中国人民大学
RENMIN UNIVERSITY OF CHINA

编译原理与技术

--概 论

—— 刘爽 ——

中国人民大学信息学院

关于这门课

期末考试：30%
平时成绩：70%（上机作业+平时作业+课堂表现）
请假：提前与老师说明请假原因，提交请假条。
答疑：每周三下午，信息楼500

- 48学时，3学分，学科核心课程
- 时间&地点：1-16周，

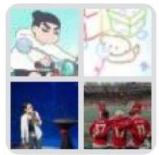
  理论课：周三1-3节，教三3505，
  上机实验：周三5-6节，理工配楼二层201B，202B

- 先修课：高级程序语言、汇编语言、数据结构和算法
- 参考书：
  - 《编译原理与技术》，孙莉等编著，高等教育出版社，2024年，ISBN：9787040619867
  - 《编译原理(第二版)》本科教学版 (Compilers: Principles, Techniques, and Tools (2nd Edition))，Alfred V.Aho等主编，赵建华等译，机械工业出版社，2009年，ISBN：9787111269298
  - 《程序设计语言编译原理（第3版）》，陈火旺等编著，国防工业出版社， 2014年，ISBN：9787118022070

课程平台

- 希冀平台
  - https://course.educg.net/indexcs/simple.jsp?loginErr=0
  - 课件、上机作业、平时作业
  - 登录学号为本人的“ruc学号”，初始密码“12345678”，登陆后尽快修改密码，账号问题联系助教
- 助教：
  - 杨世明，韩振涛，荣彦凯

- 课程微信群
  及时课程信息通知



群聊：2425-2编译原理课程群

该二维码7天内(2月22日前)有效，重新进入将更新

要求

- 按时上课（会有随机签到、提问）
- 上课时间对教师提出的问题积极思考及讨论
- 课后认真阅读参考资料，独立完成作业，按时提交作业
- 禁止抄袭

第1章：编译概述

- 1.1 编译与解释
- 1.2 编译程序概述
- 1.3 编译程序的结构
- 1.4 编译程序的设计与实现
- 1.5 编译技术应用

一个简单的自然语言翻译例子

“ I wish you success. ”

1）<u>词法分析</u>

I	wish	you	success
（代）	（动）	（代）	（名）

2）语法分析:

I	wish	you	success
（主语）	（谓语）	（间宾）	（直宾）

3）语义分析:

我希望你成功。

4）优化:

祝你成功。

计算机语言

- 低级语言（Low level Language)
  - 字位码、机器语言、汇编语言
  - 特点：与特定的机器有关；
    - 👍效率高；
    - 👎但使用复杂、繁琐、费时、易出错

- 高级语言 （High Level Language）
  - C, Java, Python, Ocaml 语言等
  - 特点：不依赖具体机器；
    - 👍移植性好、对用户要求低、易使用、易维护等；
    - 👎无法立即执行，需通过“编译程序”转化为与其等价的机器语言

程序设计语言发展与新一代编程语言

- 第一代语言

第一代语言（First-generation Language）通常称为机器语言，它与机器孪生。实际上，它完全依赖于机器的指令系统（Instruction System），以二进制代码表示。这类语言的程序既难编写、又难读懂

- 第二代语言

第二代语言（Second-generation Language）通常称为汇编语言，它将机器语言符号化。用符号来代表机器语言的某些属性

程序设计语言发展与新一代编程语言

- 第三代语言

第三代语言（Third-generation Language）通常是指高级语言，这类语言的设计基础与冯·诺依曼体系结构有关。高级语言程序按语句顺序执行，因此又称为面向语句的语言（Sentence-oriented Language）

- 第四代语言

第四代语言（Fourth-generation Language）是说明性语言（Declaration Language），它只需要告诉（说明）计算机去“做什么”，不必告诉计算机“怎么做”。所以，这类语言又称为超高级语言或甚高级语言（Veryhigh-level Language），典型的例子是SQL语言

程序设计语言发展与新一代编程语言

- 新一代语言

未来的编程语言将更加简洁、易用、易理解，脱离传统的冯诺依曼体系约束，扩展到了一系列新兴领域。代表语言包括QCL、Latte、Haskell、AutoPASS以及Rust等

  - 量子编程语言
  - 神经网络编程语言
  - 人工智能编程语言
  - 嵌入式编程语言
  - 其他编程语言

程序设计语言的处理系统

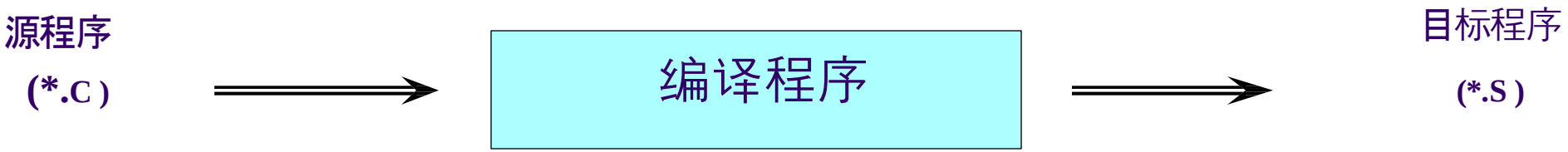
- 程序语言处理系统的输入程序被称为源程序（Source Program），编写输入程序的语言称为源语言（Source Language）
- 语言处理系统必须把源程序变换为能在某种计算机上执行的形式，这部分被称为综合部分，其输出的程序称为目标程序（Object Program）或目标代码（Object Code），相应的语言称为目标语言（Object Language）

翻译程序

- 翻译程序：汇编程序或者编译程序
  - 汇编（Assemble) 程序 :源语言为汇编语言，目标语言为机器语言。
  - 编译（Compile）程序: 源语言是高级语言，目标程序为某种中间语言或者汇编语言。

编译程序

- 将高级程序设计语言（源语言程序）翻译成逻辑上等价的低级语言(目标语言程序，例如汇编语言,机器语言)的翻译程序

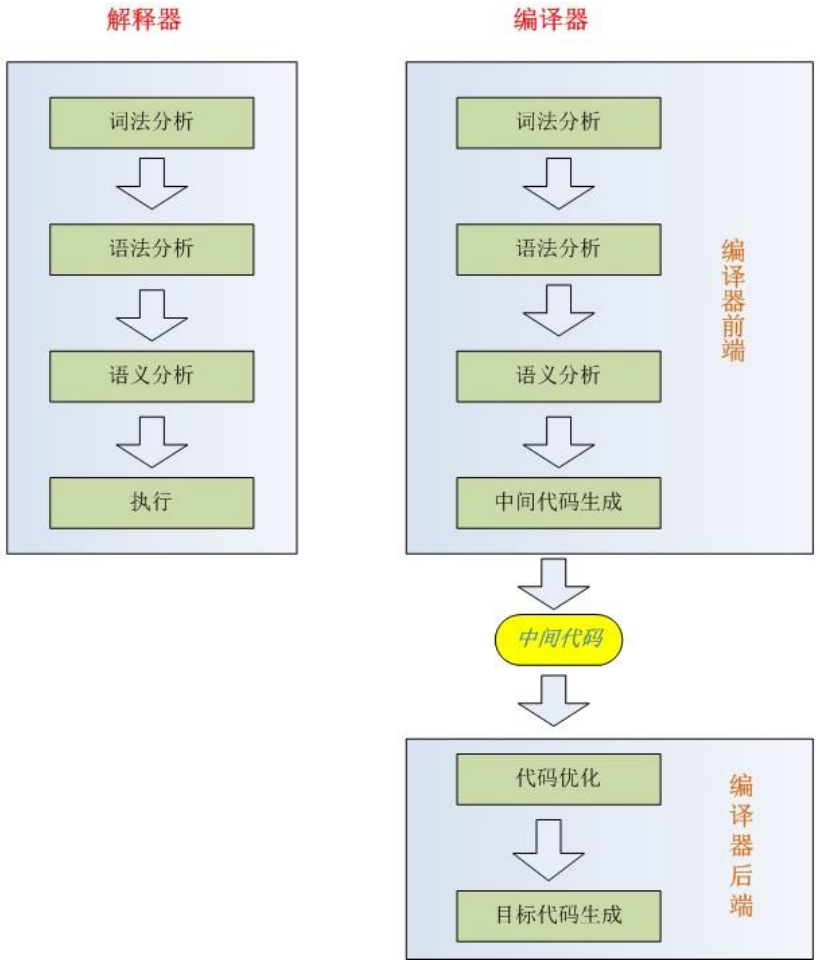


编译与解释

- 编译程序（Compiler)： 把**高级程序设计语言**翻译成**等价**的**低级语言**的程序。
  - **高级语言源程序→编译程序→目标程序**， e.g., C/C++, Java
- 解释程序（Interpreter)： 输入源程序，但不产生目标程序，而是按照语言的定义，边解释边执行源程序本身。e.g., scripting languages, Python
- 混合编码（编译-解释执行程序）：是一种折衷形式。即对运行较慢的部分采用编译，其它部分采取解释执行。
  - 例如JVM以及浏览器中的JIT技术
  - 许多用于编译程序的构造技术同样也适用于解释程序。

