编译原理与技术

葛宁
gening@buaa.edu.cn

自我介绍

葛宁，软件学院副教授，博士生导师

- 本科毕业于北京航空航天大学计算机学院
- 博士毕业于法国图卢兹国立综合理工学院
- 法国IRT-Saint Exupéry研究所和CNRS-LAAS研究所任研究人员
- 担任中国计算机学会形式化专委会委员、软件工程专委会委员

主讲课程

- 本科生“编译原理与技术”、“走进软件”
- 研究生“软件分析与测试”

主要研究方向

- 形式化方法、模型驱动与低代码开发，智能化软件工程等

欢迎有志从事形式化方法、模型驱动与低代码开发、智能化软件工程研究，希望解决有挑战的实际问题的同学到实验室学习

实验室研究方向

- 低代码、图形化编程、代码自动生成
- 软件/模型的分析、验证与测试
- 智能化/群智化软件工程

思考问题

编译，这个词大家一定不陌生。你使用的C语言程序是需要通过编译才能执行的。

- 你能告诉我什么是编译吗？
- 所有的程序都需要通过编译才能运行吗？
- 你认为这门课会讲什么？
- 你希望有什么收获？

大家思考一下。

课程目的

★目的:

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的构造（结构和机理）。

能运用所学技术解决实际问题，独立编写一个小型编译系统。

课程特点

- 经典课程
  - 国内外大部分学校都开设
  - 历史悠久
  - 有经典教材
  - 重视实践教学，尤其是国内外重点大学
  - 强调教学过程

编译技术

- 国外大学：名称不一样：
  - **Compiler Design**:　Carnegie Mellon
  - **Programming Languages and Compilers**: University of Illinois–Urbana-Champaign, University of California–Berkeley,
  - **Compilers and Interpreters** :Yale
  - **Compiler Design and Implementation** :Harvard
  - **Compiling Techniques**: Princeton
  - **Compilers**: Stanford
- 国内：编译原理、编译技术
- 内容差异不大

Purdure

- ECE 468 Introduction to Compilers and Translation Engineering
- ECE 573 Compilers and Translation Systems
- ECE 495S Introduction to Compilers and Translation Engineering
- ECE 663 Advance Optimized Compilers
- EE 573 Compilers & Translator Writing Systems

课程定位的理解

- 系统性：编译是一个完整的系统，也是同学们接触到的第一个系统

- 过程完整性：编译不仅讲解了编译技术，其实质是讲解了模型从一种语言表达形式到另一种语言表达形式的等价转化方法，应该保证过程的完整性

- 实践性：理论和实践相结合。无论是研究性大学、非研究性大学都应该注重实践。具体要求的可能不同

教学要求

- **理论学习**
  - 理解高级程序语言的工作原理
  - 学习不同语言表示的程序之间的等价转化方法
  - 编译程序的构造和工作原理
- **技术学习**
  - 程序分析技术和编译常用优化技术
  - 特定程序的自动生成技术
- **能力培养**
  - 理解系统的概念，完整设计一个不同难度的编译系统
  - 提高设计安全软件的能力

课程要求

理论基础课时：48学时（1－13周）

- 周一上午1-2节： 8:00-9:35 新主楼F122
- 周四上午3-4节： 9:50-11:25 新主楼F122

实践部分课时：32学时（10－17周）

- 周一下午8-10节：15:50-18:15 机房

成绩由两部分组成（暂定比例，有可能微调）：

- 理论基础(3学分)：课堂教学，按时交作业，占比60%
  - 作业，10分，每周一提交前一周作业
  - 3-6次随堂考试，共计30分
  - 期末闭卷考试，60分
  - 主动回答问题，每次奖励0.5分，5分封顶（考前公布）
- 实践部分(2学分)：上机实践（50机时），占比40%
  - 12月底检查作业

- 教材
  - 张莉等，《编译原理及编译程序构造》，清华大学出版社，2011年6月
  - 张莉等，《编译技术》，高教出版社，2016年9月
- 参考书
  - Alfred V. Aho等，《编译原理》（龙书）第2版:本科教学版，机械工业出版社，2009年5月
  - 刘春林等，《编译原理——学习指导与典型题解析》，国防工业出版社。2004年8月
- 要求
  - 提前预习，上课认真听讲
  - 课后及时复习，认真完成作业，按时提交作业
  - 独立完成实验内容

