编译原理

- 授课老师：徐 金安
- 电话：13716186689 51688451
- 邮箱：xja2010@gmail.com
- 办公室：9教北511
- 教室：思西504（周一、三）

北京交通大学计算机学院

学习中注意的问题

- 本课程共64学时
  - 48学时理论课程（作业、考试）
  - 16学时实践课程（6周后安排，4-6个设计）
- 在课程学习中体会
  - 抽象问题 、形式化描述、系统的观点、语言的实现机制
  - 原理→技术→实现

什么是编译程序？

用户： 要用basic, pascal ,c,java……　　高级语言

计算机： 只识别二进制表示的指令和数据　　低级语言

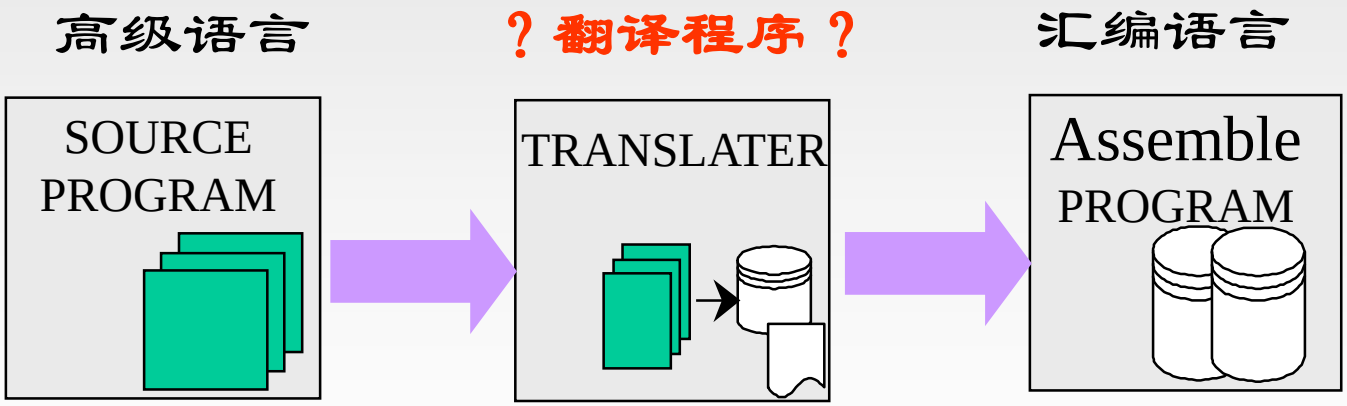
同类课程

- 编译原理：实现编译过程的基本原理，即把源程序变成目标程序的理论基础，比较抽象；
- 编译方法：是讲实现编译过程的一些手段，即怎样把源程序变成目标程序，比较具体；
- 编译技术：介于方法和原理之间，在实际实现时，编译过程的实现往往与其编译原理不能完全吻合，因机器的构造不完全相同。

为什么要学习编译原理

- 程序设计语言是计算机软件专业的重要核心
- 学习编程的历程：

  –C语言－－汇编语言－－数据结构



为什么要学习编译原理

- 必修主干课程，操作系统和编译系统构成程序设计者与计算机之间的基本界面。
- 通过学习该课程，掌握编译的基本理论、常用的编译技术，了解编译过程及编译系统结构和机理。能运用所学技术解决实际问题，能独立编写一个小型编译系统。
- 此外，通过学习编译原理可以更好地理解程序语言的内部机制，从而更好地理解和运用程序设计语言。能运用编译程序构造的原理和技术完成相关软件工具的设计和开发工作。

为什么要学习编译原理

- 计算机软件学科理论与实践相结合的典范。
- 在学习过程中既要注重该领域在理论上取得的完美结论，也要注重这些理论在实际中的应用。

先修课程

- 要求先学习以下课程
  - 1. 程序设计语言
  - 2. 算法与数据结构：栈分配、堆分配、静态分配等各种存储分配方式。线性表、二叉查找树、哈希表等多种数据结构。
  - 3. 离散数学：集合论与数理逻辑是进一步学习形式语言与自动机理论的数学基础。
- 最好学习过或同时学习以下课程
  - 1. 软件工程：掌握大型程序设计以及工程化的软件生产方法。
  - 2. 形式语言与自动机：相当于本课程中词法分析与语法分析的理论基础。

- 本课的重点在原理及相关技术方面；
- 编译课程蕴涵着计算机科学中解决问题的思路、抽象问题和解决问题的方法；

  问题的由来？问题的解决？

- 本课程可体会理论与实践的结合，理论如何指导实践；

  编译原理指导编译程序设计？

教材： 蒋立源等 编译原理 第3版 西北工大出版社

参考资料

1、陈火旺等　　程序设计语言编译原理

国防工业出版社

2、吕映芝等　　编译原理　　清华大学出版社

3、(美)Kenneth C.Louden著　冯博琴等译

编译原理与实践　　机械工业出版社

4、陈意云　编译原理　　高等教育出版社

5、 Alfred V.Aho, Compilers:Principles Techniques,and Tools　　人民邮电出版社

6、何炎祥等 编译原理　　机械工业出版社

要求及学习方法

课程特点：理论性强，算法复杂

- 平时（10%）
  - 无故旷课：扣分
  - 一本教材，认真听课
  - 认真完成课堂和课后作业
  - 完成要求的实验内容
  - 平时习题课及小测验
- 实验（20%）： 6个设计 （多做加分）
- 期末（70%）： 闭卷

重要信息

- 1. 课件下载：

  bianyibjtu@126.com

  密码： bianyi2017
- 2. 作业提交（仅收电子版）

  zuoyebjtu@126.com

第1章　编译概述

教学目标

1. 掌握编译程序中所涉及的有关名词术语
2. 理解编译程序总的框架，明确编译程序工作的基本过程及各阶段的基本任务
3. **重点**　对编译程序的功能和结构做一综述
4. **难点**　了解编译程序各个成分在编译阶段的逻辑关系以及他们怎样作为一个整体完成编译任务的

教学内容

1.1. 程序的翻译

1.2. 编译程序的组成

1.3. 编译程序构造

1.4. 编译技术的应用及发展

程序的翻译

- 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，功效高，但使用复杂、繁琐、费时、易出错
- 高级语言