编译器和解释器

- 编译器(compiler)和解释器(interpreter)的比较
  - 相同点（执行相同的任务）：
    - 检查输入程序并确定这个程序是否为有效程序
    - 建立一个内部模型来刻画输入程序的结构和含义
    - 决定在执行期间值的存放位置
  - 不同点（执行的行为不同）：
    - 编译器以一个可执行程序的描述作为输入，以另一个等价的可执行程序的描述作为输出。
    - 解释器以一个可执行程序的描述作为输入，以执行这一可执行程序描述的结果作为输出。



程序设计语言的处理系统

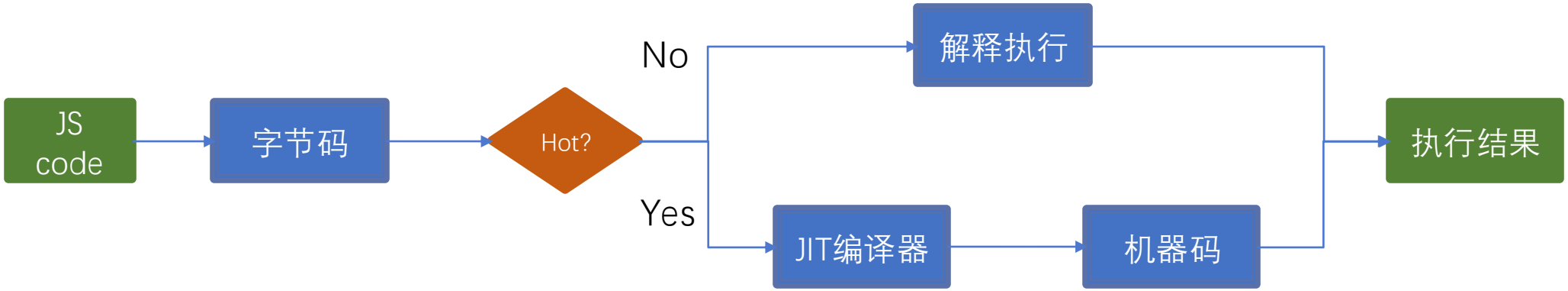
- 语言处理系统的作用是把用软件语言书写的各种程序处理成可在计算机上执行的程序，或最终的计算结果，或其他中间形式
- 按照处理方法，语言处理系统可分为编译型解释型和混合型三类
  - 编译型语言处理系统：采用编译方法的语言处理系统
  - 解释型语言处理系统：采用解释方法的语言处理系统
  - 混合型语言处理系统：兼有编译和解释两种方法的语言处理系统

我们为什么需要编译程序

- 编写、调试、维护、理解用汇编语言(assemble language)程序是很困难的
- 自从第一个编译器出现之后，软件产品的数量有了巨大的增加
- 好的编译器是计算机科学的缩影
  - 包含大量的技术：贪婪算法（寄存器分配）、动态规划（指令筛选）、有穷自动机和下推自动机（扫描和语法分析）、不动点算法（数据流分析）
  - 处理复杂的问题：动态分配、同步、命名、局部化、存储器分层管理
  - 提供完整的解决方案：有机的结合算法、软件体系结构和软件工程的各种理论，对棘手问题给出综合性的解答方案。

思考

- 网页（html/js 格式）的翻译程序是什么？
- （接上题）这个翻译程序是编译器还是解释器？



第1章：概论

- 1.1 编译与解释
- **1.2 编译程序概述**
- 1.3 编译程序的结构
- 1.4 编译程序的设计与实现
- 1.5 编译技术应用

1.2 编译程序概述

- 源程序----------词法分析，语法分析，语义分析和中间代码生成，优化，目标代码生成----------->目标程序

1. **词法分析：输入源程序，对其扫描并分析，识别 各单词符号（关键字、标识符、常数、算符、界符等）**

   如：Python语句

   - **for　　i　　in　　range　　(　　1　　,　　100　　)**

     关键字　　整变量　　关键字　　关键字　　界符　　整常量　　界符　　整常量　　界符

- 词法分析的依据是词法规则。

1.2 编译程序概述

2. 语法分析：根据语法规则,把单词串分解成各类语法范畴（短语、子句、表达式、程序段、程序）。

- 语法分析所依循的是语言的语法规则。 **sum=current + accumulative**

3. 语义分析和中间代码生成

- 根据各语法成分的**语义规则**写出其中间代码。
- 常用的中间代码有：三元式、四元式、间接三元式、树等。

例如： 四元式表示为：

+	i	M	M
操作码	第一操作数	第二操作数	结果

1.2 编译程序概述

4. 优化: 对中间代码进行加工变换， 以期在最后 阶段 产生出更为高效的 (省时间和空间) 的目标代码。

•常做的优化有：公共子表达式的外提、循环优化、算符归约等。

例如：

```
for k := 1 to 100 do
begin
  m := i + 10 * k ;
  n := j + 10 * k ;
end;
```

四元式代码

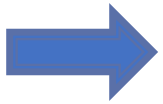
序号	操作码	第一	第二	结果	
1	:=	1		k	
2	j<	100	k	(9)	\\ for k:=1 to 100
3	*	10	k	t1	
4	+	i	t1	m	\\ m:=i+10*k
5	*	10	k	t2	
6	+	j	t2	n	\\ n:=j+10*k
7	+	1	k	k	
8	j			(2)	

- 3和5, 200次乘法（优化）

1.2 编译程序概述

- 优化后的四元式: 优化所依循的原则是程序的**等价变换原则**。

序号	操作码	第一	第二	结果	
1	: =	1		k	
2	j<	100	k	(9)	\\ for k:=1 to 100
3	*	10	k	t1	
4	+	i	t1	m	\\ m:=i+10*k
5	*	10	k	t2	
6	+	j	t2	n	\\ n:=j+10*k
7	+	1	k	k	
8	j			(2)	



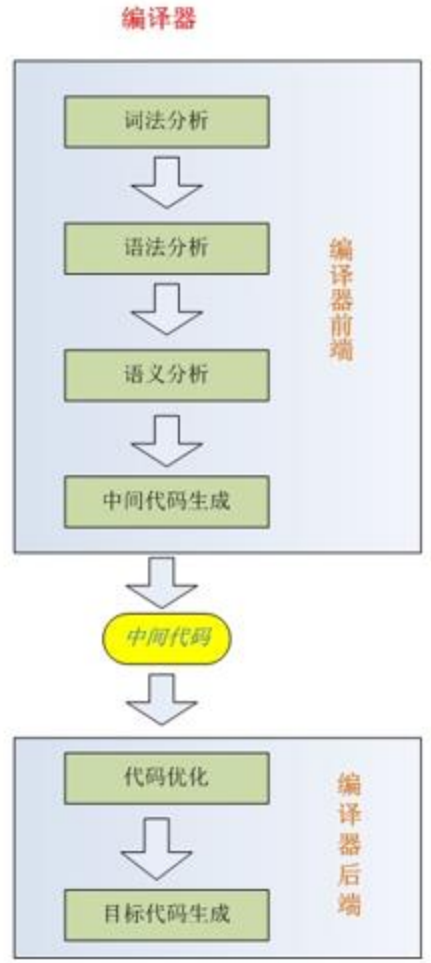
序号	操作码	第一	第二	结果
1	: =	i		m
2	: =	j		n \\ m,n初值
3	: =	1		k
4	j <	100	k	(9) \\ for k:=1 to 100
5	+	m	10	m
6	+	n	10	n \\ m,n循环+10
7	+	1	k	k
8	j			(4)

