编译原理

武汉大学计算机学院

编译原理课程组

与课程有关的问题

- 时间安排：　　讲课：54学时　　实习：18学时

- 教材和参考书：

《Compilers Principles, Techniques and Tools》，《编译原理》本科教学版，机械工业出版社

《编译原理》，何炎祥，华中科技大学出版社

《编译原理与实践》Compiler Construction: Principles and Practice, Kenneth C.Louden，机械工业出版社

《程序设计语言编译原理》，陈火旺等，国防工业出版社

- 作业：每次视所讲内容布置1-2道习题

- 成绩：以考试为主（70%），参考平时成绩(作业、实习等)（30%）

与课程有关的问题

·本课程的性质、目的和任务：

本课程是计算机类专业的专业课，目的是使学生了解并掌握编译程序的基本理论和方法，具有分析和实现编译程序的初步能力。

·本课程的基本要求：

通过对本课程的学习，对形式语言有初步了解，并能对编译程序的整个结构有较清楚地了解，熟悉和掌握几种主要编译方法。

·课程内容的重点、深度与广度：

文法和形式语言、词法分析和有穷自动机、语法分析、语义分析、目标代码的生成，此外还要求掌握和了解符号表的构造、存储分配与管理、代码优化和错误校正。

课程简介

- 前导课程：高等数学、离散数学、高级程序设计语言（C/PASCAL）、汇编语言、数据结构
- 《编译原理》的应用领域
  - 编译器的设计
  - 一般的软件设计
    - 文本编辑器、信息检索系统、模式识别器
    - 排版、绘图系统
    - 程序验证器

国外高校的相关课程

- 课程的层次设计
  - Stanford (Jeffrey Ullman、Monica Lam)
    - CS143: Compilers
    - CS243: Advanced Compiling Techniques
  - UC Berkeley
    - CS164: Programming Languages & Compilers
    - CS264: Implementation of Programming Languages
  - UC Irvine
    - CS/E142: Compilers & Interpreters
    - CS/E141: Programming Languages

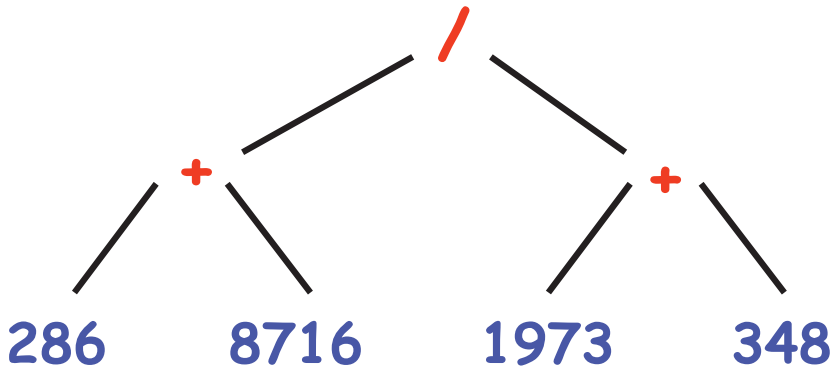
为何要学《编译原理》？

- ？只是一门“脑保健操”课程
- ？毕业后不会从事编译器的开发工作

- 语言与语言处理器
- ！语言是求解问题的一种有效途径
  - Language is an alternative approach to problem solving.

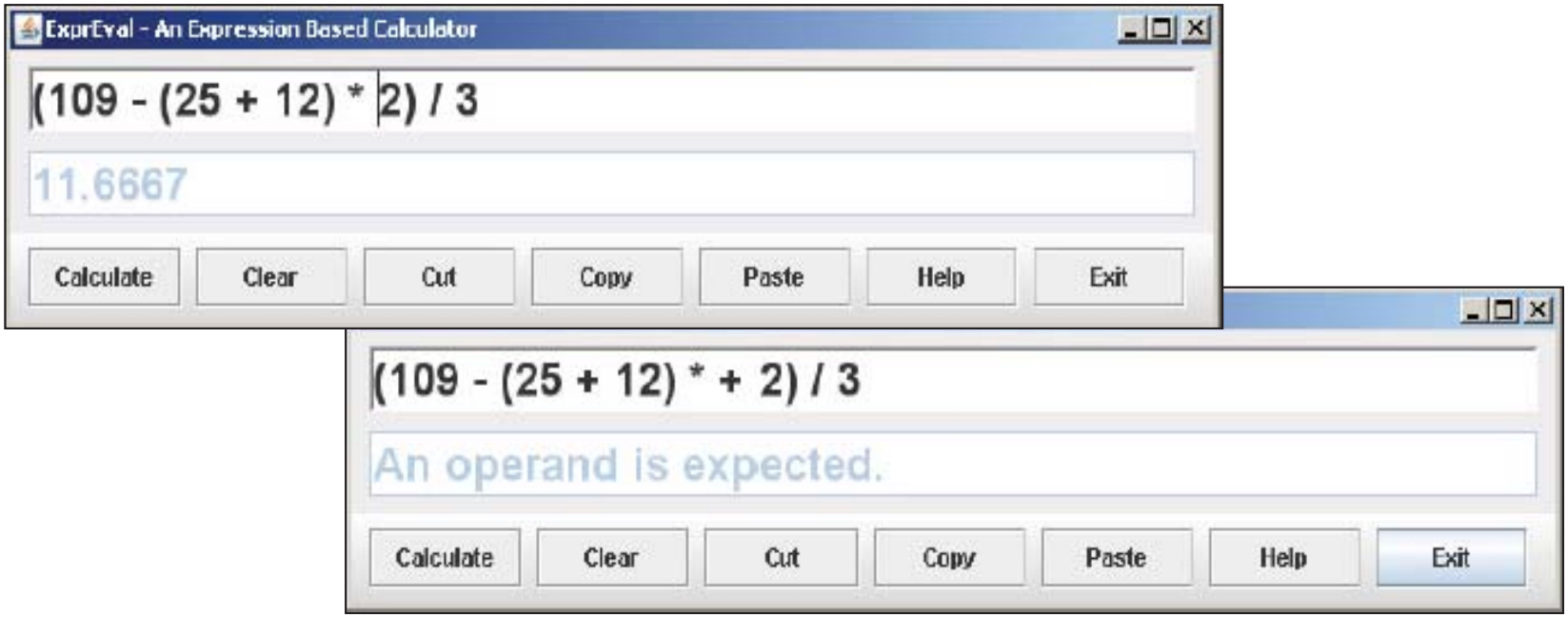
一个小游戏

- 利用Windows计算器计算 (鼠标操纵GUI)
  - 5 + (8 - 2)
  - (286 + 8716) / (1973 + 348)



用语言来求解问题！

- 将表达式从键盘输入计算器
  - 语言：表达式
  - 语言处理器：接受表达式的计算器



计算机“语言”及其“处理”

- Language
  - Programming languages
    - Scripts
  - Domain-specific languages
    - SQL, HTML, XML, PostScript/PDF/LaTex, HDL, etc.
    - Report, workflow, music, recitation, etc.
- Processing
  - Compiling
  - Beautifier, complexity evaluation, structured editor, reverse engineering, etc.

学习中应注意的问题

- 一定要预习
- 上课专心致志
- 重视习题
- 确立好学习的角度
- 重视实习

急功近利是学习的一大敌人!

课程内容

- 第1章：编译程序概述
- 第2章：文法和语言的形式定义
- 第3章：有穷自动机
- 第4章：词法分析
- 第5—7章：语法分析
- 第8章：语义分析和中间代码生成
- 第9章：运行阶段的存储组织与分配
- 第10章：符号表
- 第11章：中间代码优化
- 第12章：目标代码生成