  -- Fortran、Pascal、C 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。

高级程序设计语言优点

- 效率高（编程），可移植性好；
- 不必考虑与机器有关的问题；

  如存贮单元的分配等。
- 丰富的数据结构；
- 接近自然语言，易掌握；

高级程序设计语言定义

- 高级程序设计至今还没有严格的定义，这里沿用IBM公司J.E.Sammet所给出的定义：

  高级程序设计语言，简言之，就是易于为人们理解，掌握与使用的程序设计语言，或者说，它是具有如下四个特征的表示法、约定与规则的集合。

高级程序设计语言定义（续）

(1) 高级语言不要求程序员掌握有关机器代码的知识（如寄存器，数据的内部表示，I/O通道，这一特性没有考虑效率因素）。

(2) 高级语言本身应独立于任何特定的计算机，高级语言应易于编写出能在多种计算机上运行的程序。

(3) 用高级语言编写出的源程序应能翻译成可在多种机器上运行的机器代码程序。

(4) 高级语言设施应能相当自然地表示所求解问题域，即应能用面向问题的方式编写高级语言程序。

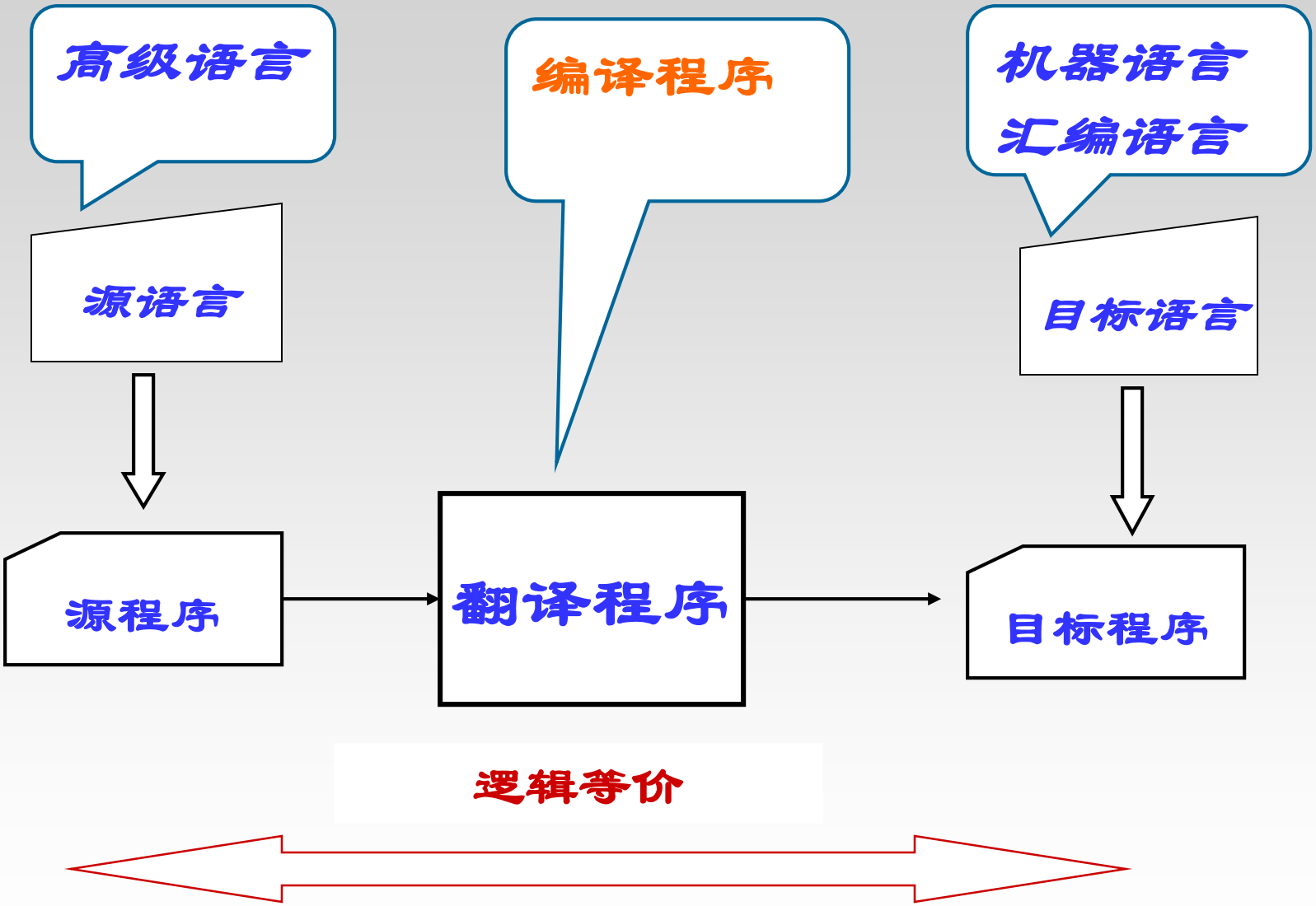
编译系统

- 高级语言的这些优点是有**代价**的，必须有一个“**翻译程序**”，也就是说高级语言必须有相应的软件支持系统“编译程序+相应的支持用户程序”——**编译系统**。
- 一般这种翻译工作有二种途径

  编译　编译程序

  　　汇编程序

  解释　解释程序


高级语言
编译程序
机器语言
汇编语言
源语言
目标语言
源程序
翻译程序
目标程序
逻辑等价

编译程序和汇编程序

- **编译程序**

  如果翻译程序的**源语言是高级语言**，目标语言是低级语言（机器语言或汇编语言），这种翻译程序称为*编译程序*。

- **汇编程序**

  如果翻译程序的**源语言是汇编语言**，目标语言是机器语言，这种翻译程序称为*汇编程序*。

说明：

- 编译程序是一个与源语言和计算机有关的概念。

  ——不同的源语言有不同的编译程序

  ——同一种源语言可以有不同的编译程序（高级语言或汇编语言书写）

- 源程序的执行分为两个阶段：

  编译阶段和运行阶段

- 编译阶段生成的目标程序不是机器代码程序，而是符号汇编程序，源程序的执行分三个阶段：编译阶段、汇编阶段和运行阶段.

- **解释程序**

  按源程序的动态顺序逐句地进行分析解释并执行直至结束。

- 解释程序边翻译边执行，不生成目标程序。交互式的工作方式，便于调试，但执行效率低，执行时也要解释。
- 编译程序生成目标程序，链接形成可执行文件运行，所有翻译工作在运行之前完成。执行效率高。

- **源程序**

  用汇编语言或高级语言编写的程序称为源程序

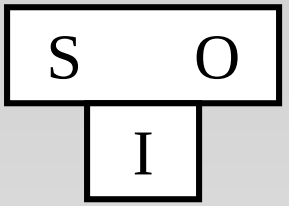
- **目标程序**

  用目标语言所表示的程序

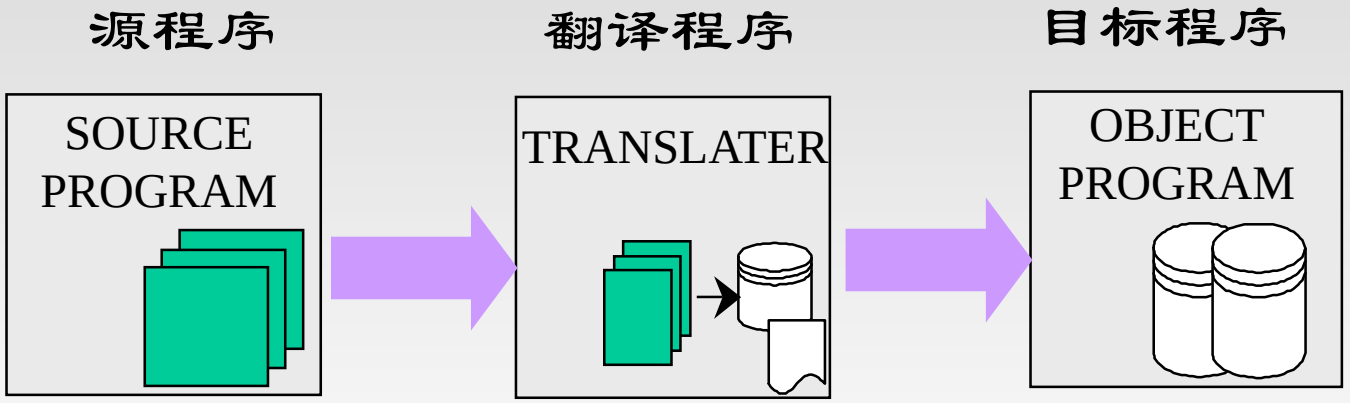
  目标语言：可以是介于源语言和机器语言之间的“中间语言”，可以是某种机器的机器语言，也可以是某机器的汇编语言。

- **翻译程序**

  将源程序转换为目标程序的程序称为翻译程序。它是指各种语言的翻译器，包括汇编程序和编译程序，是汇编程序、编译程序以及各种变换程序的总称。



源程序、翻译程序、目标程序 三者关系：



即源程序是翻译程序的输入，目标程序是翻译程序的输出

- **汇编**

若源程序用汇编语言书写，经过翻译程序得到用机器语言表示的程序，这时的翻译程序就称之为汇编程序，这种翻译过程称为“汇编”(Assemble)

- **编译**

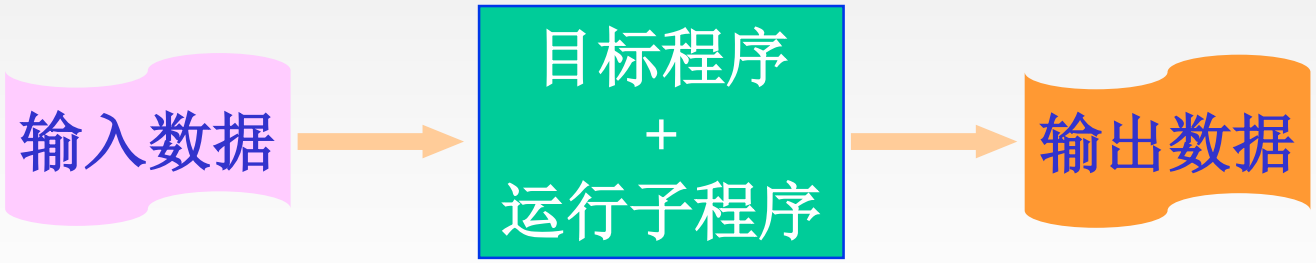
若源程序是用高级语言书写，经加工后得到目标程序，上述翻译过程称“编译” (Compile)