经典教材

- ✓ **龙书(Dragon book)**：书名：Compilers: Principles, Techniques, and Tools；作者：Alfred V.Aho, Ravi Sethi, Jeffrey D.Ullman；国内的编译原理教材基本都是参考了本书，重点是编译的前端技术。

- ✓ **鲸书(Whale book)**：书名：Advanced Compiler Design and Implementation；作者：Steven S.Muchnick；也就是高级编译原理。

- ✓ **虎书(Tiger book)**：书名：Modern Compiler Implementation in Java/C++/ML, Second Edition；作者：Andrew W.Appel, with Jens Palsberg；是3本书中最薄的一本！

助教（TA）：2人

姓名	学号	手机	邮箱
张智博	19373042	13614566701	hljzhangzhibo@buaa.edu.cn
郝泽钰	19373300	18235685812	19373300@buaa.edu.cn

建立微信群，发送学习资料，课后答疑

第一章 概论

（介绍名词术语、了解编译系统的结构和编译过程）

内　　容

1.1. 编译的起源：程序设计语言的发展

1.2. 基本概念

1.3. 编译过程和编译程序构造

1.4. 编译程序的前后处理器

1.5. 编译技术的其它应用

机器语言

裸机能够识别和执行的语言

- 机器能够识别指令系统，该指令系统即为机器语言
- 译码器翻译指令
- 左移、跳转、加1、… 功能简单，复杂功能人工构造
- 编码容易出错
- 可读性差，人难以理解
- 可维护性差

Operation	Address
0010	0000 0000 0100
0100	0000 0000 0101
0011	0000 0000 0110

```
156C
166D
5056
30CE
C000
```

汇编语言

功能与机器语言基本相同，将二进制码符号化

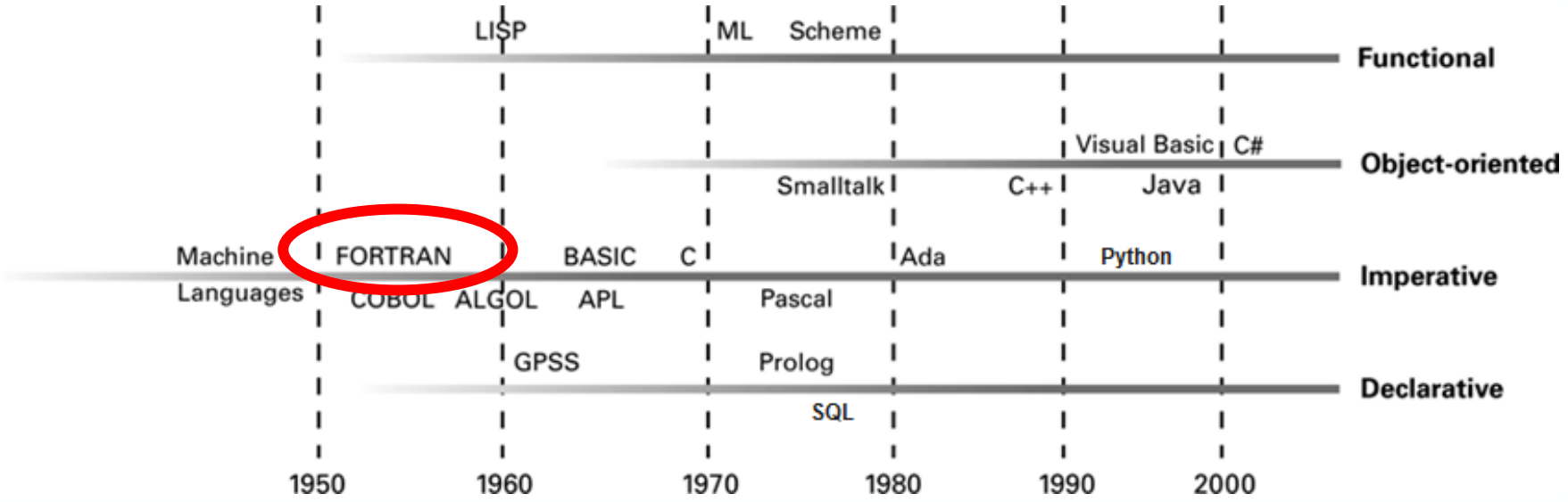
- 符号化的机器语言
- 可读性提升
- 需要理解底层硬件原理，哪些是通用寄存器、哪些是特征寄存器、内存，哪是数据区、哪是指令区
- 对用户要求高，没有“傻瓜模式”