第1章 引论

- 程序的翻译
- 编译程序的工作过程
- 编译程序的结构
- 编译程序的组织方式
- 编译程序的构造

第1章 引论

- 程序的翻译
- 编译程序的工作过程
- 编译程序的结构
- 编译程序的组织方式
- 编译程序的构造

1.1 程序的翻译

1.1.1 程序设计语言

- 机器语言　　001110010010
- 汇编语言　　ADD　R1　2
- 高级语言　　begin　x:=9+2　end

问题：

计算机只能识别二进制数0、1表示的指令和数构成的本计算机系统的机器语言。如何让计算机执行高级语言程序呢？

1.1 程序的翻译

1.1.2 翻译程序

· 所谓翻译程序是指这样一种程序，它能将用甲语言（源语言）编写的程序翻译成与之等价的用乙语言（目标语言）书写的程序。

· 程序的翻译通常有两种方式：一是“编译”方式，二是“解释”方式。

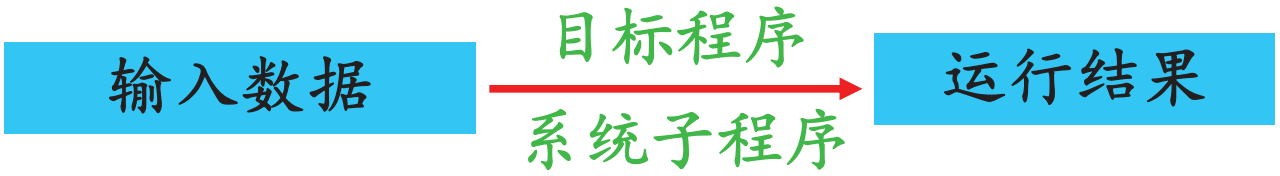
1.1 程序的翻译

1.1.3 编译方式

编译方式是一种分阶段进行的方式。

- 翻译阶段

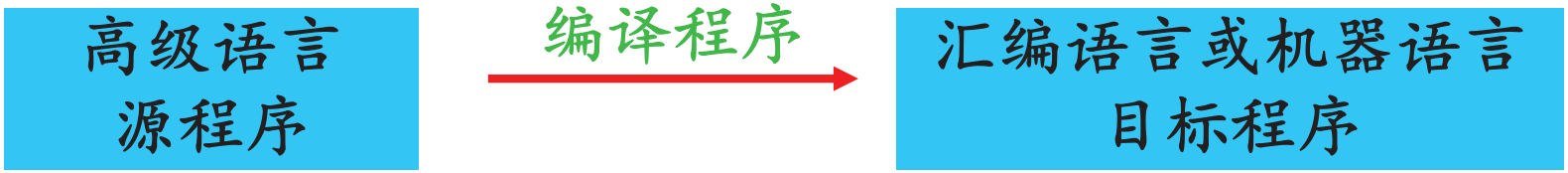
- 运行阶段



1.1 程序的翻译——编译方式

1. 编译方式下的翻译程序

- 编译程序

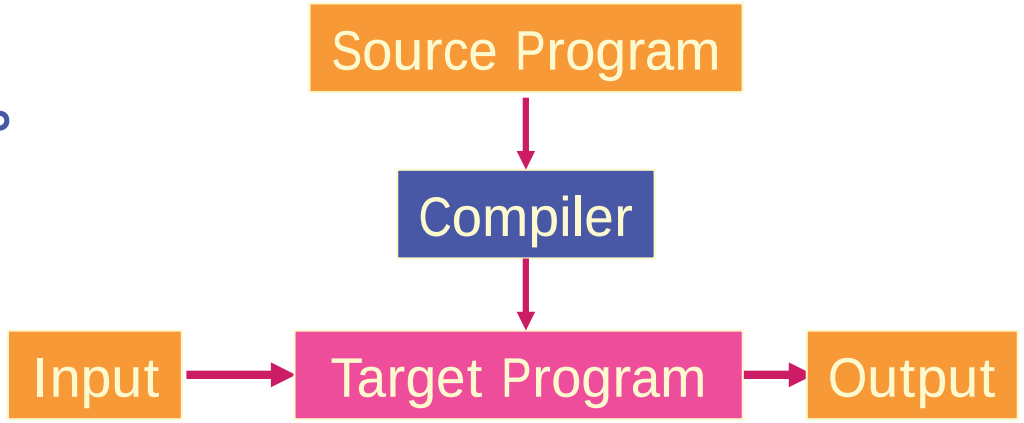


- 汇编程序

1.1 程序的翻译——编译方式

2. 编译方式的特点

(1) 源程序的执行需要分阶段。

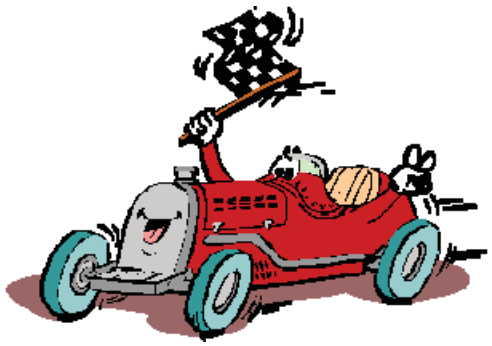


· 如果目标程序是机器语言程序，

两大阶段：编译阶段和运行阶段。

· 如果目标程序是汇编语言程序，

三大阶段：编译阶段、汇编阶段和运行阶段。



(2) 生成了目标代码，且可多次执行。

关于编译程序的几点说明

(1)编译程序生成的目标程序不一定是机器语言的程序，也有可能是汇编语言程序；

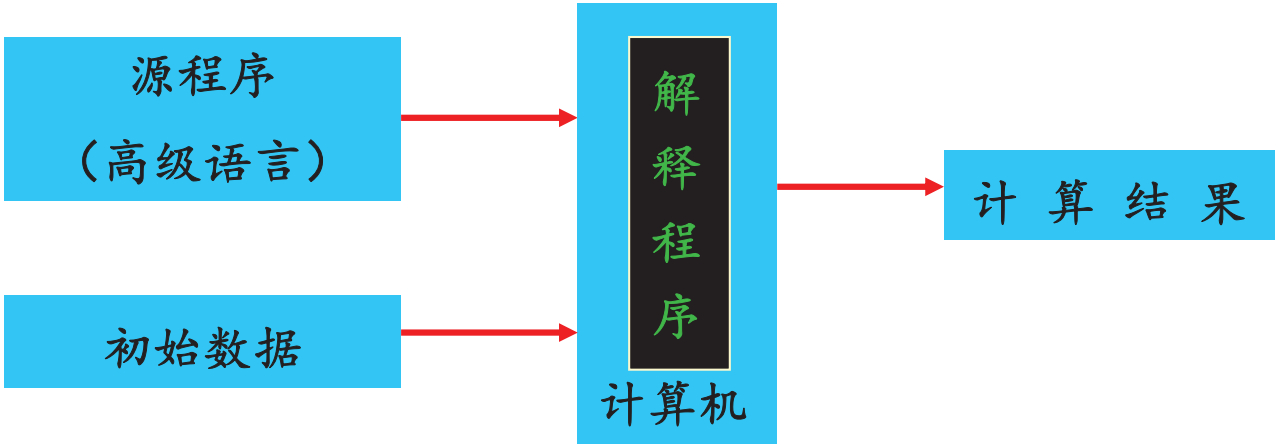
(2)编译程序与具体的机器和语言有关，即任何一个具体的编译程序都是某一特定类型的计算机系统中关于某一特定语言的编译程序；

(3)对编译程序而言，源程序是输入数据，目标程序是输出结果。

1.1 程序的翻译

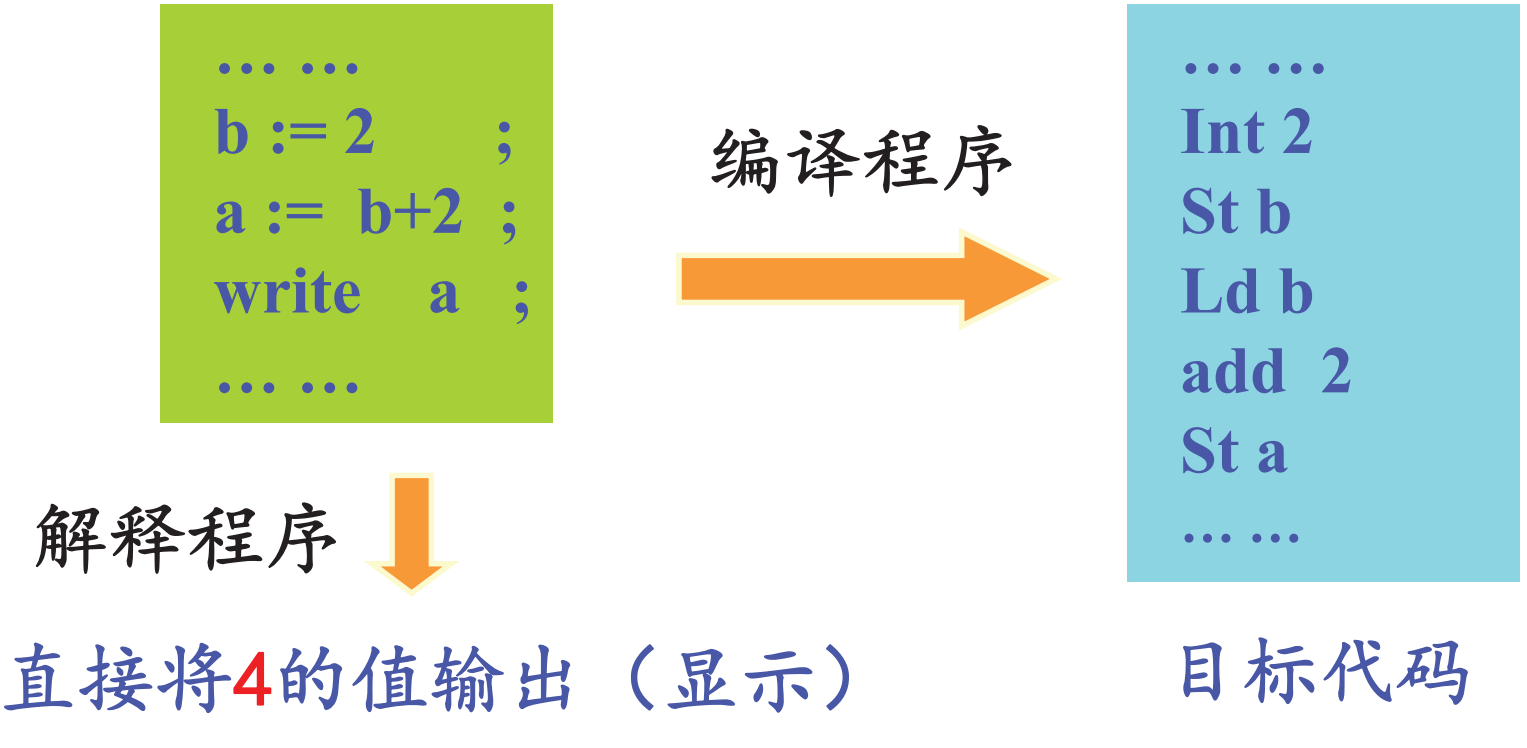
1.1.4 解释方式

完成解释工作的解释程序将按源程序中语句的动态顺序，逐句地进行分析解释，并立即予以执行。



源程序解释执行的历程

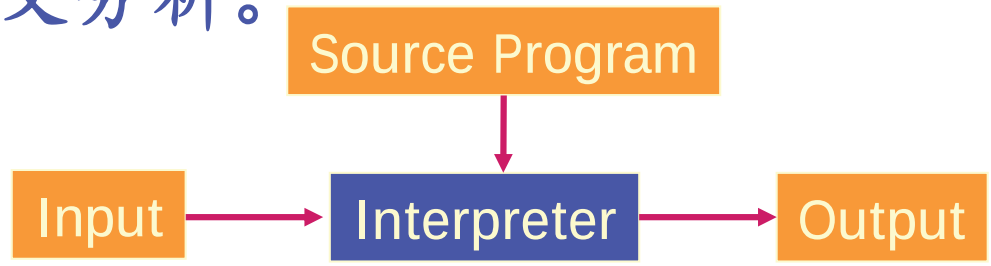
- 直接对源程序中的语句进行分析，执行其隐含的操作。
- 解释执行是按照被解释的源程序的逻辑流程进行工作的。