汇编程序与编译程序都是翻译程序，主要区别是加工对象的不同。由于汇编语言格式简单，常与机器语言之间有一一对应的关系。汇编程序所要做的翻译工作比编译程序简单的多。

源程序的编译和运行

- 编译或汇编阶段

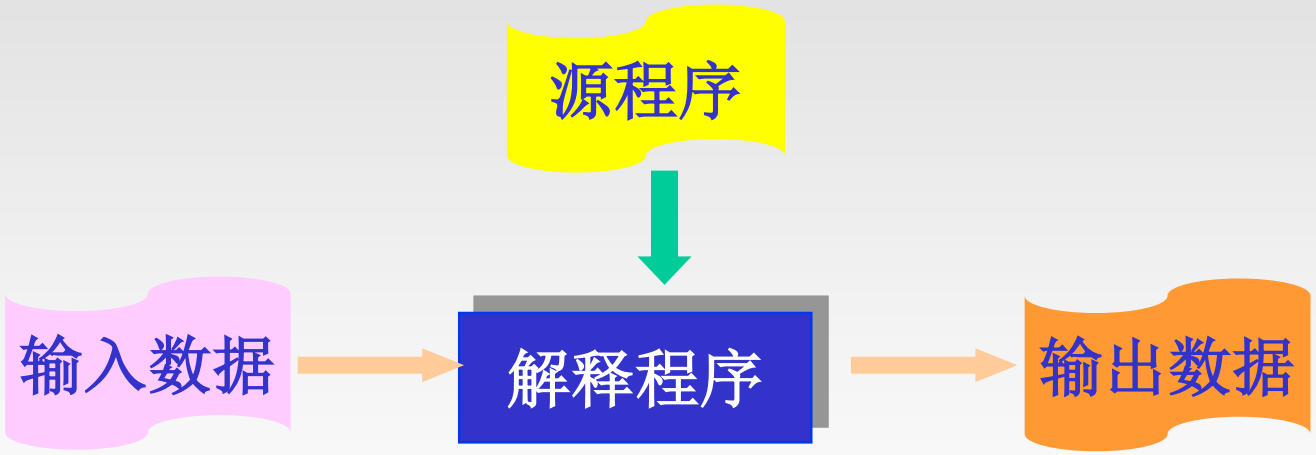
- 运行阶段



解释程序（Interpreter）（类似于口译，不生成目标代码）

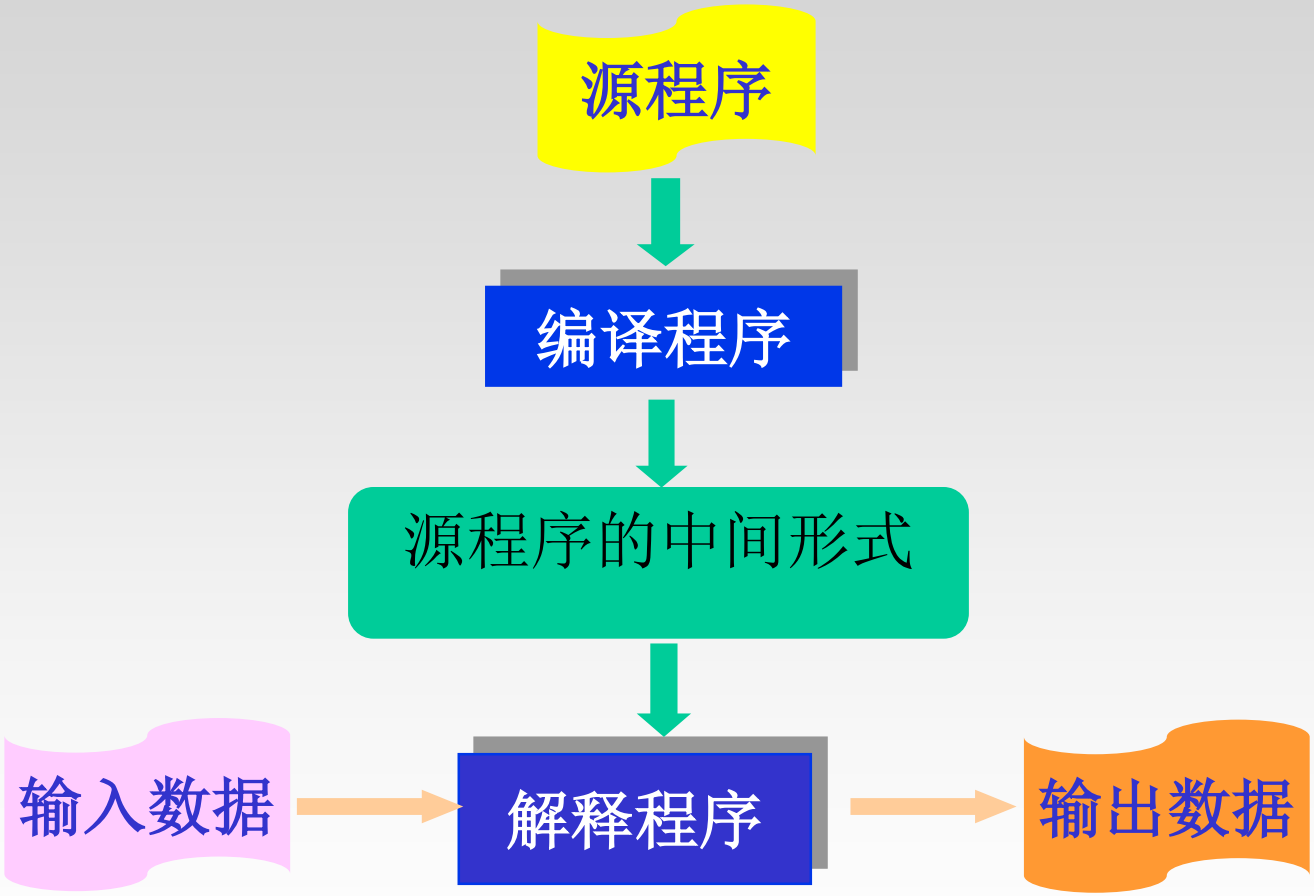
对源程序进行解释执行的程序。

- 工作过程

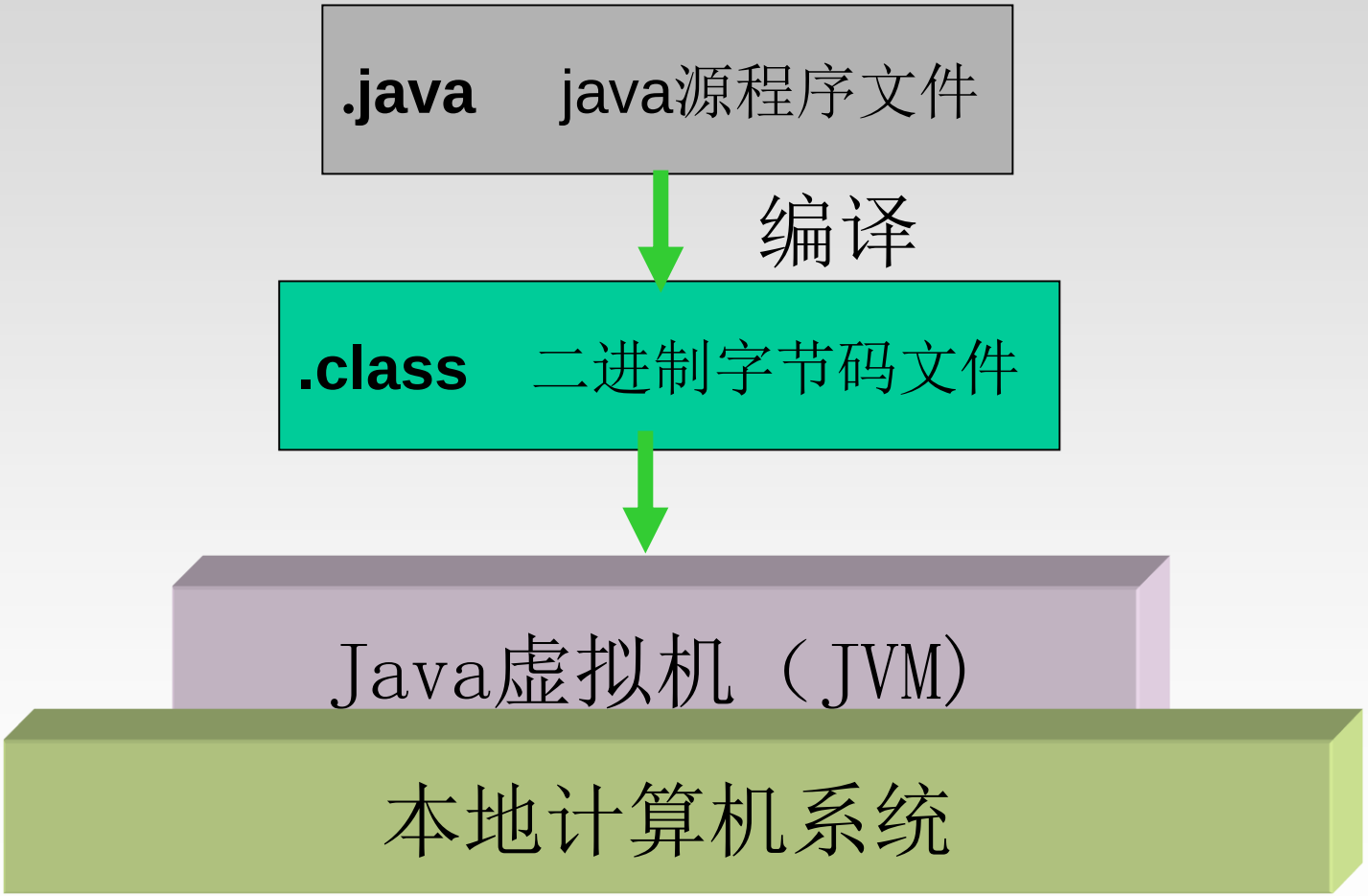


• 特点：与编译系统比较，解释系统较简单、可移植性好，易于查错，但速度慢

“编译-解释执行”系统



例如Java语言



编译程序的组成

所谓编译过程是指将高级语言程序翻译为等价的目标程序的过程。

翻译外文资料：

1、能识别出句子中的一个单词；
2、分析句子的语法结构；
3、根据句子的含义进行初步翻译；
4、对译文进行修饰；
5、写出最后的译文。

编译过程

- 翻译和编译工作的比较

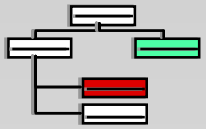
	翻译外文	编译程序
分析	识别单词 分析句子 根据语义进行初步翻译	词法分析 语法分析 语义分析、生成中间代码
综合	修辞加工 写出译文	代码优化 目标代码生成

编译过程

习惯上是将编译过程划分为5个基本阶段：

（编译程序有8个部分）

一、词法分析



任务： 根据**词法规则**分析和识别**单词**

字符序列 → 编码形式

➢ **单词**：是语言的基本语法单位

〈1〉**保留字**（如:if、else、while）

〈2〉**标识符**（如:max、min、str）

〈3〉**常数**（如:12、6.8、’a’）

〈4〉**分界符**（如:+、-、*、/、;、（、））

词法分析程序的结果---二元式

y = x + r * 6

单词值	单词类别
y	标识符
=	分界符（赋值）
x	标识符
+	分界符（加号）
