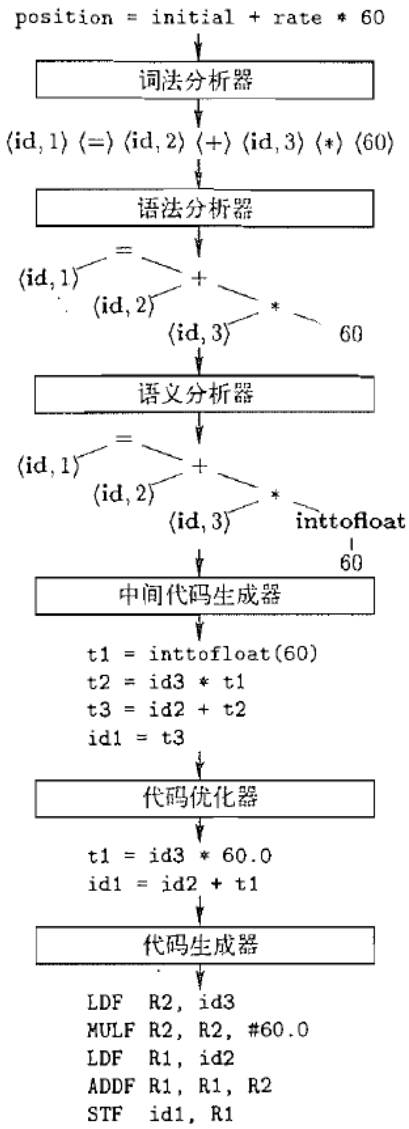
代码生成器：生成目标代码，主要任务包括**指令选择、寄存器分配**



符号表: 收集并管理**变量名/函数名**相关的信息

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

符号表

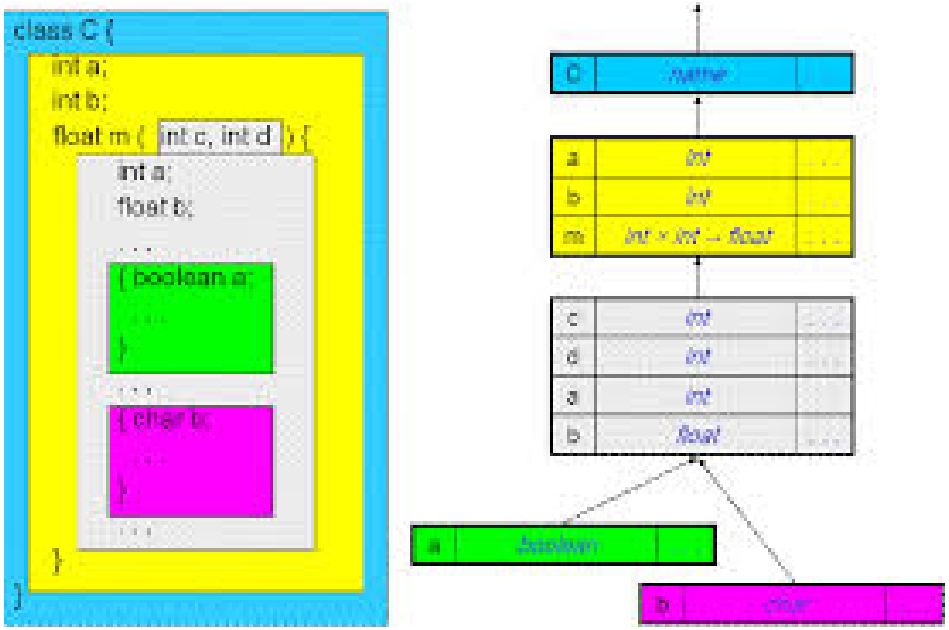


```
public class ST<Key extends Comparable<Key>, Value>
```