解释方式的特点

在解释方式下，并不生成目标代码，而是直接执行源程序本身。这是编译方式与解释方式的根本区别。

都需进行词法、语法和语义分析。

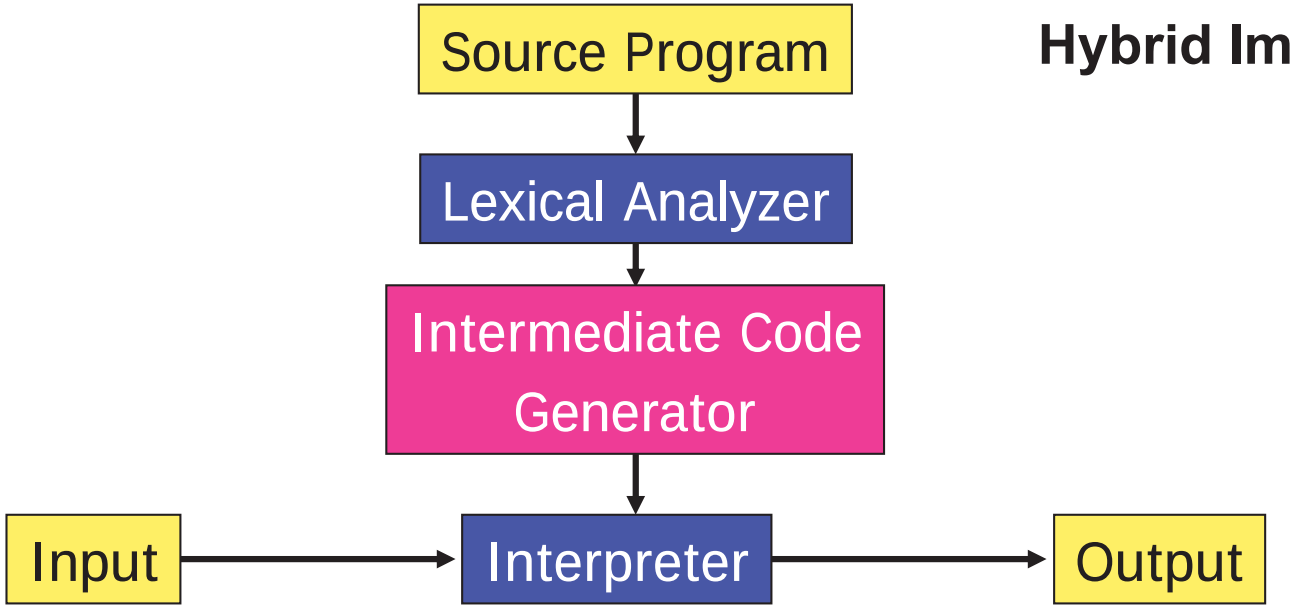


特点：

- 更灵活，交互方便，节省空间
- 效率低（时间开销、空间开销）

课下思考总结

- Low-level language —— High-level Language
- Compiler（Assembler）—— Interpreter



Hybrid Implementation Systems

第1章 引论

- 程序的翻译
- 编译程序的工作过程
- 编译程序的结构
- 编译程序的组织方式
- 编译程序的构造

1.2 编译程序的工作过程

- 词法分析
- 语法分析
- 语义分析和中间代码生成
- 中间代码优化
- 目标代码生成

1.2 编译程序的工作过程——词法分析

依据语言词法规则，分析由字符组成的源程序，把它识别为一个一个具有独立意义的最小语法单位，即“单词”，并识别出与其相关的属性(如是标识符，是界限符，还是数，等等)，再转换成长度上统一的标准形式(这种统一的标准形式既刻画了单词本身，又刻画了它所具有的属性，称为属性字)，以供其它部分使用。

编译过程——词法分析

- 扫描源程序的字符串，识别单词（关键字、标识符、常量、运算符、界限符）

例： position := initial + rate * 60;

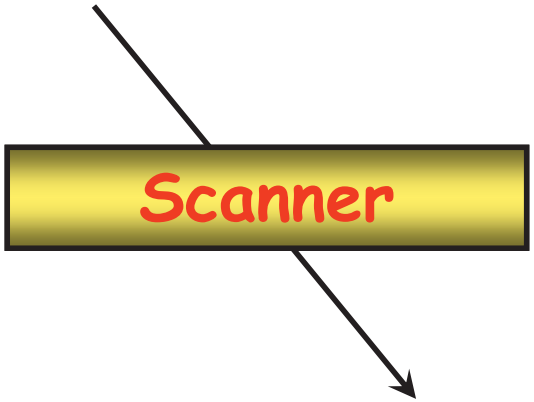
属性字：<类别号，自身值>
class value

单词类型	单词值
标识符1(id1)	position
运算符(赋值)	:=
标识符2(id2)	initial
运算符(加)	+
标识符3(id3)	rate
运算符(乘)	*
整常数	60
界限符	;

编译过程——词法分析

属性字：<类别号，自身值>
class value

position := initial + rate * 60



符号表

	NAME	TYPE	VAL	ADDR	…
1	position	…			
2	initial	…			
3	rate	…			
4	60	…			

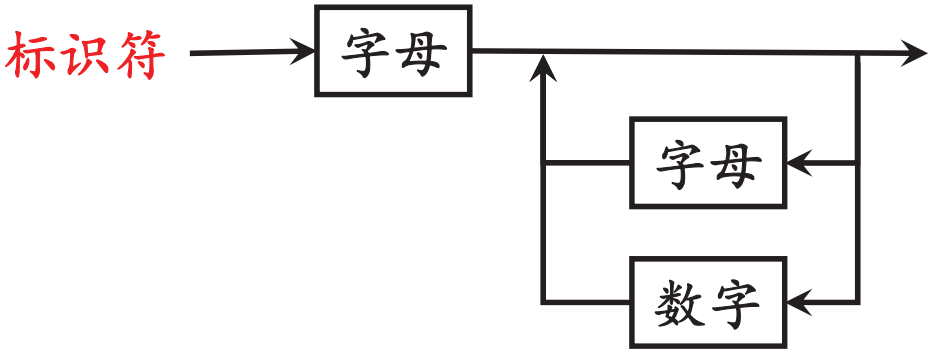
<id,1> <:=> <id,2> <+> <id,3> <*> <number,4>

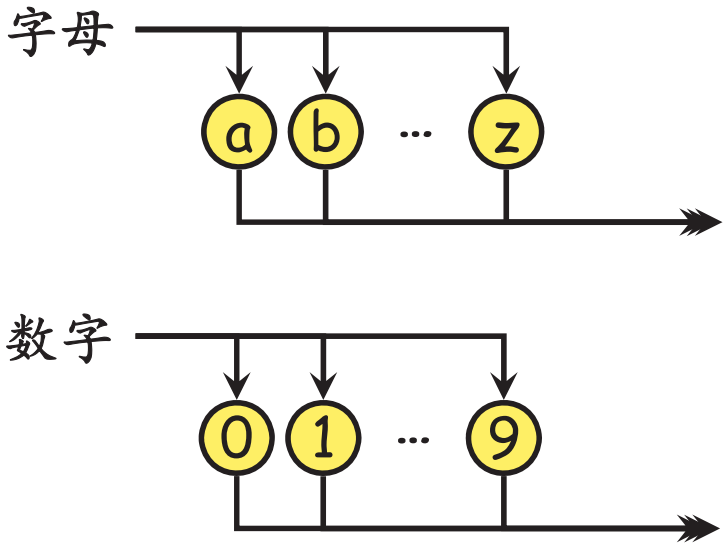
1.2 编译程序的工作过程 ——语法分析

依据语法规则，逐一分析词法分析时得到的单词，把单词串分解成各类语法单位，即确定它们是怎样组成说明和语句，以及说明和语句又是怎样组成程序的。分析时如发现有不合语法规则的地方，便将出错的位置及出错性质打印报告给程序员；如无语法错误，则用另一种中间形式给出正确的语法结构，供下一阶段分析使用。

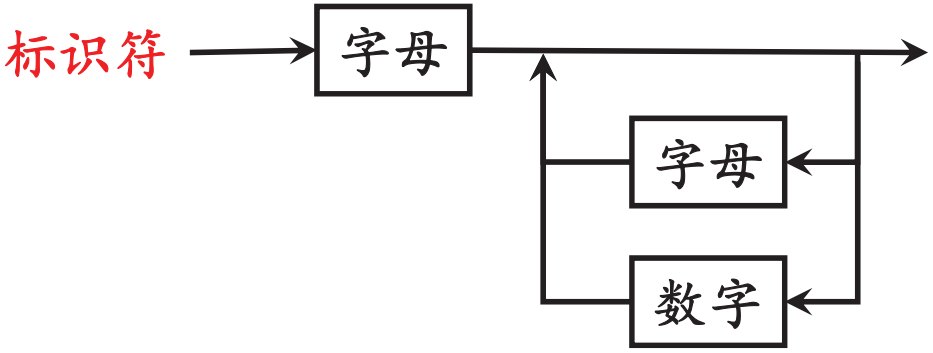
语法描述

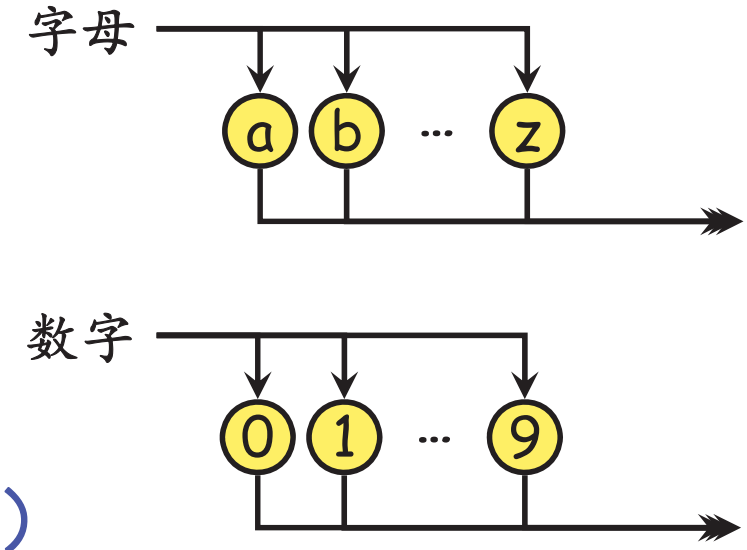
- 自然语言
  - 如：<标识符>
  - “是由字母后跟若干个(≥0)字母或数字的符号串组成”
- 语法图（Syntax Graph）