1.2 编译程序概述

5 目标代码的生成：

- 在中间代码（或经过优化）的基础上，生成某具体的硬件代码，可以是绝对机器代码，或汇编代码。

- 总之，编译要先**分析**，以确定源程序的功能；然后**综合**，以生成该功能的目标程序。



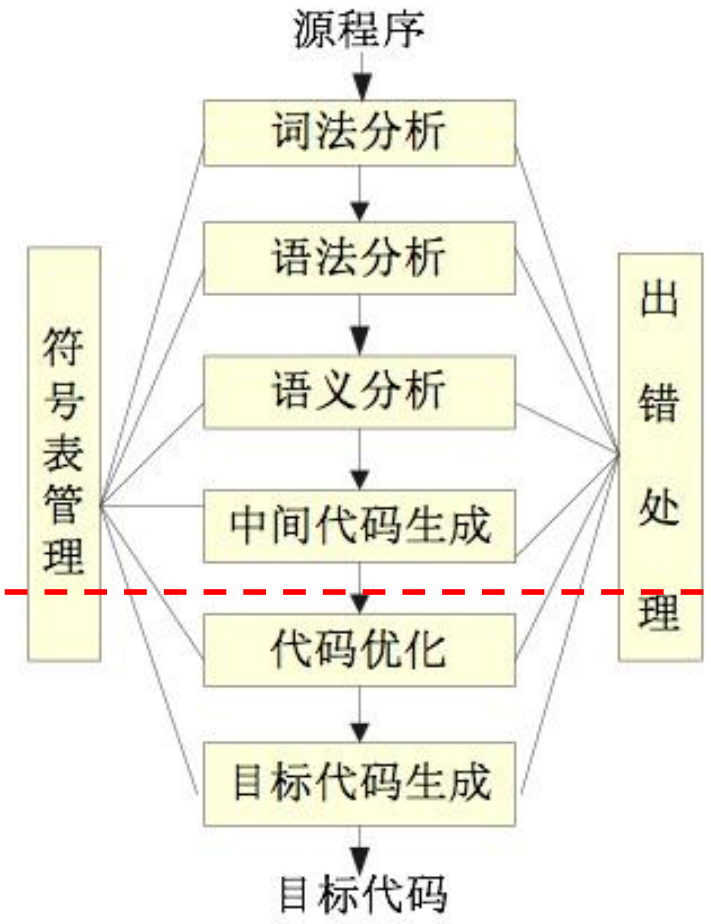
思考

- 编译程序的5个阶段分别是什么？

第1章：概论

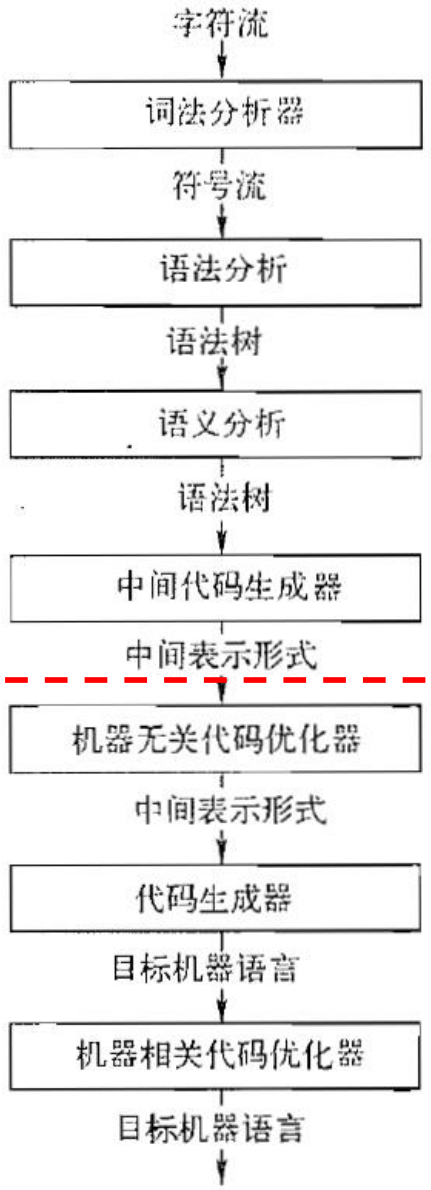
- 1.1 编译与解释
- **1.2** 编译程序概述
- 1.3 编译程序的结构
- 1.4 编译程序的设计与实现
- 1.5 编译技术应用

编译程序的结构



编译的前端
(Front End)
分析部分
与源语言有关

编译的后端
(Back End)
综合部分
与目标语言有关



例子（1/3）

- 例1： 对赋值语句的分析与综合过程。

position ：= initial+rate* 60（其中 var position, initial, rate: real;)

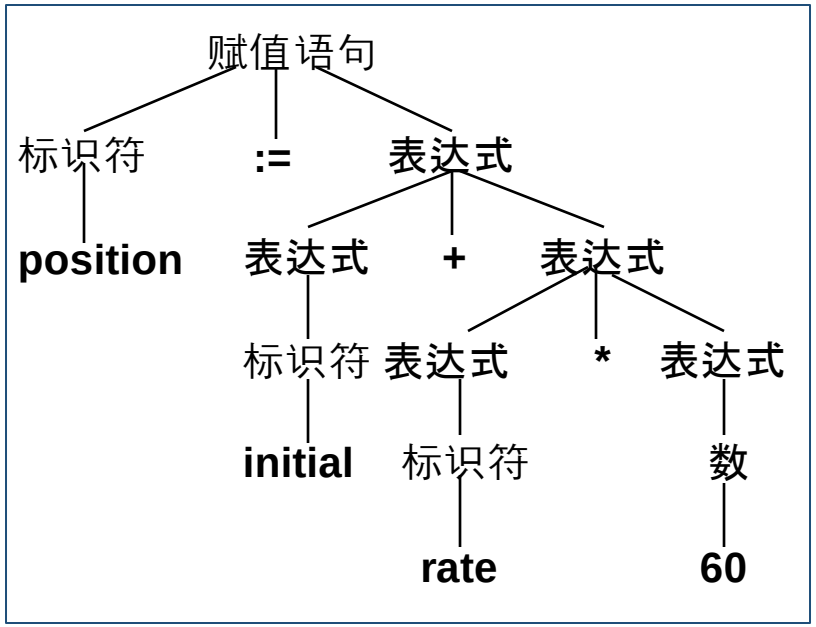
(1) 词法分析

- 跳过空格,注释语句

标识符	position initial rate
赋值号	: =
加号	+
乘号	*
整形（数）	60

(2) 语法分析

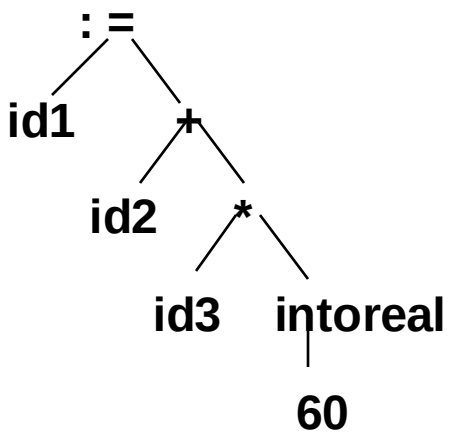
- 结构层次分析—语法分析树，对于表达式的判定。



例子（2/3）

position ：= initial + rate * 60（其中 var position, initial, rate: real;)

- （3）语义分析:
  - 类型检查：标识符的类型匹配。
  - 类型转换：整--实（intoreal) itoa,atoi　字符--整 (chartoint) ctoi,itoc



例子（3/3）

符号表

position	---
initial	---
rate	---

- 4) 中间代码生成:
  - 三地址代码（类似汇编语言）

```
intoreal   60          t1
 *         id3   t1    t2
+          id2   t2    t3
:=         t3          id1
```

- 5) 代码优化:　（时、空）

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

6)目标代码生成

—— 中间代码 (**R1**、**R2** 寄存器)

```
MOVF   id3 ,  R2
MULF   #60.0 , R2
MOVF   id2 ,  R1
ADDF   R2  ,  R1
MOVF   R1  ,  id1
```

*说明：第一操作数（源操作数） 第二操作数（目标操作数）

遍（PASS）

- 遍：对源程序（包括源程序的中间表示形式）**从头到尾扫描一次**并作有关的加工处理，**生成新的源程序中间形式或目标程序**，通常称之为一遍。上一遍的结果是下一遍的输入，最后一遍生成目标程序。