r	标识符
*	分界符（乘号）
6	常数

有关术语

- 词法分析(lexical analysis or scanning)

  --The stream of characters making up a source program is read from left to right and grouped into tokens,which are sequences of characters that have a collective meaning.

- 单词---token
- 保留字---reserved word
- 标识符 ---identifier(user-defined name)

二、语法分析（编译程序的核心）

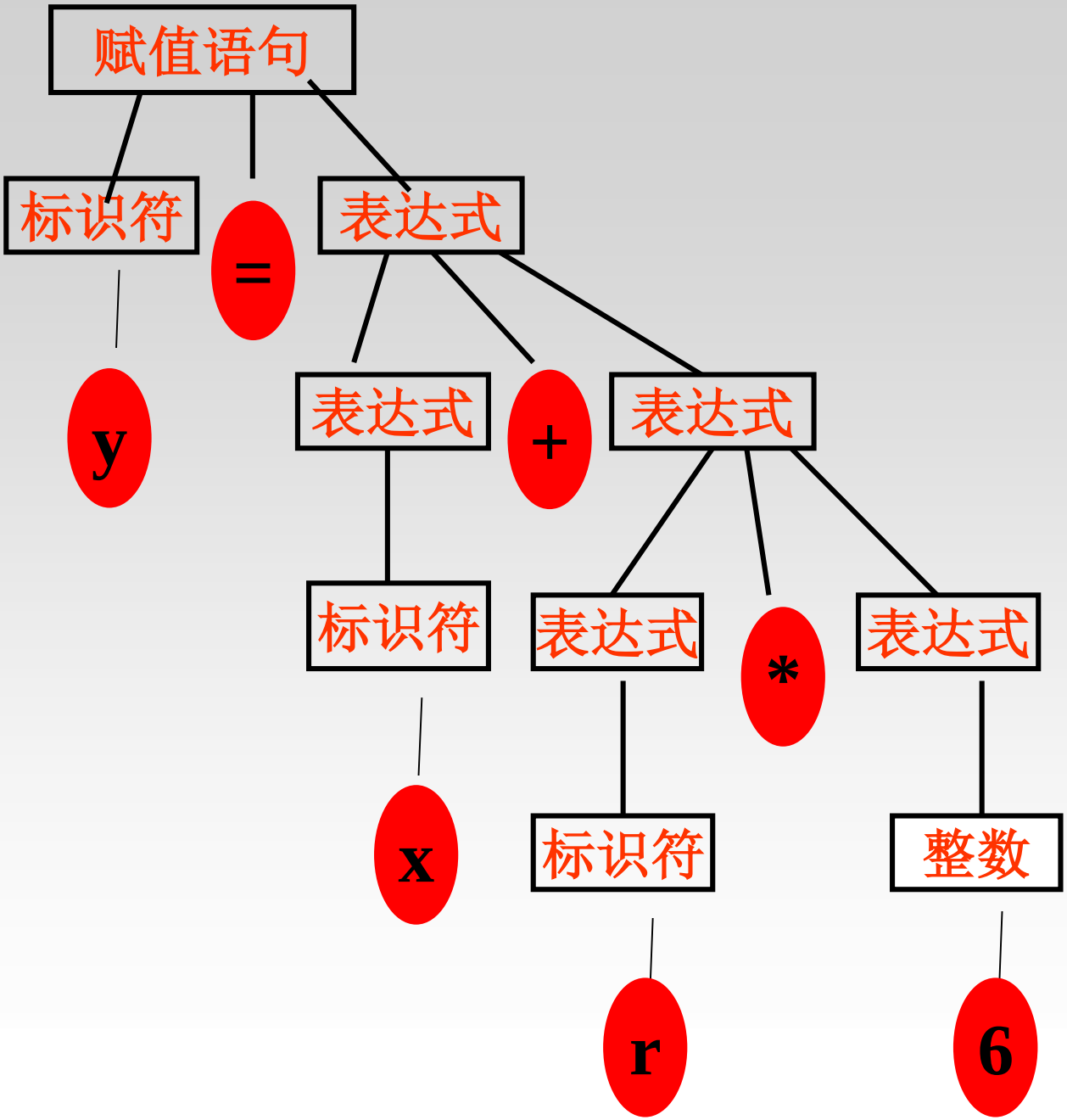
任务： 根据**语法规则**（即语言的文法），分析并识别出各种**语法成分**（如表达式、语句、函数等），并进行**语法正确性检查**。

文法

〈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〉“)” | 〈标识符〉 | 〈整数〉 | 〈实数〉



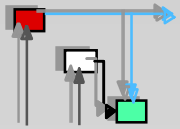
语法分析的结果----语法树

有关术语

- **语法分析(syntax analysis or parsing)**

  The purpose of syntax analysis is to determine the source program's phrase structure.This process is also called parsing.The source program is parsed to check whether it conforms to the source language's syntax,and to construct a suitable representation of its phrase structure.