语法描述





- BNF范式（Backus-Naur Form）

  <标识符> ::= <字母>|<标识符><字母>|<标识符><数字>

- 扩充的BNF（EBNF：Extended BNF）

  <标识符> ::= <字母> {<字母>|<数字>}

编译过程——语法分析

例： position := initial + rate * 60

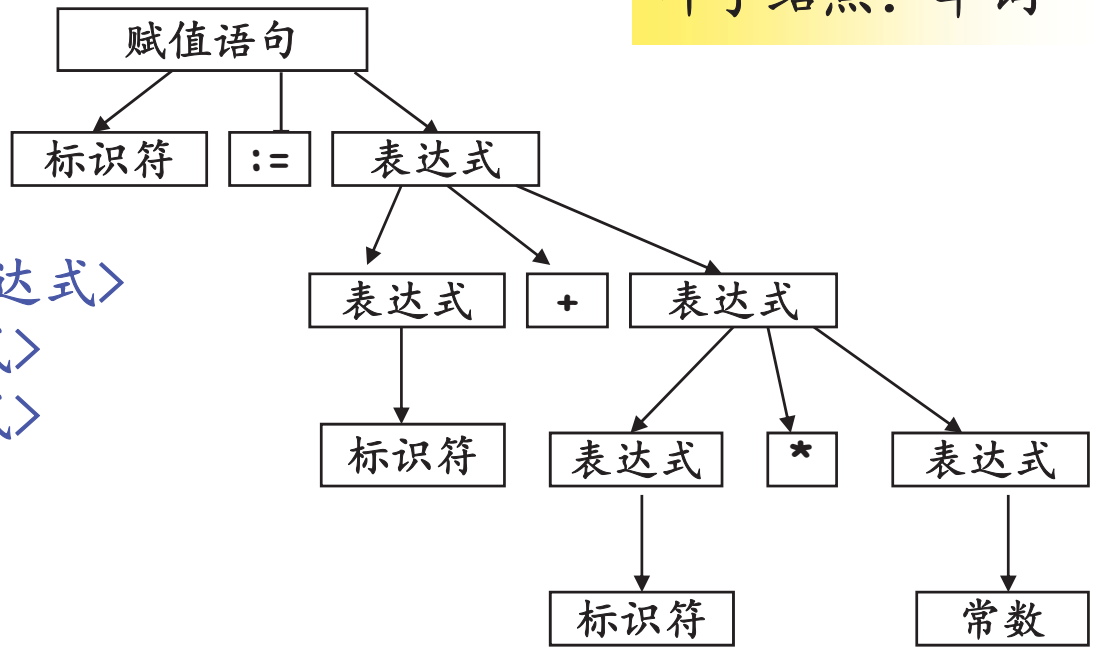
显示程序语法

内部结点：结构名

叶子结点：单词

语法规则：

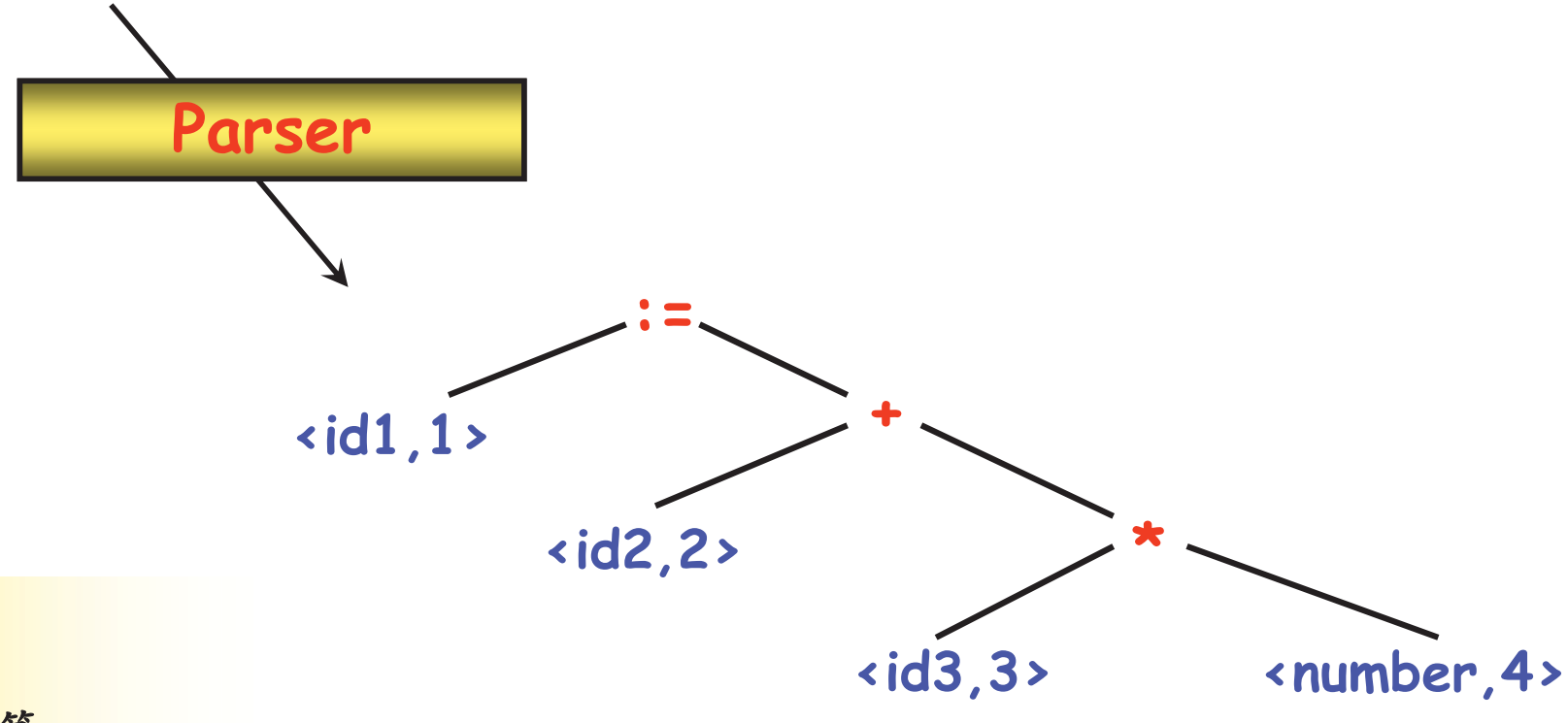
<赋值语句>::=<标识符>“:=”<表达式>
<表达式>::=<表达式>“+”<表达式>
<表达式>::=<表达式>“*”<表达式>
<表达式>::=“(”<表达式>“)”
<表达式>::=<标识符>
<表达式>::=<整数>
<表达式>::=<实数>