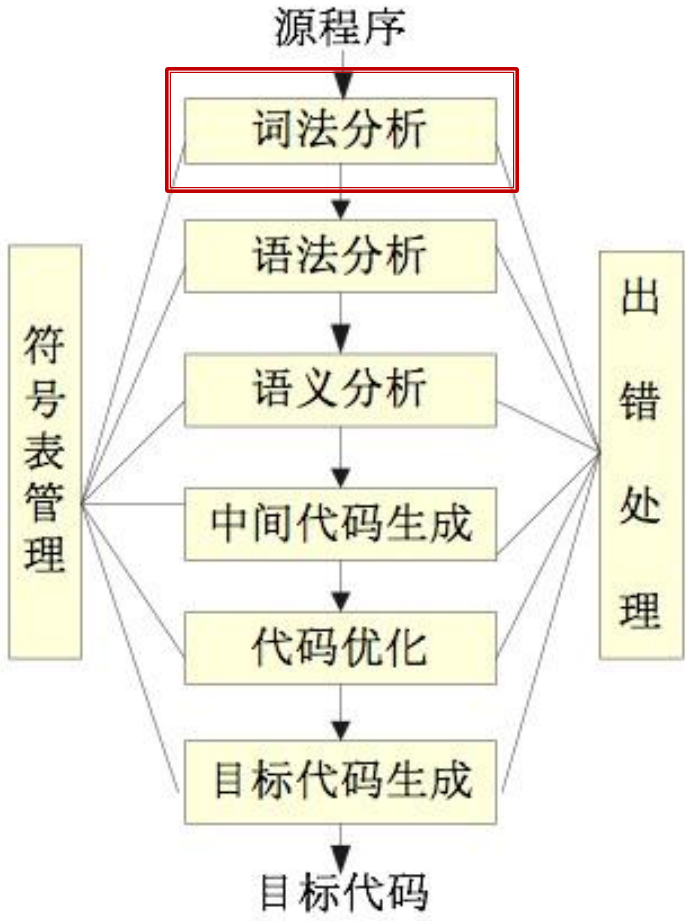
- 遍与基本阶段的区别：
  - 五个基本阶段是将源程序翻译成目标程序在逻辑上要完成的工作
  - 遍是指完成上述五个基本阶段的工作要经过几次扫描处理

编译程序

- 编译程序必须完成两个主要任务：
  - 一是对源程序的分析
  - 二是生成目标程序
- 通常将分析部分划分为词法分析、语法分析和语义分析三个阶段
- 综合部分接受分析部分所产生的中间代码，然后根据具体的目标机系统生成目标代码，并对代码进行优化处理
- 一般是将综合部分分为目标代码生成和代码优化两个阶段。

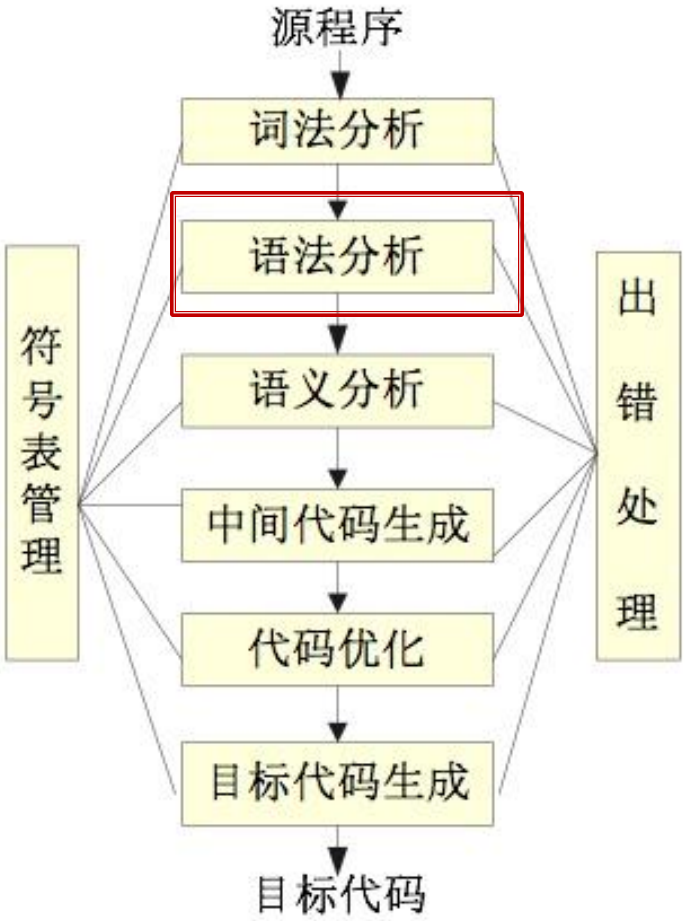
编译程序

- 词法分析
  - 编译器将源程序看成一个很长的字符串，首先对它进行从左到右的扫描，并进行分析，识别出符合词法规则的单词（符号），或称记号（Token）。如基本字、标识符、常数、运算符和界符等。这些符号以某种内码表示方法提供给后续过程进一步处理
  - 如果在词法分析过程中发现不符合词法规则的非法单词符号，则做出词法出错处理，给出相应出错信息



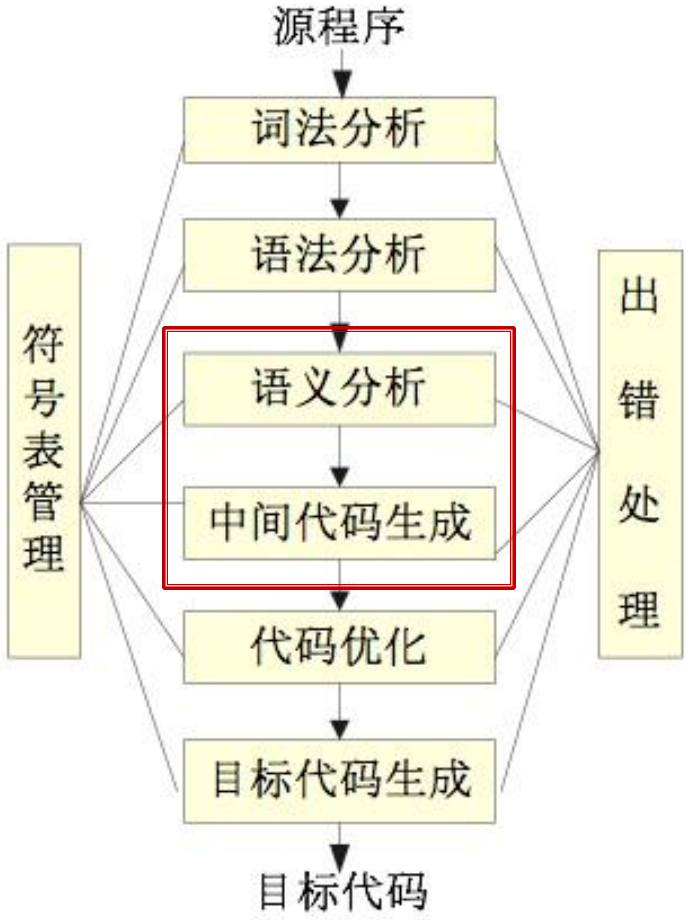
编译程序

- 语法分析
  - 语法分析是对词法分析识别出来的符号流（也可看成符号串）。按语法规则进行分析，识别出各类语法单位，如表达式、短语、子句、句子和程序等，以便后续步骤进行分析与合成
  - 语法分析通常是一种结构分析，分析的结果形成一棵语法树（分析树）。如果在分析过程中发现不符合语法规则的符号串，将做出语法出错处理，给出相应的出错信息



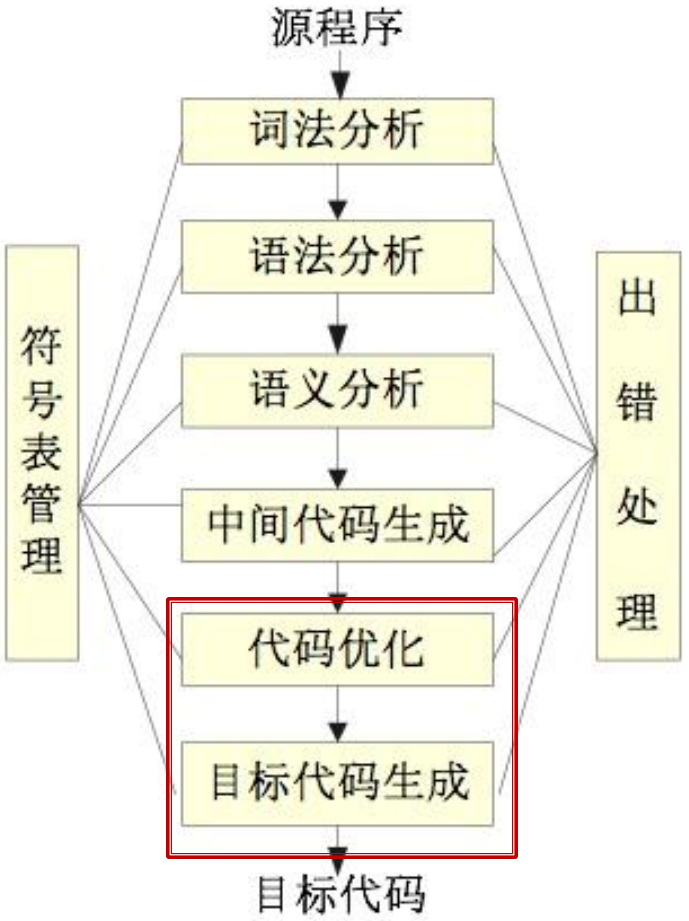
编译程序

- 语义分析
  - 对经过词法分析和语法分析后的程序，如果没有错误，就可对其进行语义分析，并可按语义要求对各种语法单位进行实质性翻译
  - 大多数编译器采用中间语言来描述源程序的语义
  - 这种中间语言往往对应某种抽象机，其结构简单，语义明确，易于被翻译成二进制代码，同时也便于优化和移植