三、语义分析及中间代码生成



任务： 依据语义规则对识别出的各种语法成份分析其含义，并进行初步翻译，生成中间代码。

- **静态：**

  分析语法成份的含义，进行语义上的正确性检查

- **动态：**

  根据相应语义，生成中间代码（介于源语言和目标语言之间的中间语言形式）

生成中间代码的目的：

- 1、利于代码优化
- 2、利于目标代码的移植

中间代码的形式：

- 四元式、三元式、逆波兰表示

★ 四元式

例：y = x + r * 6

	运算符	左运算对象	右运算对象	结果
(1)	inttoreal	6	--	t1
(2)	*	r	t1	t2
(3)	+	t2	x	t3
(4)	=	t3		y

其中t1、t2、t3为编译程序引入的临时工作单元

有关术语

- **语义分析(semantic analysis)**

  The parsed program is further analyzed to determine whether it conforms to the source language's contextual constraints:scope rules, type rules

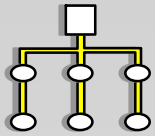
e.g. To relate each applied occurrence of an identifier in the source program to the corresponding declaration.

有关术语

- 中间代码生成(intermediate code generation)

  This is where the intermediate representation of the source program is created.We want this representation to be easy to generate,and easy to translate into the target program.The representation can have a variety of forms,but a common one is called three-address code or 4- tuple code.

四、代码优化



例：y = x + r * 6

任务： 对中间代码进行**加工变换**，以得到高质量的目标代码

	运算符	左运算对象	右运算对象	结果
(1)	inttoreal	6	--	t1
(2)	*	r	t1	t2
(3)	+	t2	x	t3
(4)	=	t3		y

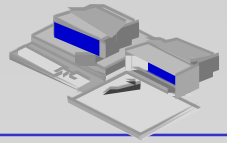
	运算符	左运算对象	右运算对象	结果
(1)	*	r	6.0	t1
(2)	+	t1	x	y

有关术语

- **代码优化(Intermediate code optimization)**

  The optimizer accepts input in the intermediate representation and output a version still in the intermediate representation .In this phase,the compiler attempts to produce the smallest,fastest and most efficient running result by applying various techniques.

五、目标代码生成

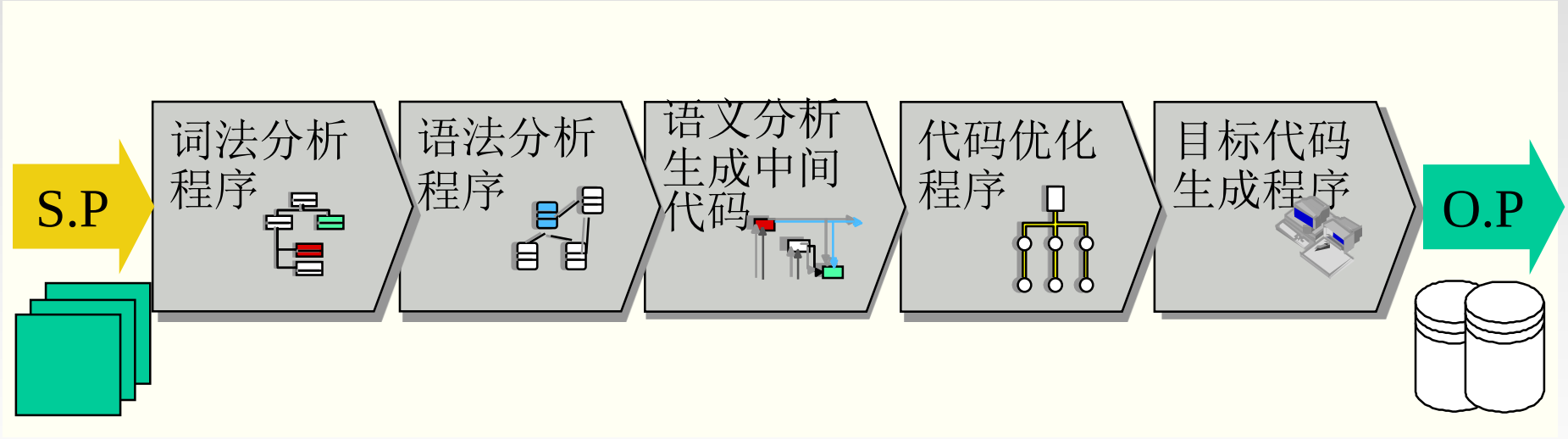


任务： 把中间代码变换成特定机器上的低级语言代码

	运算符	左运算对象	右运算对象	结果
(1)	*	r	6.0	t1
(2)	+	t1	x	y

```
mov    r,     R1
mul    #6.0,  R1
mov    x,     R2
add    R1,    R2
mov    R2,    y
```