Assembly language

```
LD    R5, Price
LD    R6, ShipCharge
ADDI R0, R5 R6
ST     R0, TotalCost
HLT
```

高级语言

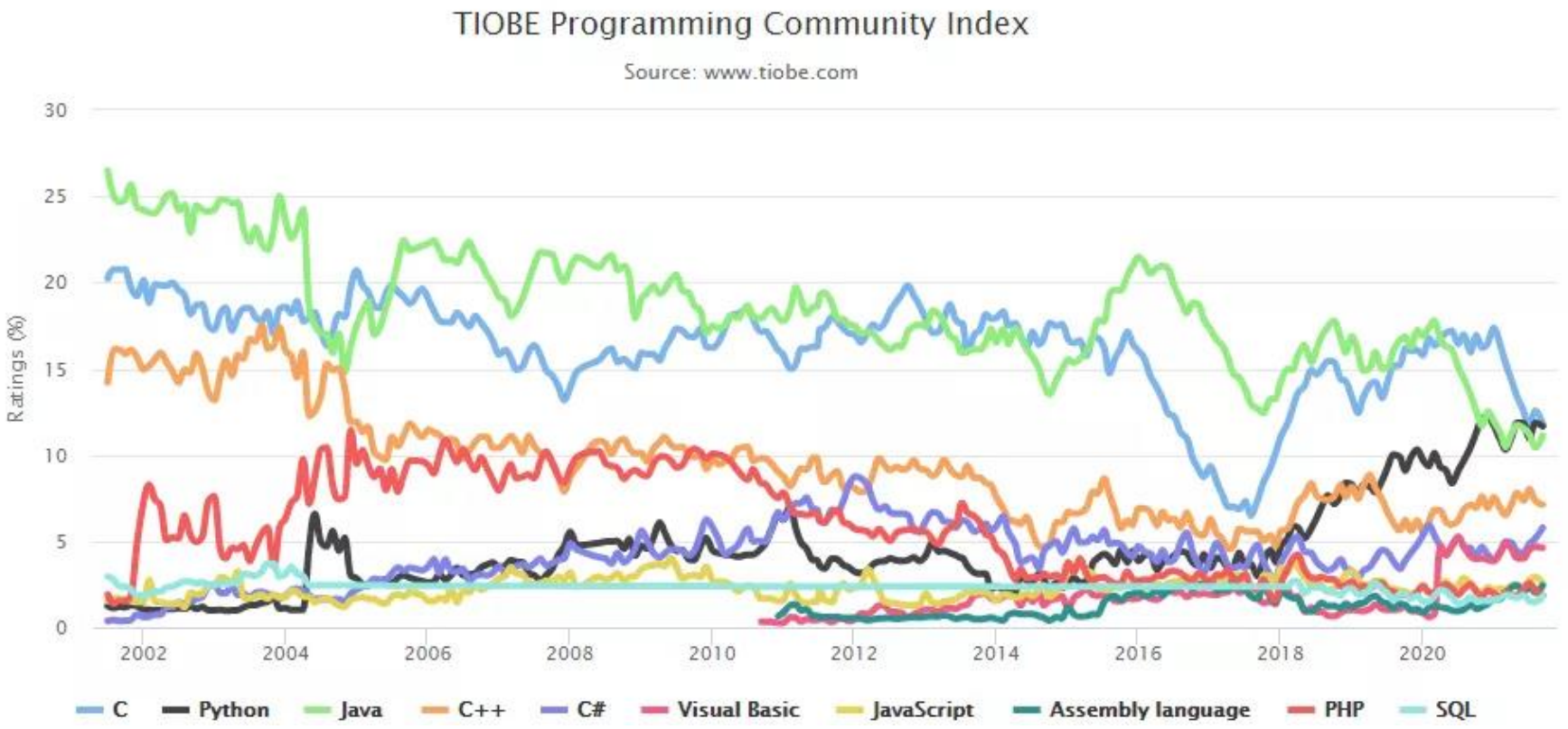
1956年，出现第一个高级语言FORTRAN



发展至今，成千上万种高级设计语言，有些流行，有些小众

高级语言

2021年9月Top10编程语言走势

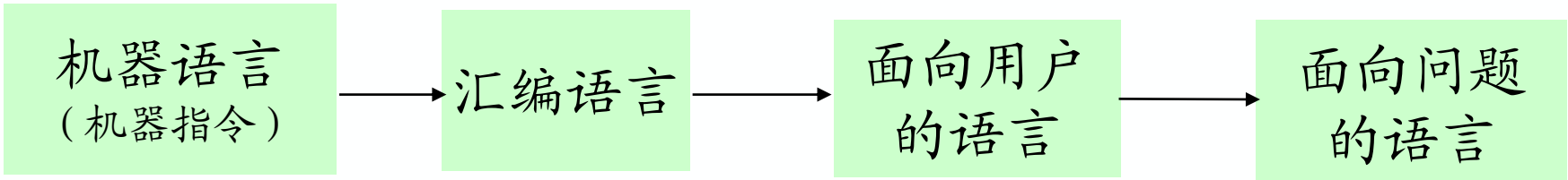


高级语言

2021年9月编程语言排行榜

Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		C	11.83%	-4.12%
2	3	↑	Python	11.67%	+1.20%
3	2	↓	Java	11.12%	-2.37%
4	4		C++	7.13%	+0.01%
5	5		C#	5.78%	+1.20%
6	6		Visual Basic	4.62%	+0.50%
7	7		JavaScript	2.55%	+0.01%
8	14	⇈	Assembly language	2.42%	+1.12%
9	8	↓	PHP	1.85%	-0.64%
10	10		SQL	1.80%	+0.04%