编译程序

- 优化
  - 语义分析产生的中间代码不依赖于任何实际机器，因此它易于实现一些等效变换，使生成的目标程序占用空间少，执行更快，从而使目标程序优化（Optimization）
- 目标代码生成
  - 根据优化后的中间代码及有关信息，可生成较为有效的目标代码，即目标机的机器语言程序或汇编语言程序
  - 如果生成的是汇编语言程序，还需由汇编器将其汇编成目标机的机器语言程序。
  - 为使目标机的机器语言程序更加有效，代码生成还应充分利用机器资源，如充分利用寄存器，选择执行时间短的指令等

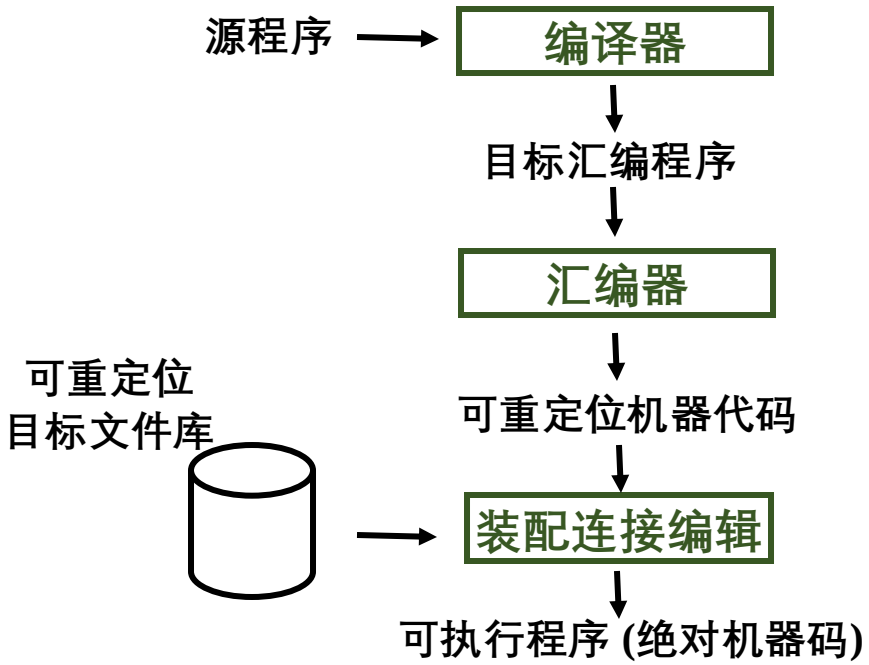


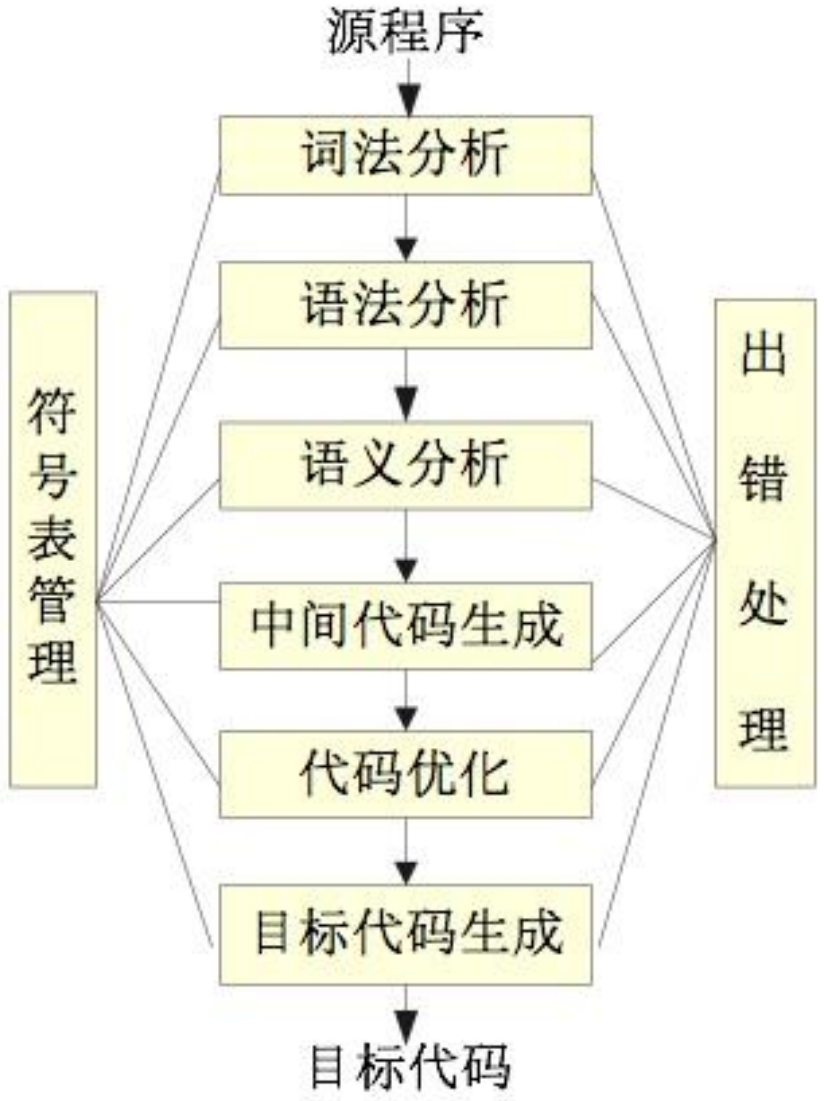
编译程序

- 从逻辑上来看，编译器就是按这5个阶段对源程序进行编译的
- 事实上，按上述5个阶段编译出来的目标程序不能立即执行，因为有些语言允许源程序的各个模块独立进行编译，生成的目标程序称为目标模块（Object Module）
- 这些目标模块和输入/输出、标准函数等系统模块需要经过链接程序（Linker）进行链接，成为一个可重定位程序（Relocatable Program），再由装入程序（Leader）装入到内存中。机器才能正确执行