编译过程小结



按逻辑功能不同，可将编译过程划分为五个基本阶段，与此相对应，我们将实现整个编译过程的编译程序划分为五个逻辑阶段（即五个逻辑子过程）。

每个阶段中都要有：

符号表管理和**错误检查处理**

编译程序的逻辑结构

★ 符号表管理

填表：把源程序中的信息和编译过程中所产生的信息登记在表格中

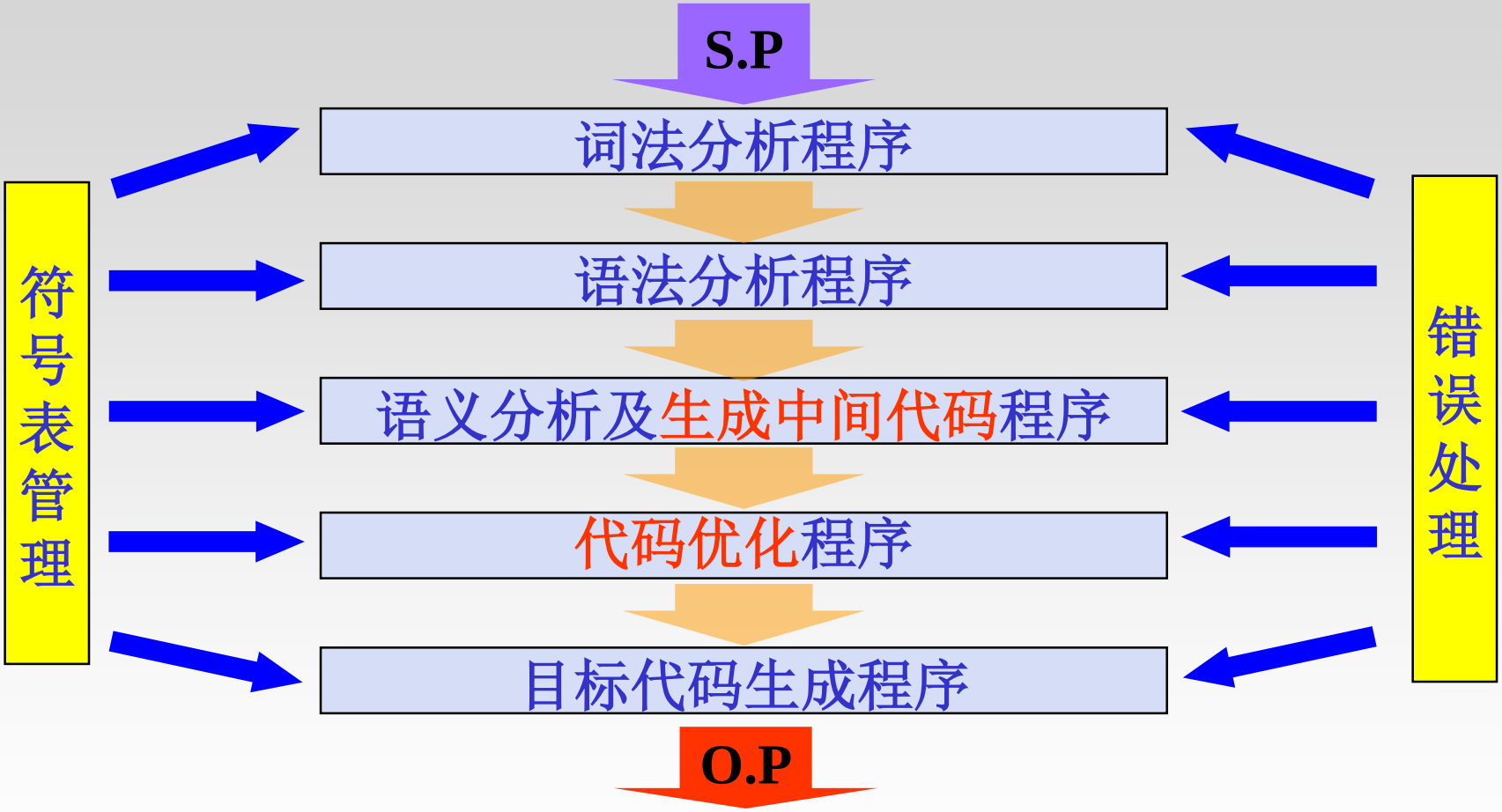
查表：在随后的编译过程中同时又要不断的查找这些表格中的信息

★ 错误处理

诊察错误，并能报告用户错误性质和位置

出错处理能力的优劣是衡量编译程序质量好坏的一个重要指标。

典型的编译程序具有7个逻辑部分



编译程序的前端和后端

根据编译程序各部分功能，将编译程序分成前端和后端

前端：通常将与源程序有关的编译部分称为前端。

词法分析、语法分析、语义分析、中间代码生成

---分析部分

特点：与源语言有关

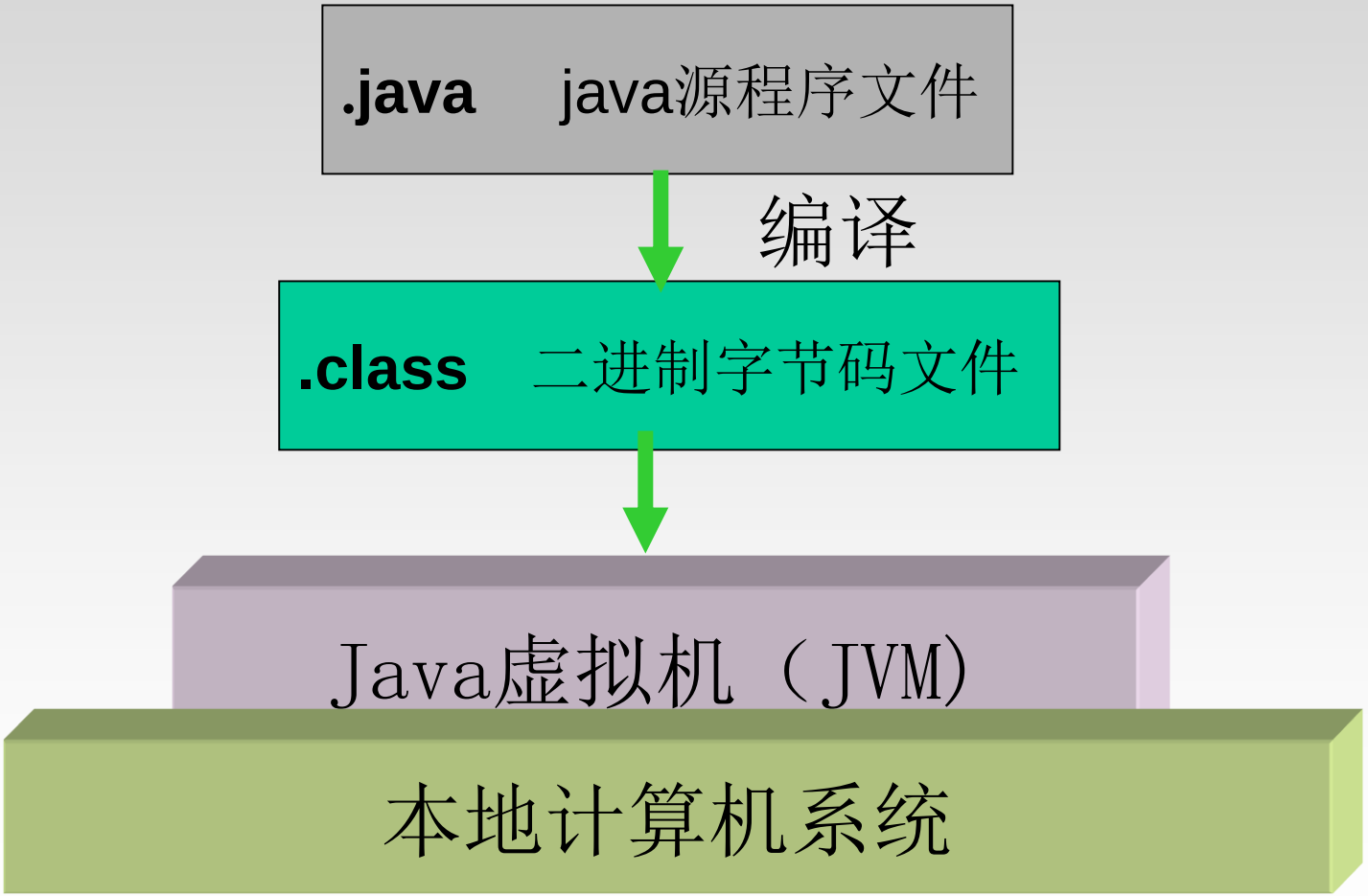
后端：与目标机有关的部分称为后端。

代码优化、代码生成 ---综合部分

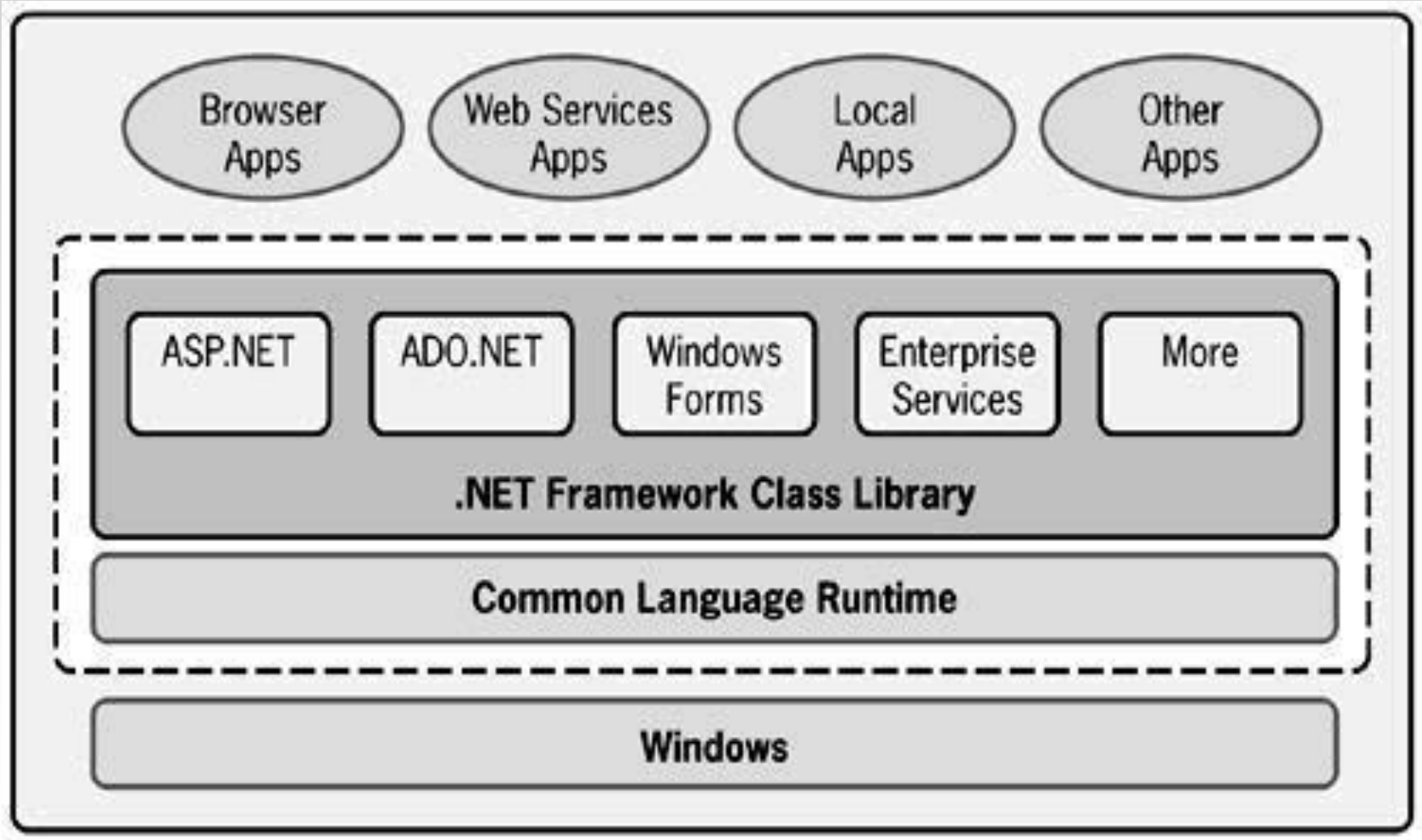
特点：与目标机有关

同一前端+不同后端 → 不同机器构成同一语言的编译程序

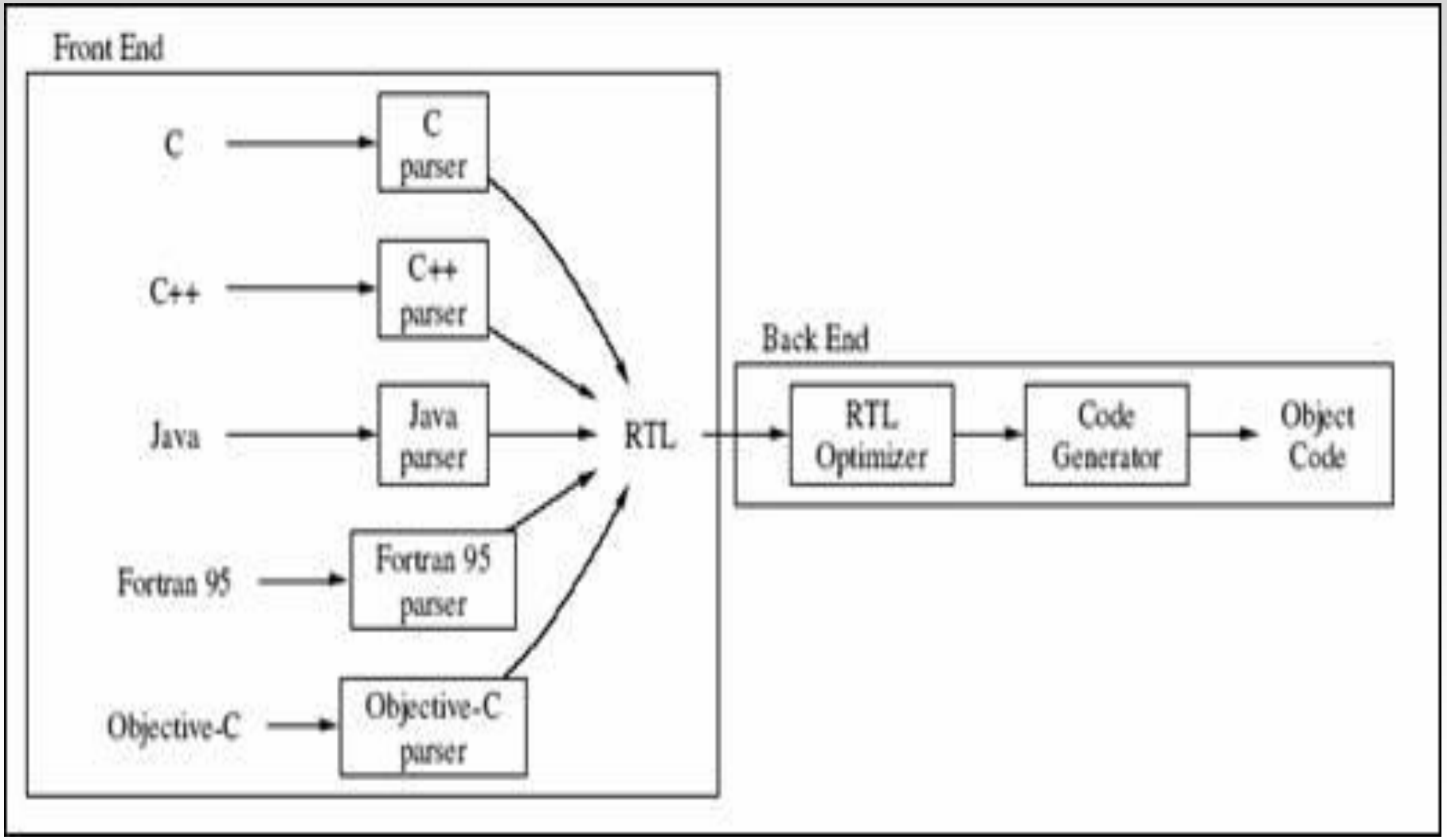
例如Java语言



例如.NET框架

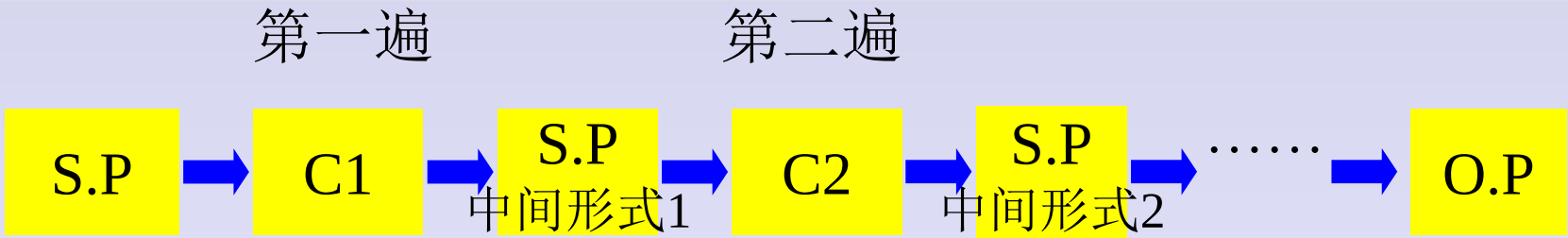

Browser Apps
Web Services Apps
Local Apps
Other Apps
ASP.NET
ADO.NET
Windows Forms
Enterprise Services
More
.NET Framework Class Library
Common Language Runtime
Windows

例如GCC


Front End
C
C++
Java
Fortran 95
Objective-C
C parser