分析树（Parse tree）

编译过程——语法分析

<id,1> <:=> <id,2> <+> <id,3> <*> <number,4>



层次结构分析

内部结点：运算

其子结点：该运算的分量

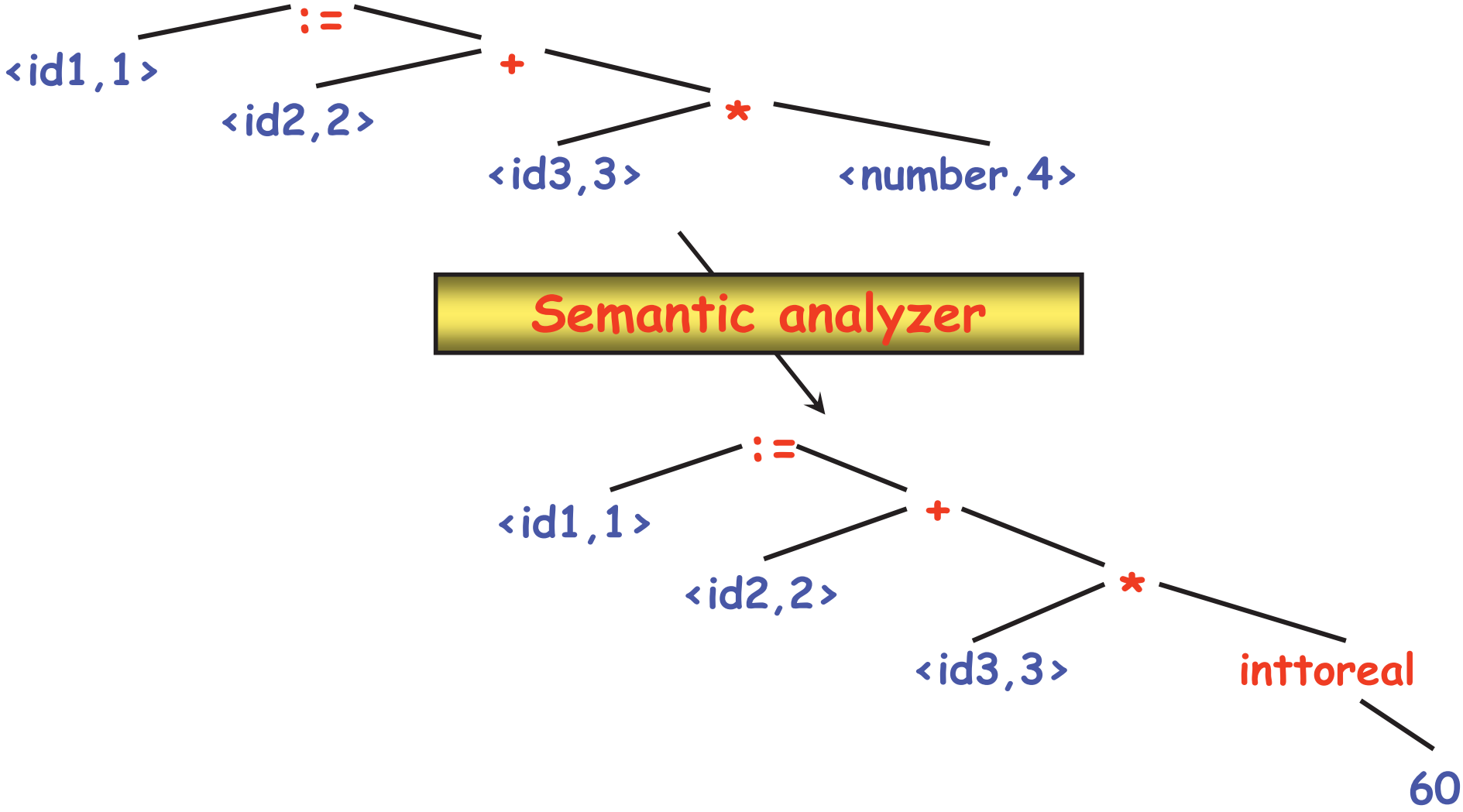
语法树（Syntax tree）

1.2 编译程序的工作过程 ——语义分析

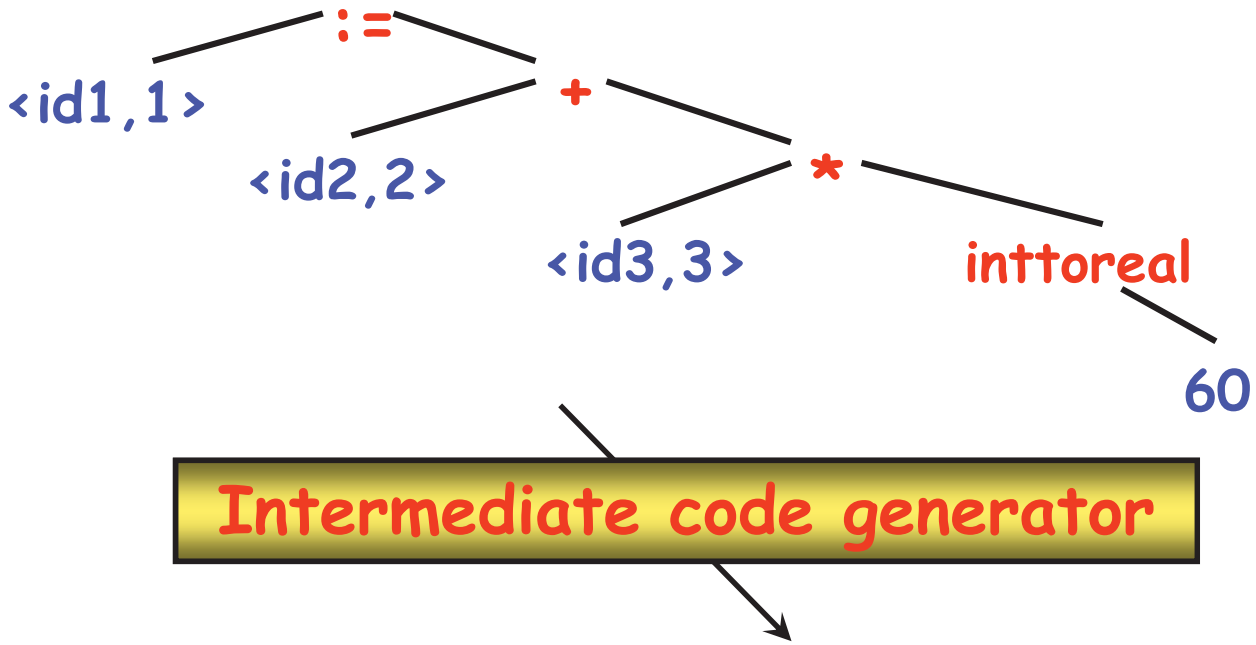
依据语言的语义规则对语法分析得到的语法结构进行静态语义检查(确定类型、类型和运算合法性检查、识别含义与相应的语义处理及其它一些静态语义检查)，并用另一种内部形式表示出来，或者直接用目标语言表示出来。

凡在编译时可以确定的内容称为“静态”的；凡必须推迟到程序运行时才能确定的内容称为“动态”的。

编译过程——语义分析


:=
<id1,1>
+
<id2,2>
*
<id3,3>
<number,4>
Semantic analyzer
:=
<id1,1>
+
<id2,2>
*
<id3,3>
inttoreal
60

编译过程——中间代码生成



Intermediate code generator

(1)	(inttoreal,	60	-	t1	)
(2)	(* ,	id3	t1	t2	)
(3)	(+ ,	id2	t2	t3	)
(4)	(:= ,	t3	-	id1	)

id1:= id2 + id3 * 60

四元式

1.2 编译程序的工作过程 ——代码优化

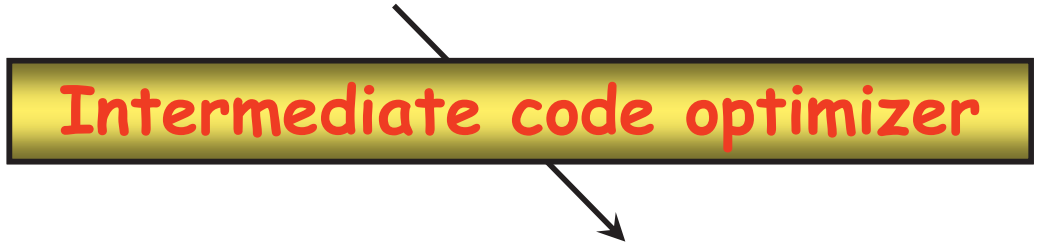
依据程序的等价变换规则，尽量压缩目标程序运行所需的时间和所占的存储空间，以提高目标程序的质量。

优化的是中间代码/目标代码的质量，而非编译程序的质量。

编译过程——代码优化

(1)	(inttoreal,	60	-	t1	)
(2)	(* ,	id3	t1	t2	)
(3)	(+ ,	id2	t2	t3	)
(4)	(:= ,	t3	-	id1	)

id1:= id2 + id3 * 60



(1)	(* ,	id3	60.0	t1	)
(2)	(+ ,	id2	t1	id1	)

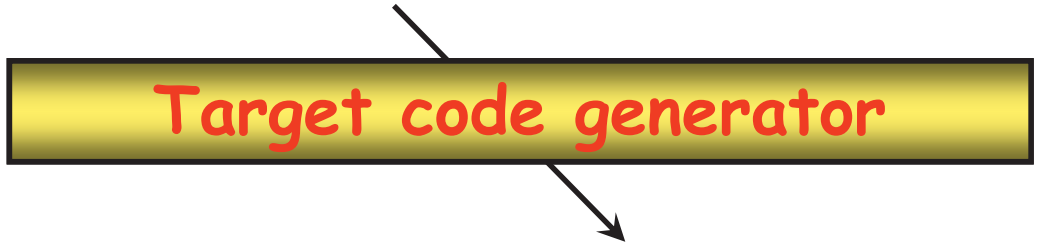
1.2 编译程序的工作过程 ——代码生成

如果语义分析时把源程序表示成中间形式而不是表示成目标指令，则由本部分完成从中间形式到目标指令的转换。如果语义分析时，已直接生成目标指令，则无需另外再做代码生成工作。

目标指令可能是绝对指令代码，或可重新定位的指令代码或汇编指令代码。该阶段的工作有赖于硬件系统结构和机器指令含义。

编译过程——代码生成

```
(1)    (*    ,    id3    60.0    t1    )
(2)    (+    ,    id2    t1      id1   )
```



id1:= id2 + id3 * 60

```
movf    id3,  R2
mulf    #60.0,  R2
movf    id2,  R1
addf    R2,  R1
movf    R1,  id1
```

1.2 编译程序的工作过程 ——表格管理

登记源程序中出现的每个名字以及名字的各种属性。有些名字的属性需要在各个阶段才能填入。

符号表

NAME	TYPE	CAT	…	VAL	ADDR
position	…				
initial	…				
rate	…				

1.2 编译程序的工作过程 ——出错处理

源程序中的错误有**语法错误**和**语义错误**两种。

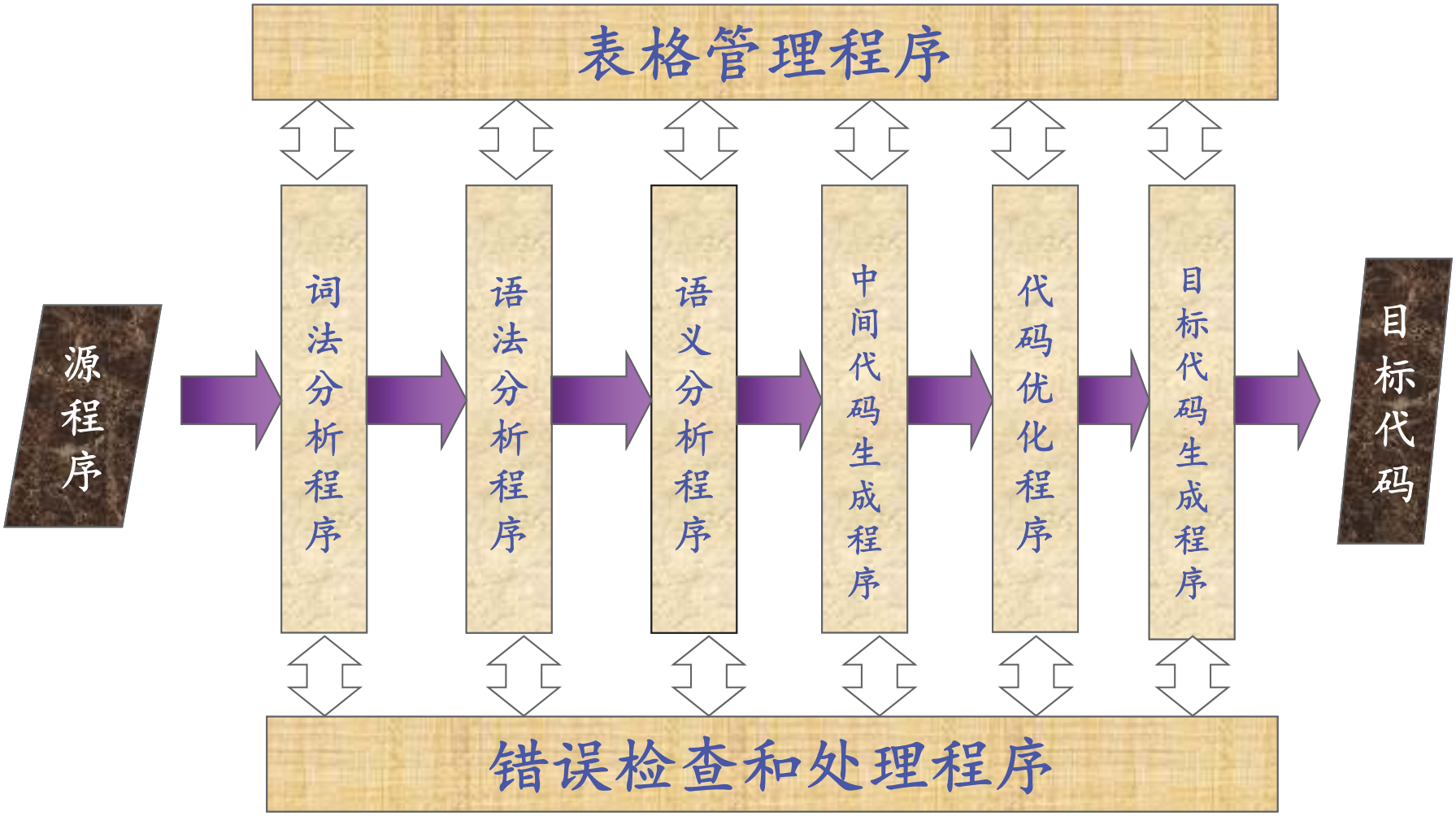
语法错误：源程序中不符合语法(或词法)规则的错误，它们可在词法分析或语法分析时检测出来。