编译过程

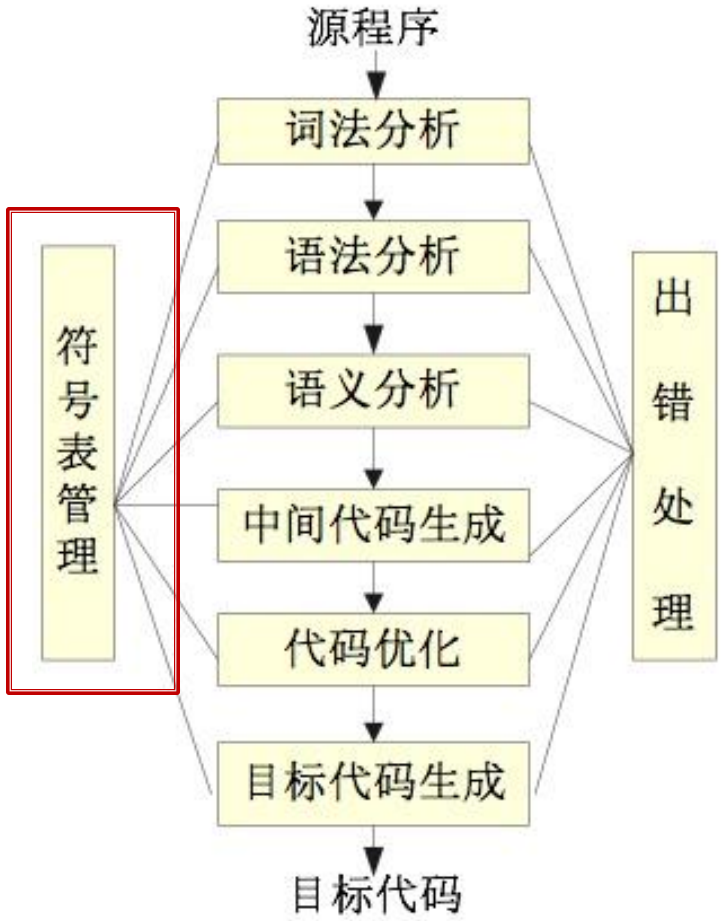
- 源代码到可执行程序
- 本课程关注编译器的原理





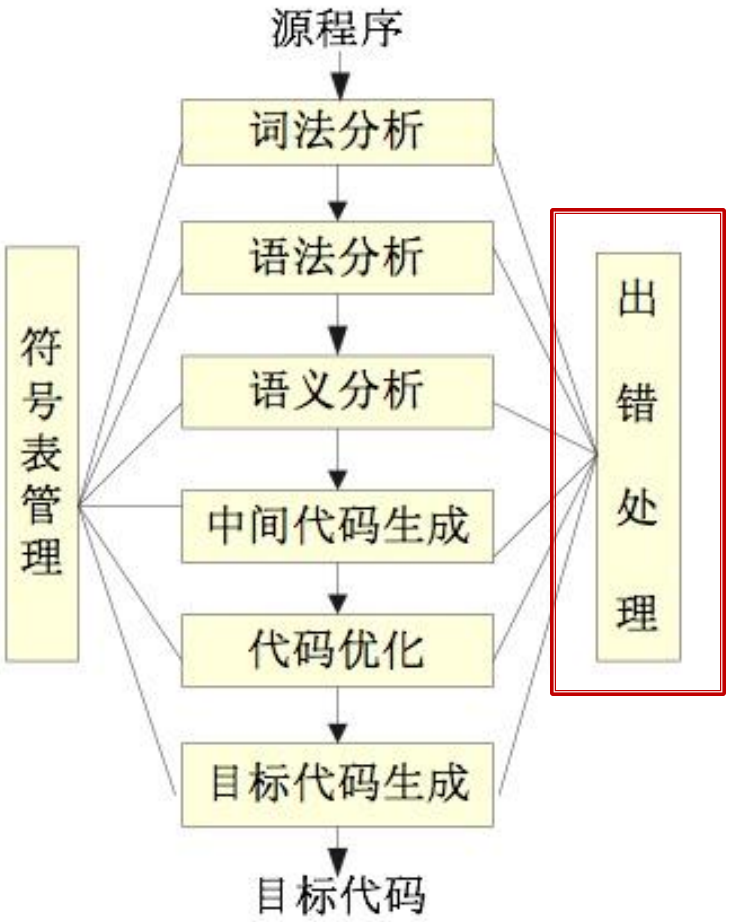
编译程序

- 符号表管理
  - 程序员在编写程序时，对各种数据对象和实体进行说明。编译程序根据这些说明，为各种实体建立描述符，并搜集这些实体的属性信息。以供编译各阶段使用
  - 描述符实际上是存放实体属性的表格，根据实体的不同，表格形式也不同，其中用得最多的是符号表。编译程序应当有一组表格管理程序，负责对这些表格的建立和维护（包括引用、查找和更新等）



编译程序

- 出错处理
  - 编译程序各个阶段都可能发现源程序中的错误，特别是在语法分析阶段，可能出现大量的语法错误
  - 发现错误后，编译程序就要进行处理，报告错误的性质，发生错误的地方等，同时要将错误所造成的影响限制在尽可能小的范围，以便对源程序的剩余部分继续编译下去，查出剩余部分的错误



练习

- 分别列举属于编译器前端和编译器后端的阶段

- 编译器的每个阶段都需要独立的一遍来完成，这种说法是否正确?

第1章：概论

- 1.1 编译与解释
- **1.2** 编译程序概述
- 1.3 编译程序的结构
- 1.4 编译程序的设计与实现
- 1.5 编译技术应用

编译器的基本原则

- 编译器是工程对象，是具有独特目标的大型软件系统，两个设计原则必须遵守
  - 不违背原义
    - 编译器必须保持被编译程序的含义不变
    - 这一原则是编译器设计者与编译器用户之间的契约的核心
  - 实用性原则
    - 编译器必须用某种明确的方式改进输入程序
    - 例如代码优化等对输入程序的改进

1.4编译程序的设计与实现

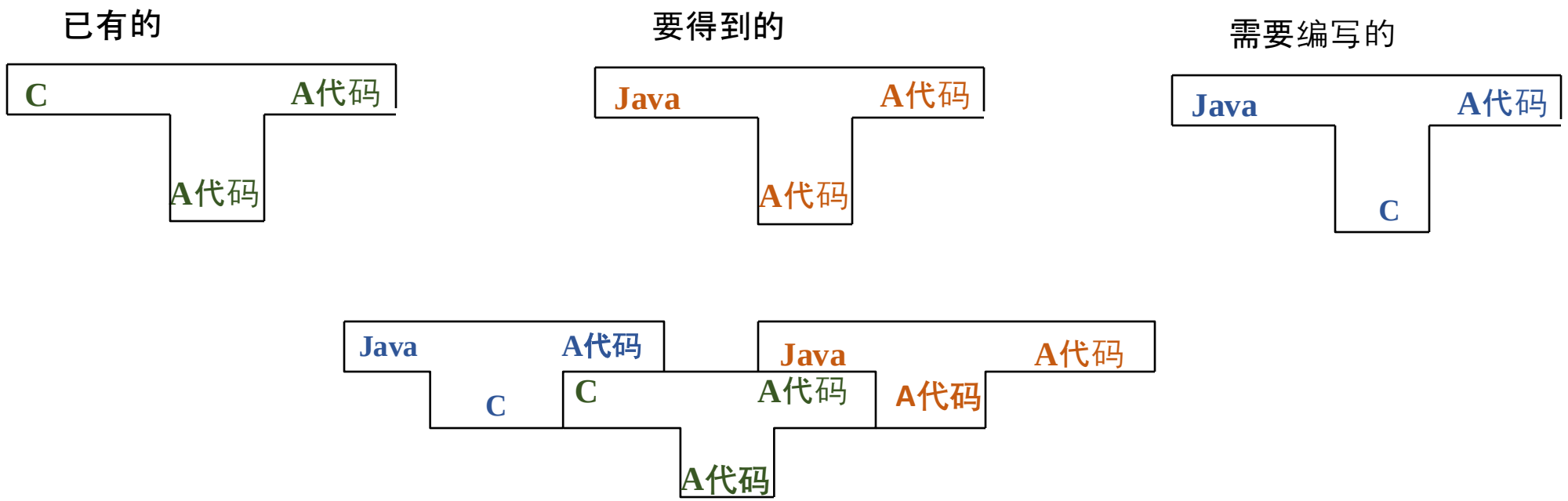
- 用“ T”型图表示编译程序。
- 读作：用实现（编译）语言编写的，
  生成目标语言程序的源语言编译程序。
- 例如：用C语言编写的，
  生成Java Bytecode的Java编译程序。