C++ parser
Java parser
Fortran 95 parser
Objective-C parser
RTL
Back End
RTL Optimizer
Code Generator
Object Code

遍

对源程序（包括源程序中间形式）从头到尾扫描一次，并做有关的加工处理，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。

一遍扫描即可完成整个编译工作的称为**一遍扫描编译程序**

遍的划分视具体情况而定（内存的大小、源语言的繁简、目标程序质量的高低）

优点：

1、减少对内存容量的要求

2、编译程序结构清晰、各遍功能独立、相互联系简单

缺点：

增加读写中间文件的次数，降低效率

编译程序的构造

构造编译程序必须精通：

- 源 语 言
- 目标语言
- 编译方法

1.4 编译技术的应用及发展

•**应用：大部分软件工具的开发，都要使用编译技术和方法**

- 语法制导的结构化编辑器
- 程序格式化工具
- 软件测试工具
  - 静态分析器：不可能执行的代码、定义后未引用的变量
  - 动态测试工具：运行后与期望结果比较
- 程序理解工具：确定调用关系，画出流程图
- 高级语言的翻译工具

其它应用：

- 文本编辑器
- 信息检索系统
- 模式识别器
- 排版、绘图系统

发展

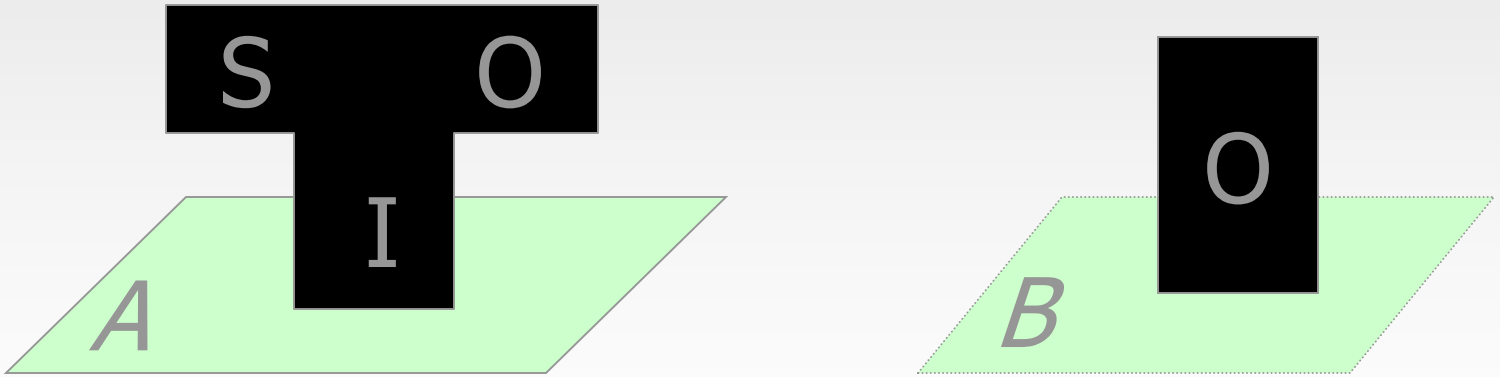
并行编译技术

- 目的：提高并行计算机体系结构的性能，超大规模计算的日益增长的需求
- 两种实现方法
  - 利用重构技术将串行程序并行化
  - 直接编写并行程序

交叉编译技术

交叉编译器

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**。



❑语言开发环境中的伙伴程序

- 编辑器（editor）
- 预处理器（preprocessor）
- 连接程序（linker）
- 装配程序（loader）
- 调试程序（debugger）