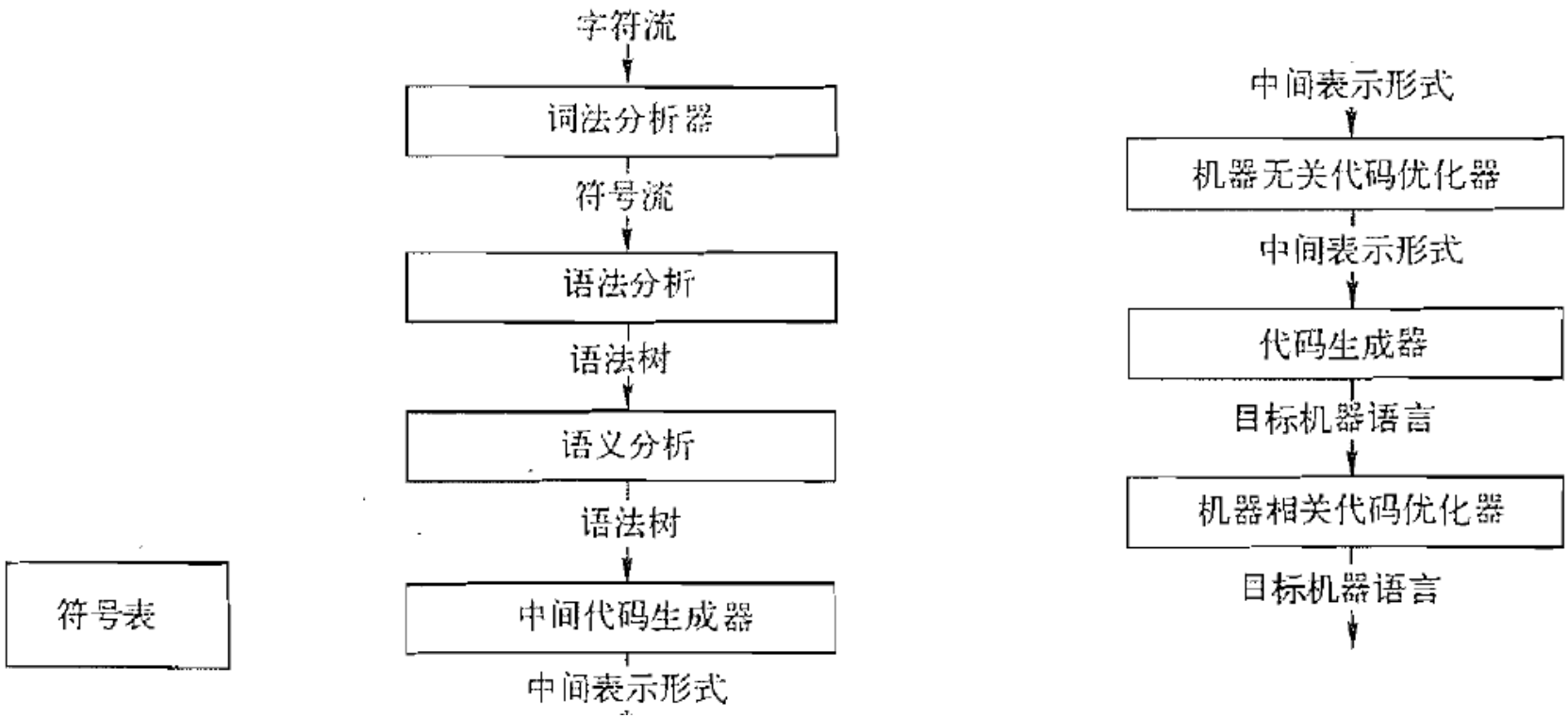
	ST()	*create an empty symbol table*
void	put(Key key, Value val)	*associate* `val` *with* `key`
Value	get(Key key)	*value associated with* `key`
void	remove(Key key)	*remove* `key` *(and its associated value)*
boolean	contains(Key key)	*is there a value associated with* `key`*?*
int	size()	*number of 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

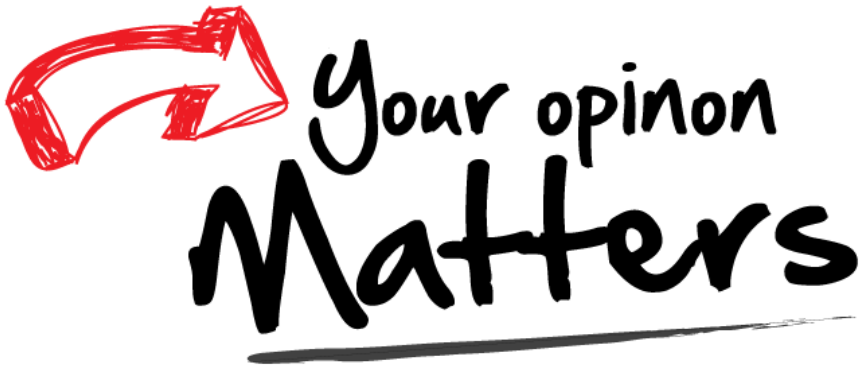


时间苦短，来不及优化



但是，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


Thank
You!



Office 926

hfwei@nju.edu.cn