- 一般不采用低级语言作为实现语言。

1.4编译程序的设计与实现

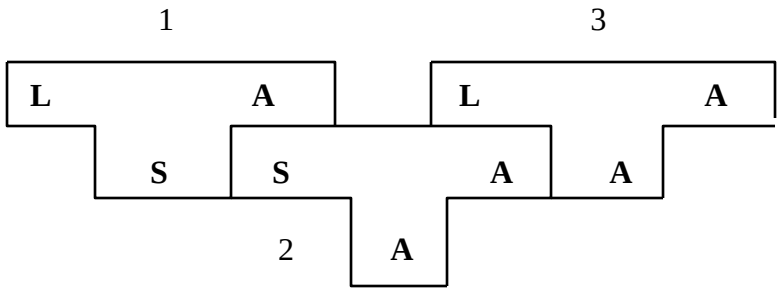
T型图的结合规则：

（1）中间T图的两臂上的语言分别与左右两个T图脚上的语言相同。（2）对于左右两个T图而言，其两个左端的语言必须相同，两个右端的语言亦必须相同。



1.4编译程序的设计与实现（cont）

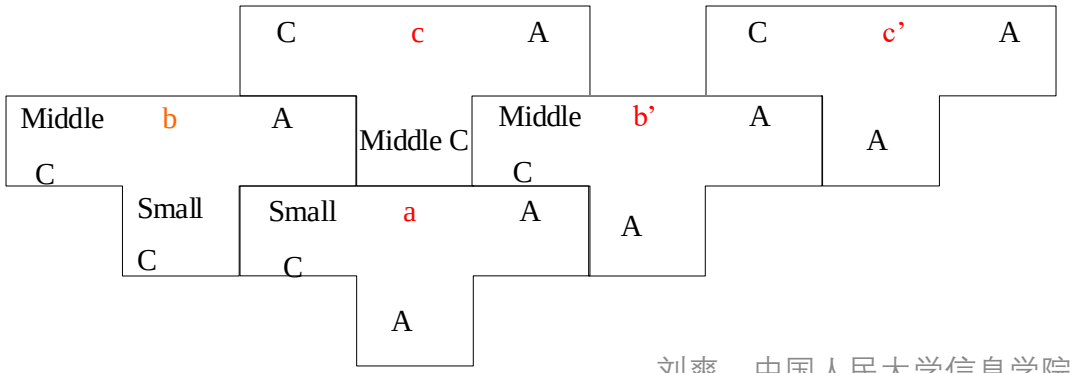
- 用第2个编译程序去编译第1个，得到第3个编译程序



- 如何获得A机器上第一个可书写编译程序的高级语言？
  - 自展
  - 移植

自展 (自编译)

- 在A机器上用机器语言或汇编语言编写高级语言的子集，例如，Small C的编译程序(a)
- 经Small C 编写的Middle C的编译程序（b）
- 经Small C 的编译程序编译后，得Middle C的编译程序（b’）
- 由Middle C 编写全C的编译程序（c）
- 经Middle C 的编译程序编译后，得C的编译程序（c’）

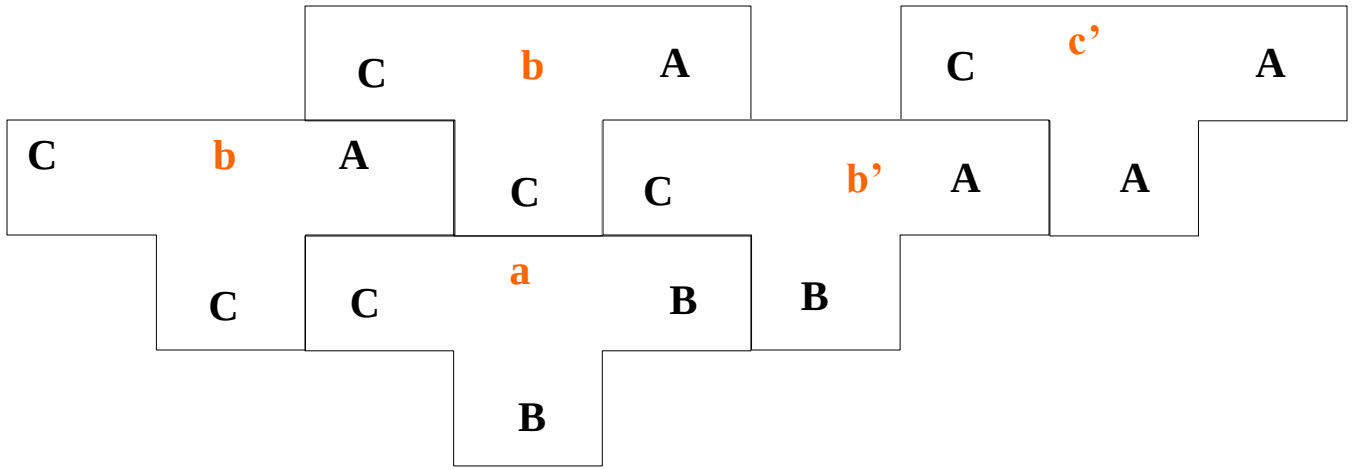


自展(cont)

- 自展方式就像滚雪球那样扩大语言的编译程序。
- 利用自展的方式使我们用机器语言或汇编语言编写高级语言的编译程序的工作量减少到最低限度。
- 注意：实际实现要做多次的自编译才能得到可靠的编译。

移植

- 在B机器上已有一个可运行的C编译程序（a）
- 只要我们编写一个用C语言书写的，产生A机器代码的C编译程序(b),按如下T型图的运算去做，就得到A机器上所要的C编译程序。
- (b),(a) ⟶ (b’)；移植；(b), (b’) ⟶ (c’);



移植(cont)

- 优点：用高级语言书写的编译程序，生产的周期短，可靠性高，易 修改、扩充与维护，并且易于移植。
- 缺点：代码量长，但随着存储能力的提高、运行速度越来越快，程序正确性是主要矛盾。

- UNIX操作系统的实用程序 LEX，YACC 自动地生成词法分析器和语法分析器.

练习

- 假设已经有用汇编语言书写的C语言的编译器（目标语言为汇编语言），现在想要得到用汇编语言书写的Python语言的编译器（目标语言为汇编语言），请用移植的方式完成该编译器的设计和实现。（画出T型图的表示方式）

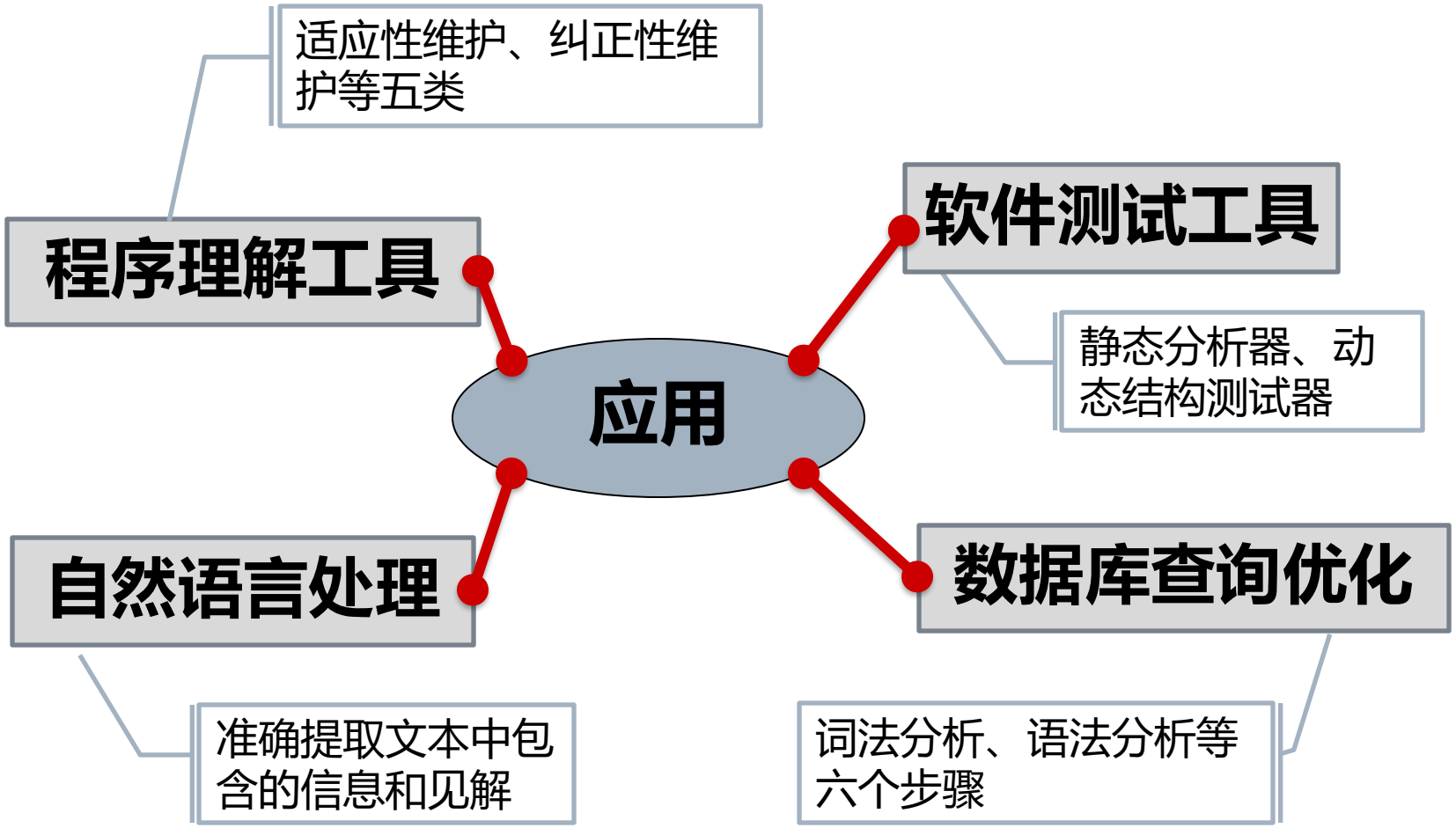
练习

- 假设已经有用汇编语言书写的C语言的编译器（目标语言为汇编语言），现在想要得到用Java Bytecode书写的Java语言的编译器（目标语言为Java Bytecode），请用移植的方式完成该编译器的设计和实现。（画出T型图的表示方式）

