

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/>

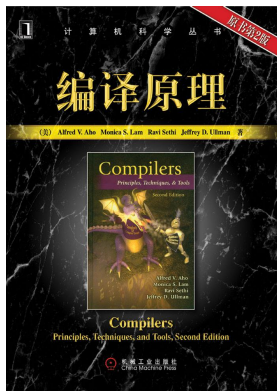
QQ 群号: 587330269



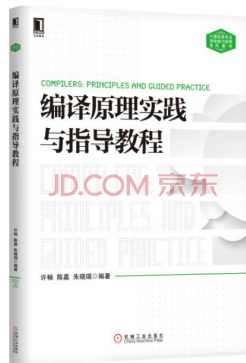
作业助教: 王腾、田汶哲、郑立铨

实验助教: 张灵毓、何伟

外援: 张天昀、唐瑞泽

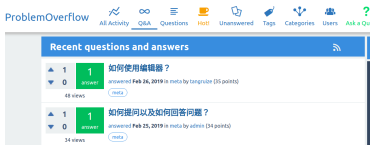


也可使用“本科教学版”



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

- 4 repositories selected...
- ☐ [courses-at-nju-by-hfwei/compiler-book](#) updated on Sep 15
 - ☒ [courses-at-nju-by-hfwei/compiler-lab](#) Private updated 2 days ago
 - ☒ [courses-at-nju-by-hfwei/compiler-lectures](#) updated 12 hours ago
 - ☐ [courses-at-nju-by-hfwei/compiler-papers-we-love](#) updated on Aug 14
 - ☒ [courses-at-nju-by-hfwei/compiler-problem-set](#) updated on Aug 10
 - ☐ [courses-at-nju-by-hfwei/compiler-resources](#) updated 17 hours ago
 - ☒ [courses-at-nju-by-hfwei/compiler-tutorials](#) updated on Aug 10

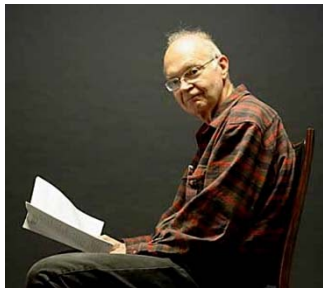


<http://n6.disalg.cn/>

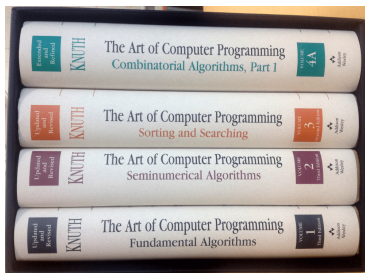
<https://github.com/orgs/courses-at-nju-by-hfwei/teams/compiler-course-at-nju-software/repositories>

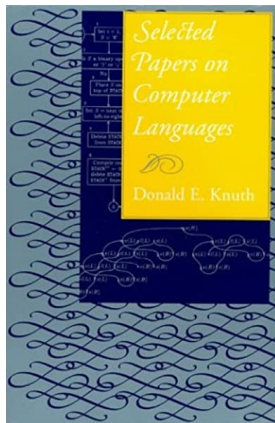


the “father of the analysis of algorithms”



Donald E. Knuth (1938 ~)

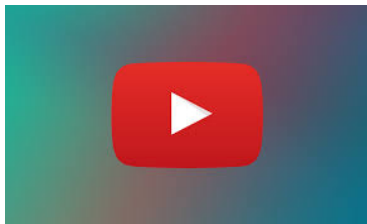




ALGOL 58 Compiler

LR Parser

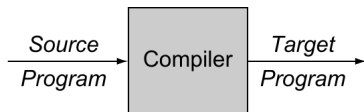
Attribute Grammar



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

“高级”语言 \implies (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

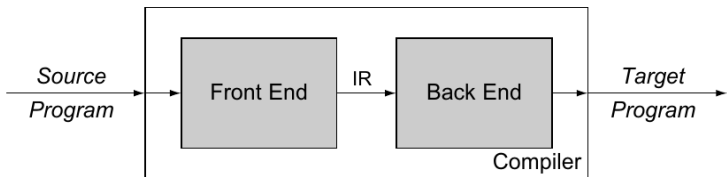


Q : 机器语言是如何跑起来的?

两个月的“编译器设计原理”之旅

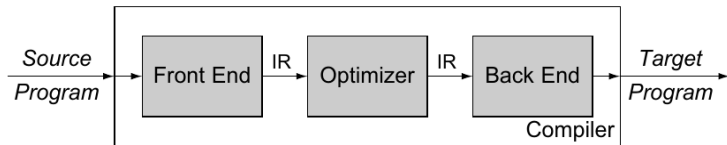


IR: Intermediate Representation (中间表示)