小结

- 内容
  - 1 什么是编译程序
  - 2 编译过程和编译程序的结构
  - 3 为什么要学习编译程序
- 重点
  - 对编译程序的功能和结构做一综述
- 难点
  - 了解编译程序各个成分在编译阶段的逻辑关系以及他们怎样作为一个整体完成编译任务的

练习

思考题

- 习题1、2、3

1. 判断下面的陈述是否正确。

（1）编译程序的五个组成部分缺一不可。

（2）高级语言程序到低级语言程序的转换是基于语义的等价变换。

（3）含有优化部分的编译程序的执行效率高。

（4）因为编译程序和解释程序具有不同的功能，所以它们的实现技术也完全不同。

（5）编译程序和解释程序的根本区别在于解释程序对源程序并没有真正进行翻译。

（6）无论一遍扫描的编译器还是多遍扫描的编译器都要对源程序扫描一遍。

答案：（1）F　（2）T　（3）F　（4）F　（5）F　（6）T

练　习

请指出下列错误信息是编译的哪个阶段报告的

(1) else没有匹配的if;

(2) 使用的函数没有定义;

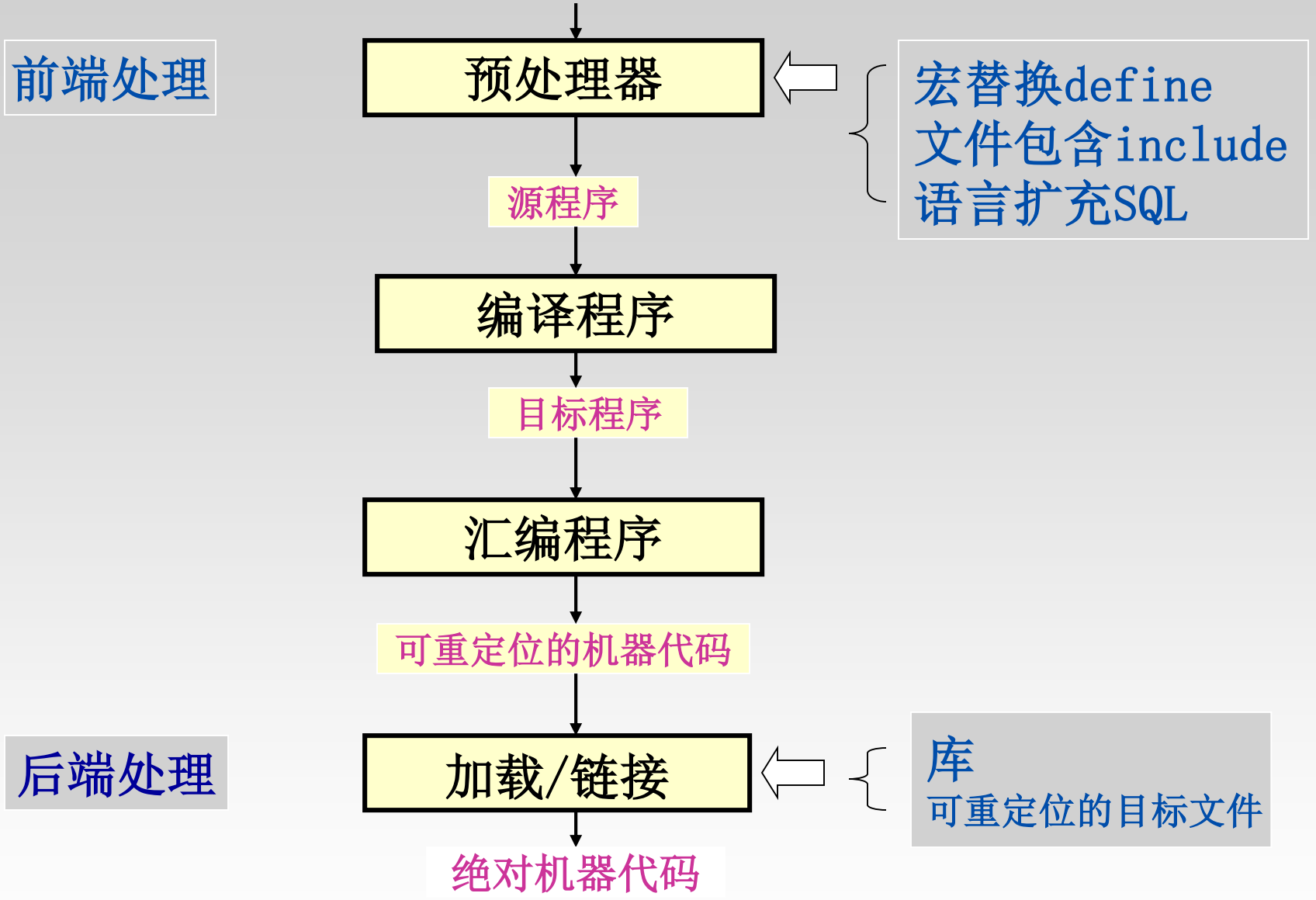
(3) 在数中出现非数字字符。

答案：(1) 语法　(2) 语义　(3) 词法

2．计算机执行用高级语言编写的程序有哪些途径？它们之间的主要区别是什么？

3．画出编译程序的总体逻辑结构图，简述各部分的主要功能。

课后习题：1-1，1-2，1-4


前端处理
预处理器
宏替换define
文件包含include
语言扩充SQL
源程序
编译程序
目标程序
汇编程序
可重定位的机器代码
后端处理
加载/链接
库
可重定位的目标文件
绝对机器代码

x:=2*x+y

↓

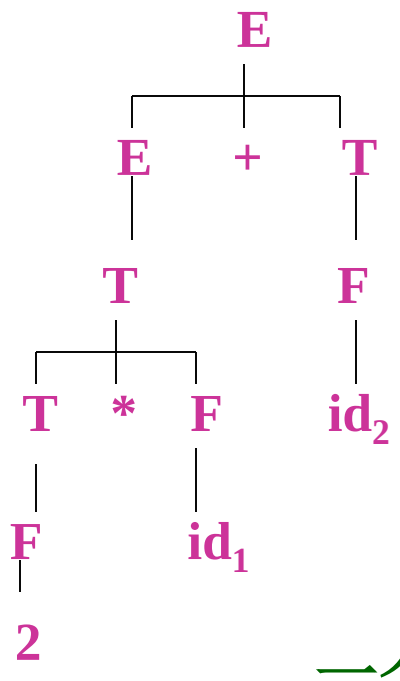
词法分析

↓

id_1:=2*id_1+id_2

↓

语法分析



↓

语义分析

↓

T_1=int to real(2)
T_2:=id_1*T_1
T_3:=id_2+T_2
id_1:=T_3

↓

代码优化

↓

T_1:=id_1*2.0
id_1:=id_2+T_1

↓

代码生成

↓

MOV R_2 id_1
MUL R_2 2.0
MOV R_1 id_2
ADD R_1 R_2
MOV id_2 R_1

一个语句的编译过程

编译技术和软件技术

- 语言的结构化编辑器
- 语言的调试和验证工具
- 语言程序测试工具
- 高级语言之间的转换
- 并行编译技术
- ……