第1章：概论

- 1.1 编译与解释
- **1.2** 编译程序概述
- 1.3 编译程序的结构
- 1.4 编译程序的设计与实现
- 1.5 编译技术应用

编译技术应用


应用
程序理解工具
适应性维护、纠正性维护等五类
软件测试工具
静态分析器、动态结构测试器
自然语言处理
准确提取文本中包含的信息和见解
数据库查询优化
词法分析、语法分析等六个步骤

程序理解工具

- 程序理解是计算机科学的一个领域，与软件工程师维护现有源代码的方式有关
- 识别和研究所涉及的认知和其他过程。结果用于开发工具和培训
- 软件维护任务有五类：适应性维护、纠正性维护、完善性维护、代码重用和代码利用
- **程序理解工具通过对程序进行分析，逆向提取面向对象程序的类关系图和程序中模块（函数）间的调用关系，记录程序数据的静态属性和结构属性，并画出程序的控制流程图，帮助用户理解程序**

软件测试工具

- 软件测试是为了发现错误而执行程序或系统的过程
- 软件测试可以提供一个客观、独立的软件视图，让企业了解软件实施的风险
- 测试技术包括但不一定限于:
  - 在行业视角、业务视角、实施的可行性、可用性、性能、安全性、基础设施考虑等各种背景下分析产品的完整性和正确性要求
  - 审查产品架构和产品的整体设计
  - 与产品开发人员合作，改进编码技术，设计模式，根据各种技术（如边界条件等）将测试编写为代码的一部分
  - 以检查行为为目的执行程序或应用程序

软件测试工具

- 软件测试可以向用户或赞助商提供有关软件质量及其故障风险的客观、独立的信息
- 有两种软件测试方法:
  - (一)静态分析器
    - 静态分析器对源程序进行静态分析，即在不运行该程序的情况下对其进行分析，以发现程序中潜在的错误或异常
    - 静态分析器的分析部分类似于语法分析程序，并需要采用编译优化常用的数据流分析和控制流分析方法
    - 可以检测出那些不可能执行到的源代码，或某些未定义（未给变量赋值）就引用的变量
    - 还能发现程序中试图使用一个实型变量作为指针等错误

软件测试工具

（二）动态结构测试器

- 动态结构测试是在给定一组测试用例下执行程序，记录程序实际执行轨迹，并将实际运行结果与期望结果进行比较，以发现程序中的错误或异常
- 为了达到对程序运行轨迹实行追踪的目的，需要对源程序进行分析，并在分析基础上插入用于记录和显示程序执行轨迹的“探针”函数
- 其他用于提高软件质量的工具还有程序切片分析工具、源程序调试器和软件质量度量和评价工具等，其工作过程也都需要对源程序进行分析并做相应处理

自然语言处理

- 自然语言处理（Natural Language Processing）是语言学、计算机科学和人工智能的一个子领域，关注计算机与人类语言之间的交互，特别是如何对计算机进行编程以处理和分析大量自然语言数据
- 目标是一台能够“理解”文本内容的计算机，包括其中语言的上下文细微差别
- 可以准确提取文本中包含的信息和见解，并对文本本身进行分类和组织
- 挑战通常涉及语音识别、自然语言理解和自然语言生成
- BNF用于指定“上下文无关语法”，通常用于表示编程语言语法。一种语言的BNF规范是一组派生规则，它们在语法上共同验证程序代码

数据库查询优化

- 查询是数据库系统中最基本、最常用的一种操作。用户借助于SQL语句，可以完成对数据库的复杂查询
- **20世纪80年代末，IBM Almaden研究中心开发的可扩充数据库系统Star-burst首次采用了查询优化这一新的技术**
- **查询优化器在接收到一条SQL语句后，一般分六个步骤处理SQL语句**
  - (一)词法分析：对SQL语句的保留字进行词法分析并转换成内部码
  - (二)语法分析：对SQL语句进行语法分析和语义检查，将语句转换成适于内部处理的形式，即采用查询图的内部表示
  - (三)查询重写：优化器将查询语句重写成一种等价的且效率较高的形式，并为基于代价的优化提供更多的机会

数据库查询优化

- (四)计划优化

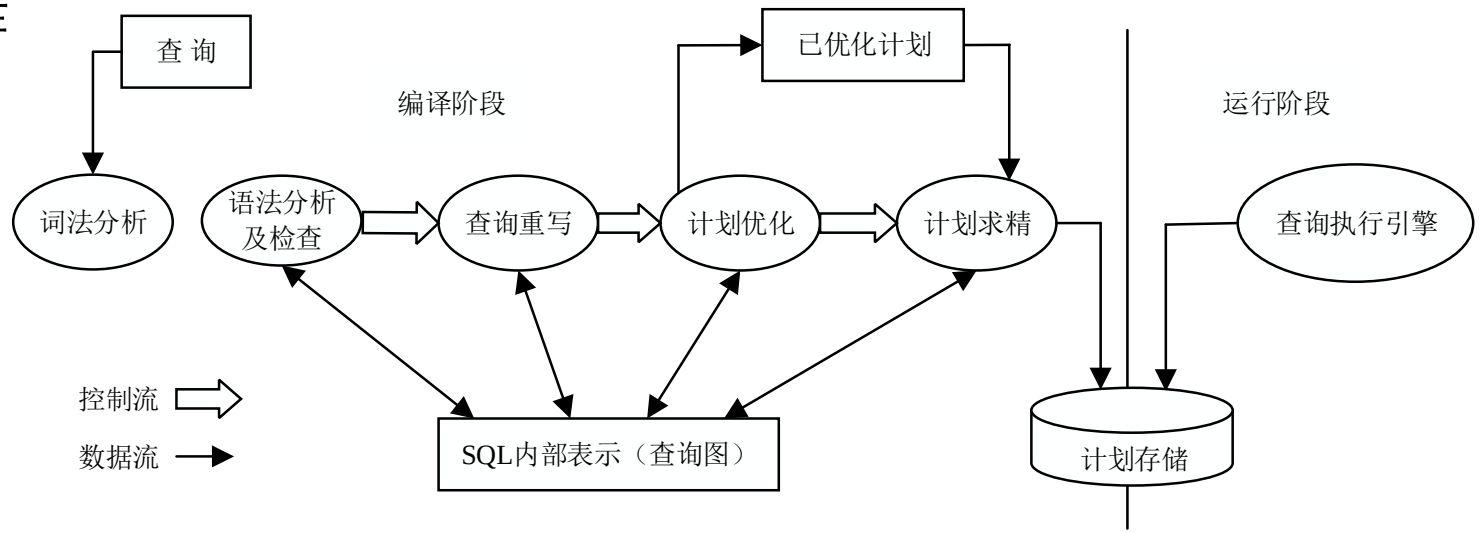
根据可用的存取路径生成一组可能的执行计划

- (五)计划求精

优化器根据表的数据分布和统计信息评估每一个执行计划代价，并选取代价最小的一个

- (六)查询执行引擎

根据执行计划执行具体的物理操作



《编译程序》的设计目标：

（1）生成尽量小的目标程序。
（2）目标程序运行速度尽量快。
（3）编译程序尽可能小。
（4）编译所花的时间尽可能少。
（5）有较强的查错和改正错误的能力。
（6）可靠性好。

小结

- 1.1 编译与解释
  - 任务相同，行为和输出不同
- 1.2 编译程序概述
  - 五个主要阶段，可以分多遍实现
- 1.3 编译程序的结构
  - 分析与综合
- 1.4 编译程序的设计与实现
  - 自展和移植
- 1.5 编译技术应用

阅读材料：
《程序设计语言编译原理（第3版）》第一章
《编译原理与技术》第一章