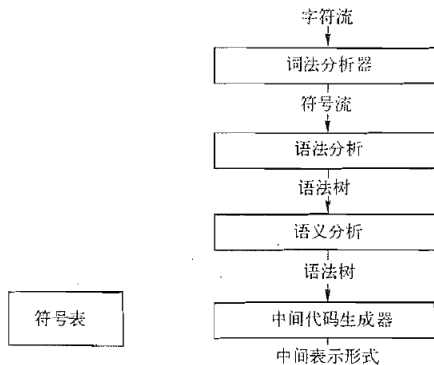
前端 (分析阶段): 分析源语言程序, 收集所有必要的信息

后端 (综合阶段): 利用收集到的信息, 生成目标语言程序



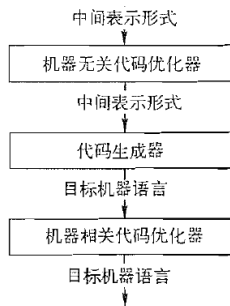
机器无关的中间表示优化

编译器前端：分析阶段



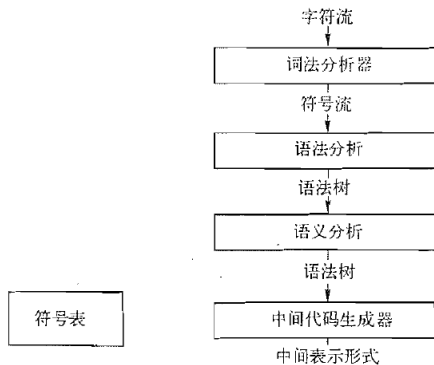
前四次必做实验

编译器后端：综合阶段



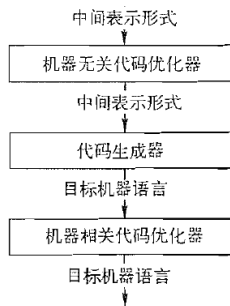
第五次选做实验

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

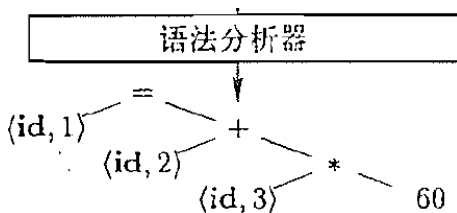
`position = initial + rate * 60`

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

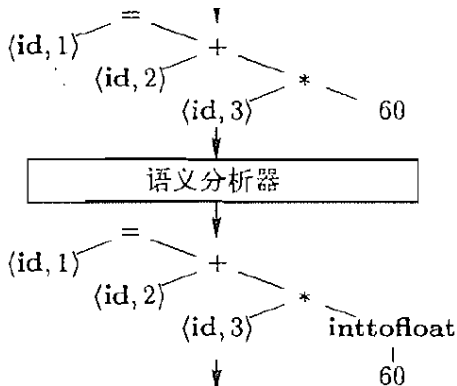
token : $\langle \text{token-class, attribute-value} \rangle$

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

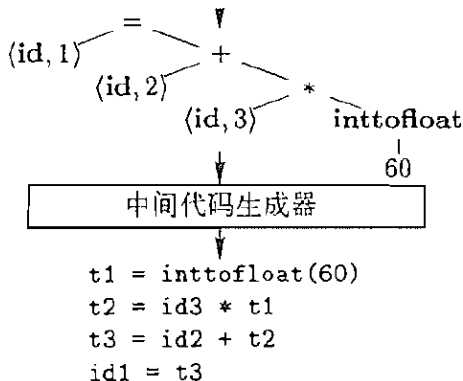
语法分析器 (Parser): 构建词法单元之间的语法结构, 生成语法树



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

```
t1 = id3 * 60.0
```

```
id1 = id2 + t1
```



代码生成器



```
LDF  R2, id3
```

```
MULF R2, R2, #60.0
```

```
LDF  R1, id2
```

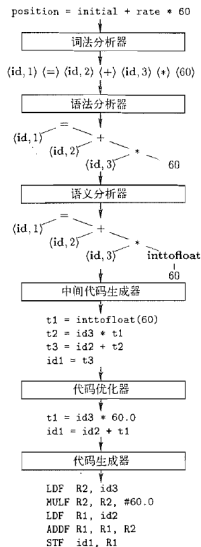
```
ADDF R1, R1, R2
```

```
STF  id1, R1
```

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

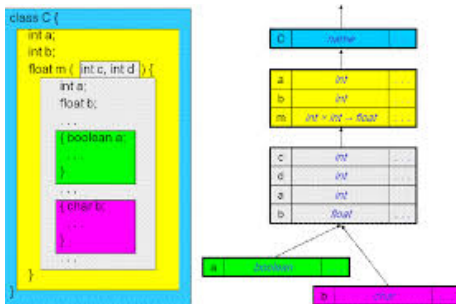



```
public class ST<Key extends Comparable<Key>, Value>
```

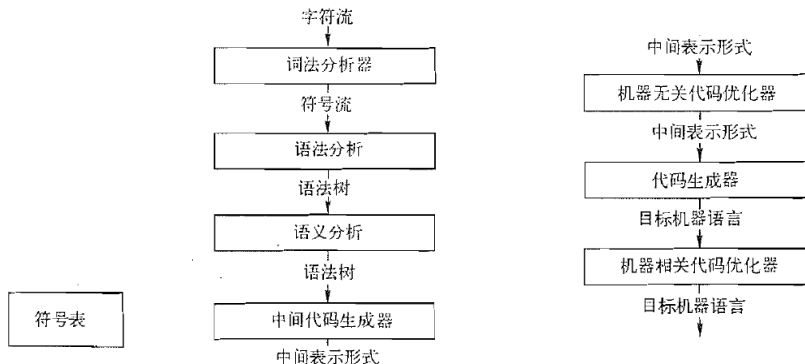
ST()	<i>create an empty symbol table</i>
void put(Key key, Value val)	<i>associate val with key</i>
Value get(Key key)	<i>value associated with key</i>
void remove(Key key)	<i>remove key (and its associated value)</i>
boolean contains(Key key)	<i>is there a value associated with key?</i>
int size()	<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



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