语义错误：源程序中不符合语义规则的错误，一般在语义分析时检测出来，有的语义错误要在运行时才能检测出来。通常包括：说明错误、作用域错误、类型不一致等等。

第1章 引论

- 程序的翻译
- 编译程序的工作过程
- 编译程序的结构
- 编译程序的组织方式
- 编译程序的构造

1.3 编译程序的结构


表格管理程序
源程序
词法分析程序
语法分析程序
语义分析程序
中间代码生成程序
代码优化程序
目标代码生成程序
目标代码
错误检查和处理程序

课程内容

- 第1章：编译程序概述
- 第2章：文法和语言的形式定义
- 第3章：有穷自动机
- 第4章：词法分析
- 第5—7章：语法分析
- 第8章：语义分析和中间代码生成
- 第9章：运行阶段的存储组织与分配
- 第10章：符号表
- 第11章：中间代码优化
- 第12章：目标代码生成

第1章 引论

- 程序的翻译
- 编译程序的工作过程
- 编译程序的结构
- 编译程序的组织方式
- 编译程序的构造

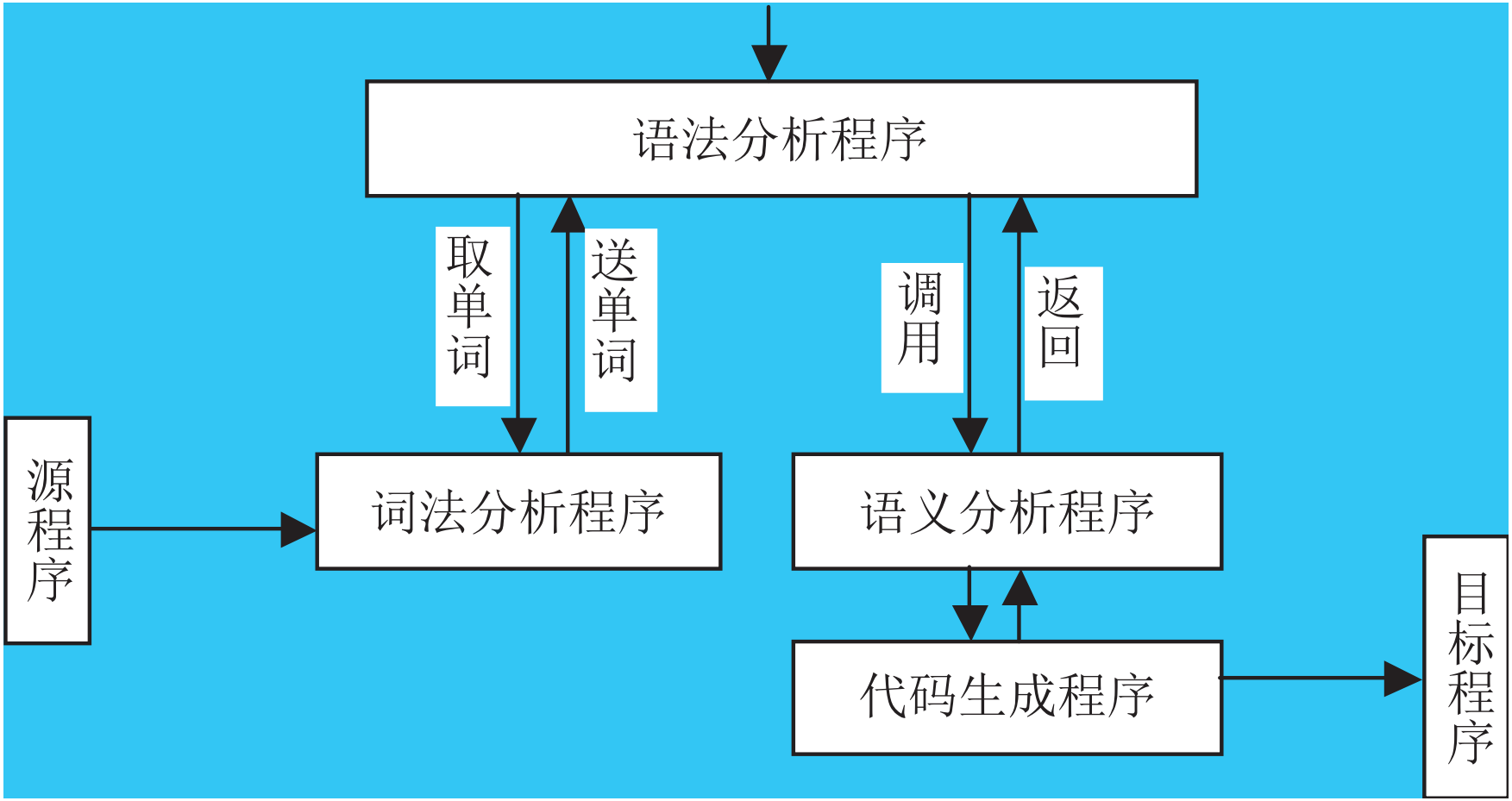
1.4 编译程序的组织形式

1.4.1 遍(趟，趟程)

所谓一趟或一遍是指一个编译程序在编译时刻把源程序或源程序的等价物(中间程序)从头到尾扫描一遍并转换成另一紧邻的等价物的全过程。

根据编译程序在完成翻译任务的过程中需要对源程序或其中间等价物扫描的遍数，可以把编译程序分为单遍扫描的编译程序(只需扫描一遍)和多遍扫描的编译程序(需扫描多遍)。

单遍扫描的编译程序



多遍的特点

- 优点
  - 节省内存空间
  - 提高目标程序质量
  - 缩短Compiler的开发周期
- 缺点
  - 重复性工作
  - 延长了编译时间，降低了编译效率

单遍——多遍

- 并非单遍/多遍一定就好，应视具体情况而定
- 选择编译的“遍”数的原则
  - 语言的大小、结构
  - 机器的规模
  - 设计的目的（如：编译速度、目标程序的运行速度）
  - 设计人员素质多少

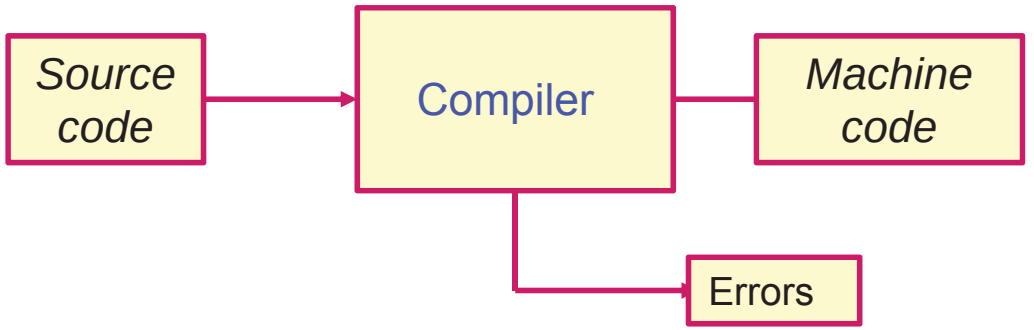
1.4 编译程序的组织形式

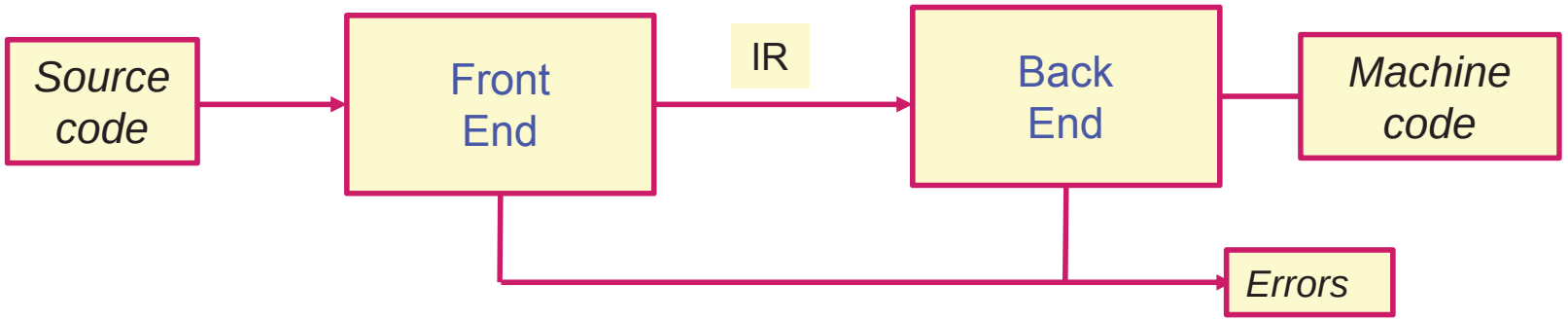
1.4.2 编译的前端和后端

前端主要由与源语言有关但与目标机器无关的那些部分组成，如词法分析、语法分析、语义分析与中间代码生成及部分代码优化工作。

后端主要包括编译中与目标机器有关的那些部分，如与目标机有关的代码优化和目标代码生成等。后端不依赖于源语言而仅依赖于中间语言。

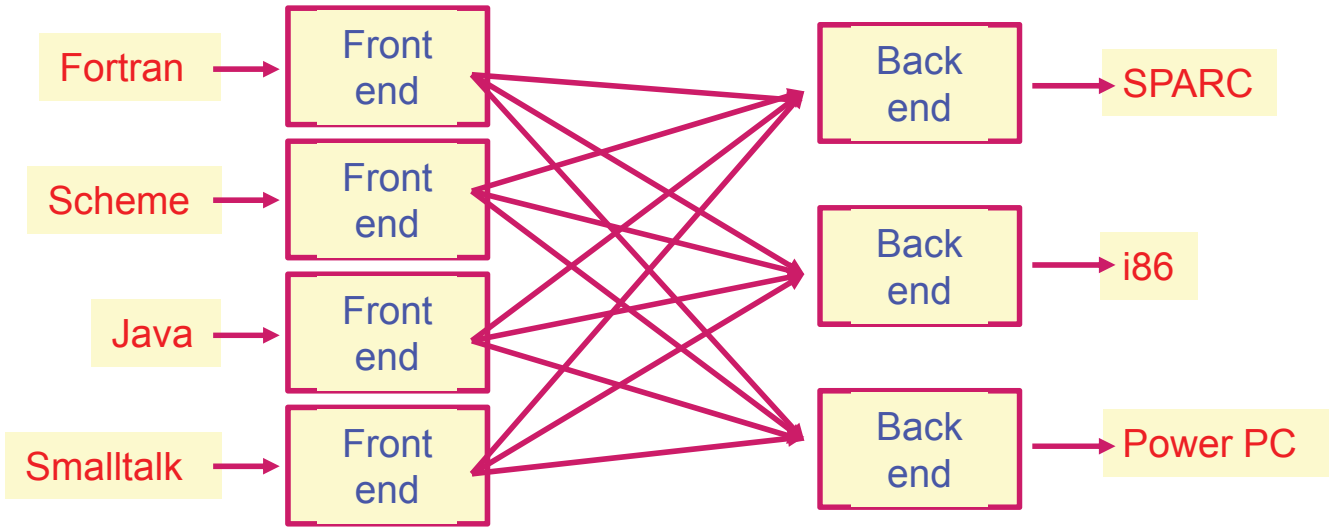
前端和后端


Source code
Compiler
Machine code
Errors


Source code
Front End
IR
Back End
Machine code
Errors

用于Compiler的移植

可以通过改变编译程序的后端来实现编译程序的移植。



Can we build *n x m* compilers with *n+m* components?

第1章 引论

- 程序的翻译
- 编译程序的工作过程
- 编译程序的结构
- 编译程序的组织方式
- 编译程序的构造

1.5 编译程序的构造

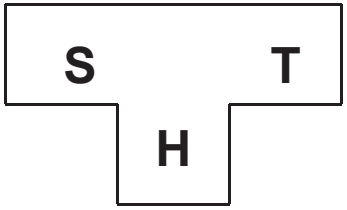
1.5.1 高级语言的自编译性

构造编译程序可以用机器言语、汇编语言和高级语言。

高级语言的自编译性：一个语言可以用来编写自己的编译程序。

1.5 编译程序的构造

T型图

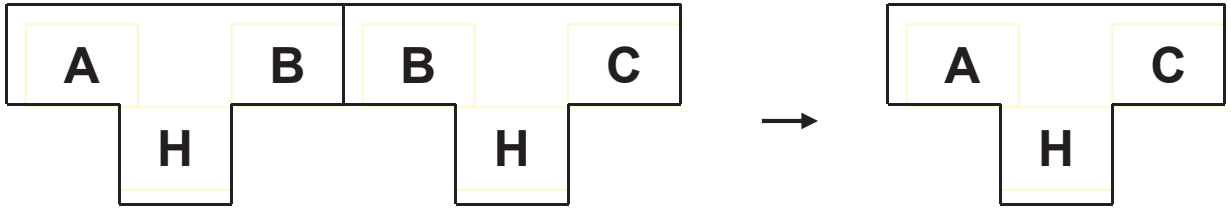


H：宿主语言

S：源语言

T：目标语言

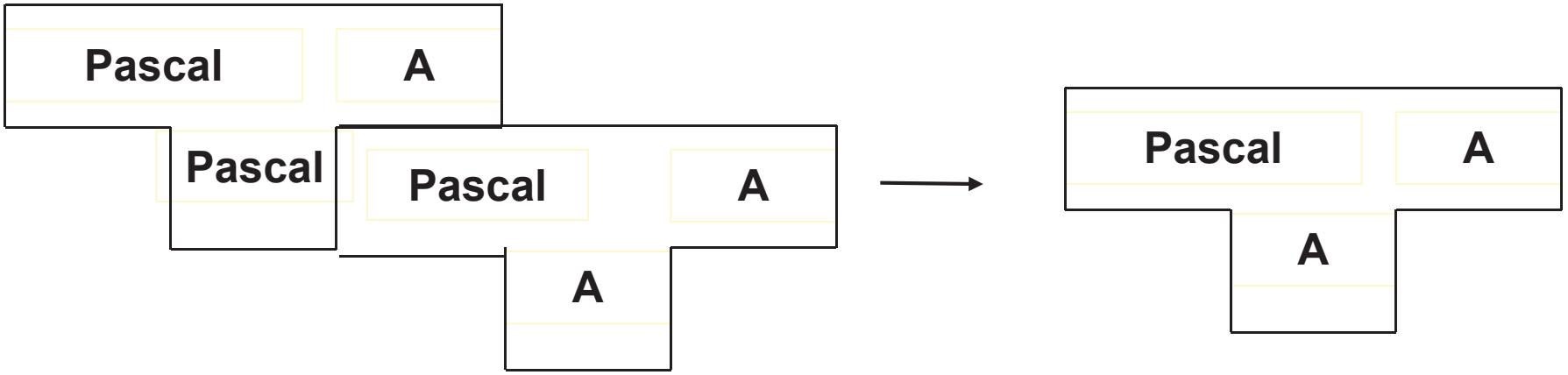
T型图的一种组合



1.5 编译程序的构造

自编译

[例] Pascal语言的编译程序



1.5 编译程序的构造

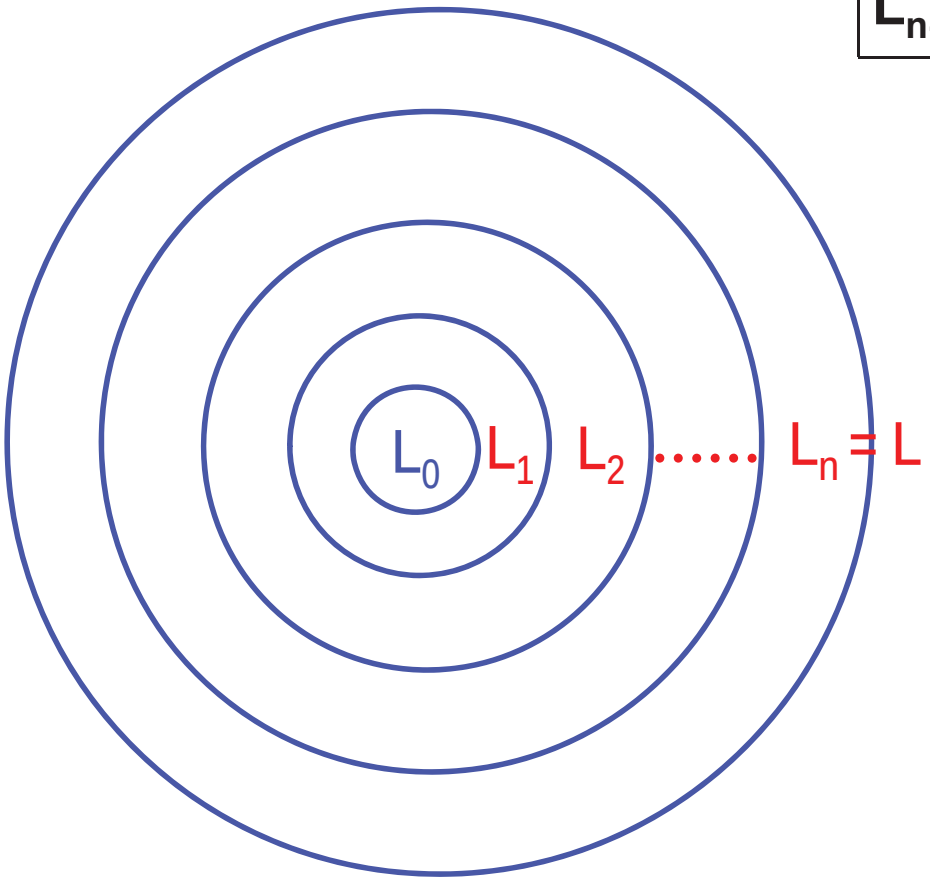
1.5.2 编译的自展技术

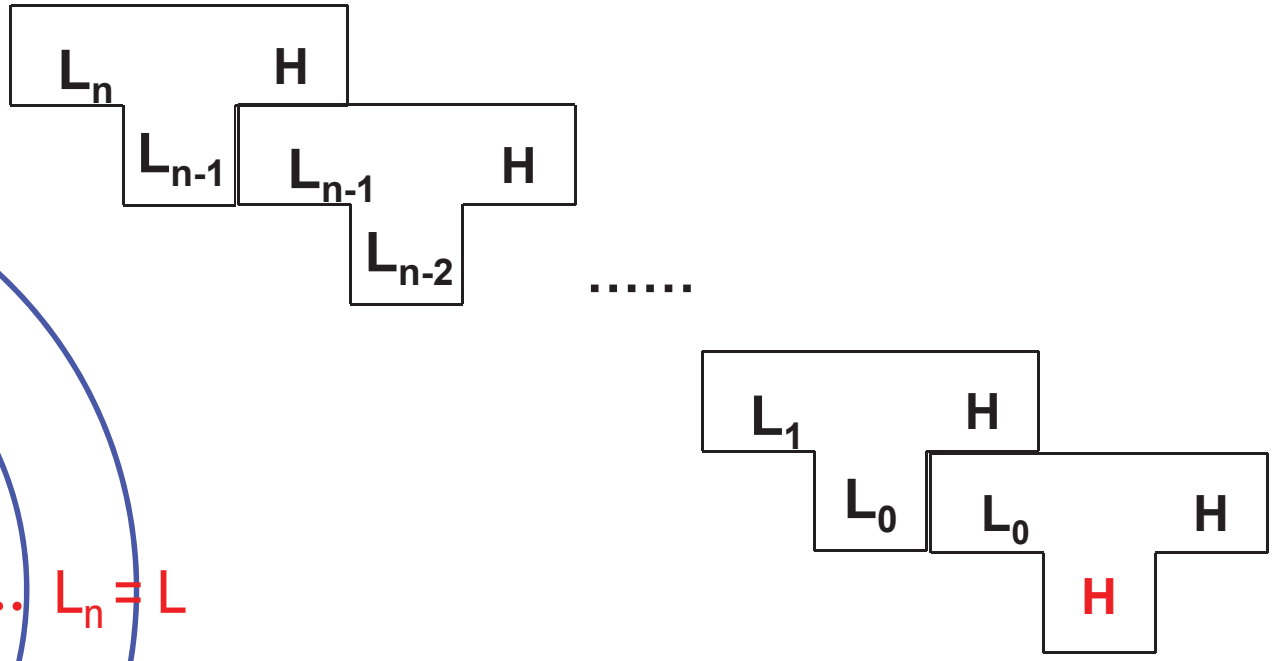
——通过一系列自展途径而形成编译程序的过程。

先对语言的核心部分构造一个小小编译程序(可用低级语言实现)，再以它为工具构造一个能够编译更多语言成分的较大编译程序。如此扩展下去，越滚越大，最后形成所期望的整个编译程序。

1.5 编译程序的构造

自展





1.5 编译程序的构造

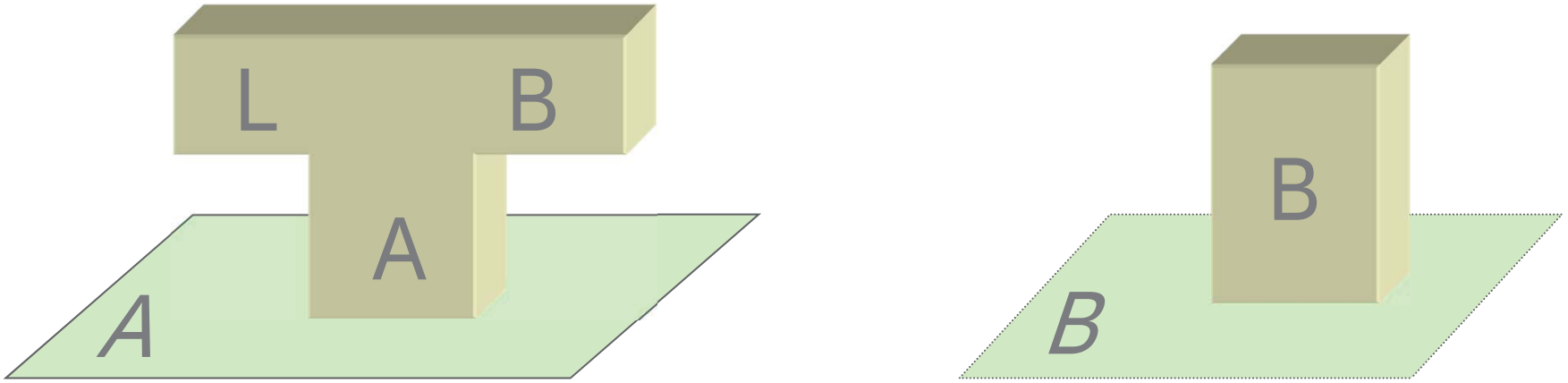
1.5.3 编译的移植

——将一个机器（宿主机）上的一个具有自编译性的高级语言编译程序移植到另一个机器（目标机）上。

利用A机器上的高级语言L编写能在B机器上运行的高级语言L的编译程序。

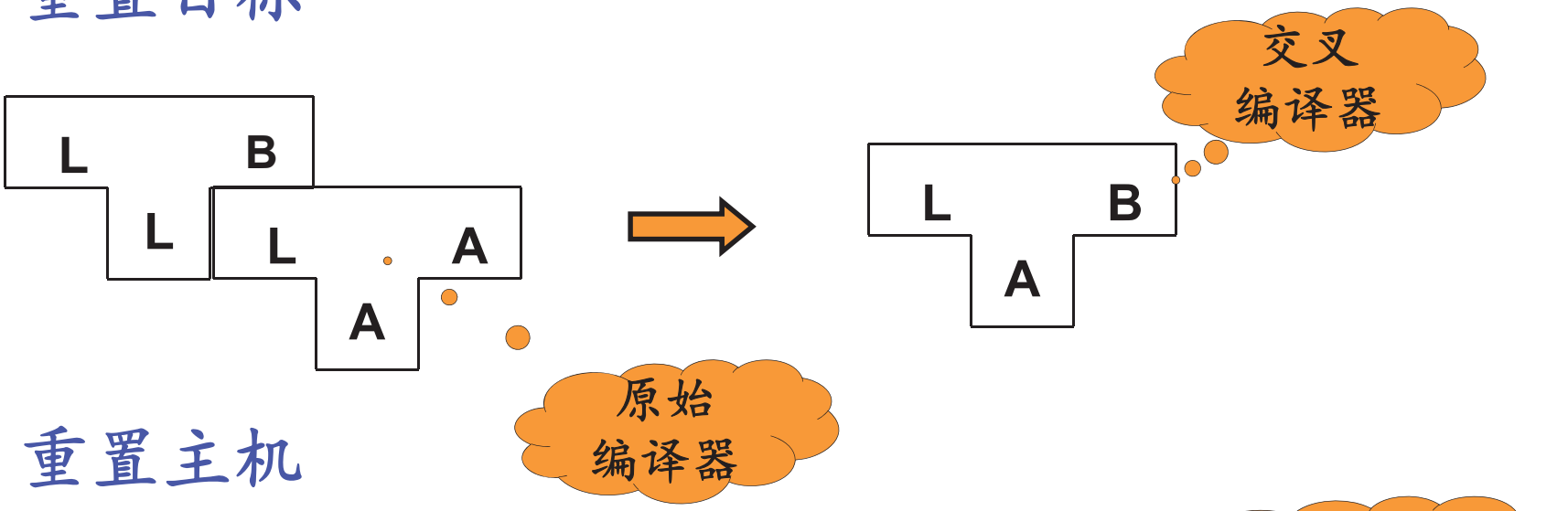
交叉编译器 （cross-compiler）

由于目标机指令系统与宿主机的指令系统不同，编译时将应用程序的源程序在宿主机上生成目标机代码，称为交叉编译。

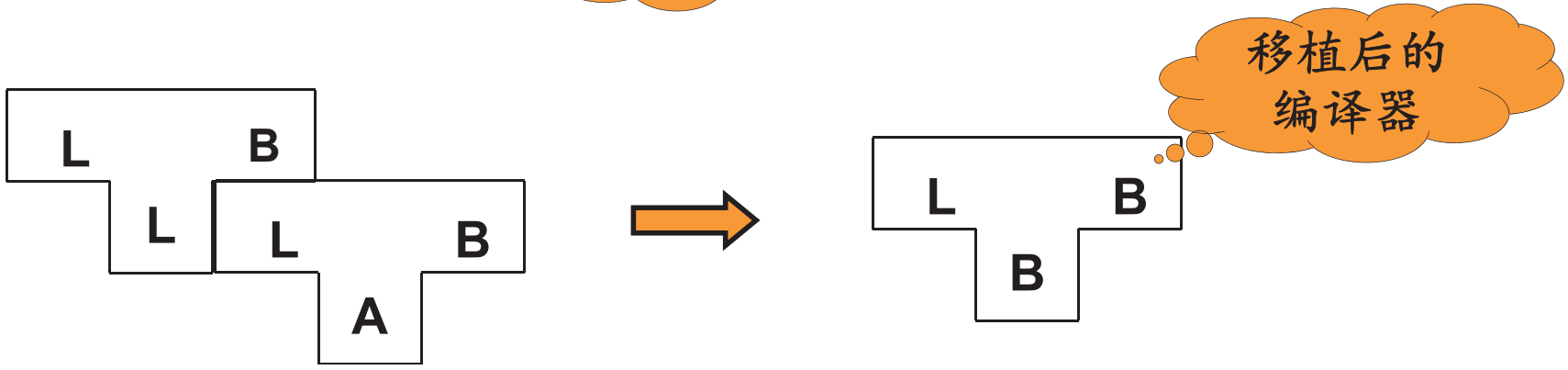


1.5 编译程序的构造

（1）重置目标



（2）重置主机



1.5 编译程序的构造

1.5.4 编译程序的自动化

第1章 内容小结

- 什么是编译程序
- 编译方式的特点
- 解释方式的特点
- 编译方式与解释方式的根本区别
- 编译程序的工作过程
- 编译程序的结构
- 遍与编译程序的组织形式
- 编译程序的构造方法

思考题

1. 分别指出编译程序、汇编程序、解释程序的含义。
2. “解释方式与编译方式的区别在于解释程序对源程序没有真正进行翻译”，这种说法对吗？
3. 编译程序的开发过程大致分几个阶段？任务如何？
4. 有人认为编译程序的五个组成部分缺一不可，对吗？
5. “多遍扫描的编译程序是高质量的编译程序，优于单遍扫描的编译程序”，对吗？

下章内容简介

- 文法的形式定义与文法分类
- 语言的形式定义
- 为语言构造文法
- 与语法分析有关的概念
- 文法的实用限制