11	22	⇈	Classic Visual Basic	1.52%	+0.77%
12	17	⇈	Groovy	1.46%	+0.48%
13	15	↑	Ruby	1.27%	+0.03%
14	11	↓	Go	1.13%	-0.33%
15	12	↓	Swift	1.07%	-0.31%
16	16		MATLAB	1.02%	-0.07%
17	37	⇈	Fortran	1.01%	+0.65%
18	9	⇊	R	0.98%	-1.40%
19	13	⇊	Perl	0.78%	-0.53%
20	29	⇈	Delphi/Object Pascal	0.77%	+0.24%

程序设计语言的发展



C7 06 0000 0002　　MOV x, 2

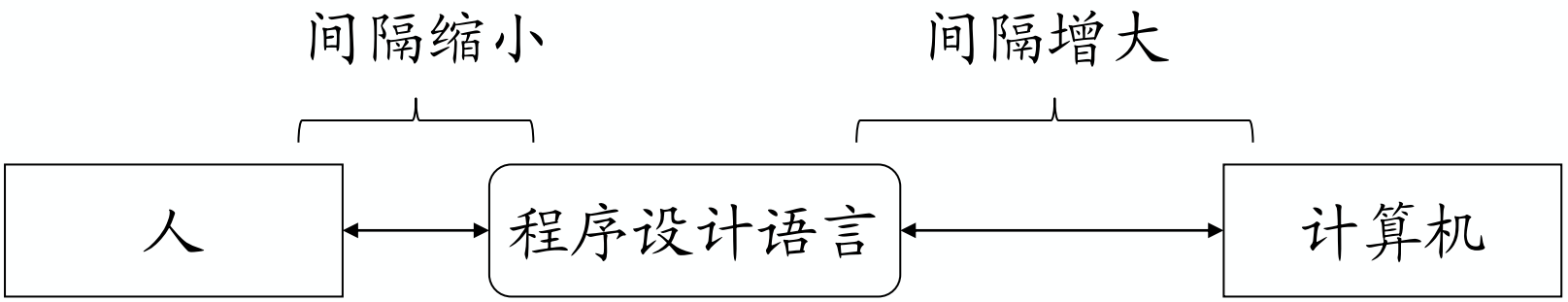
x = 2

低级语言

高级语言

程序设计语言

程序设计语言是人与计算机间沟通的桥梁



1.2 基本概念

- 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

用高级语言编制的程序，计算机不能立即执行，必须通过一个“翻译程序”加工，转化为与其等价的机器语言程序，机器才能执行。这种翻译程序，称之为“编译程序”。

- 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

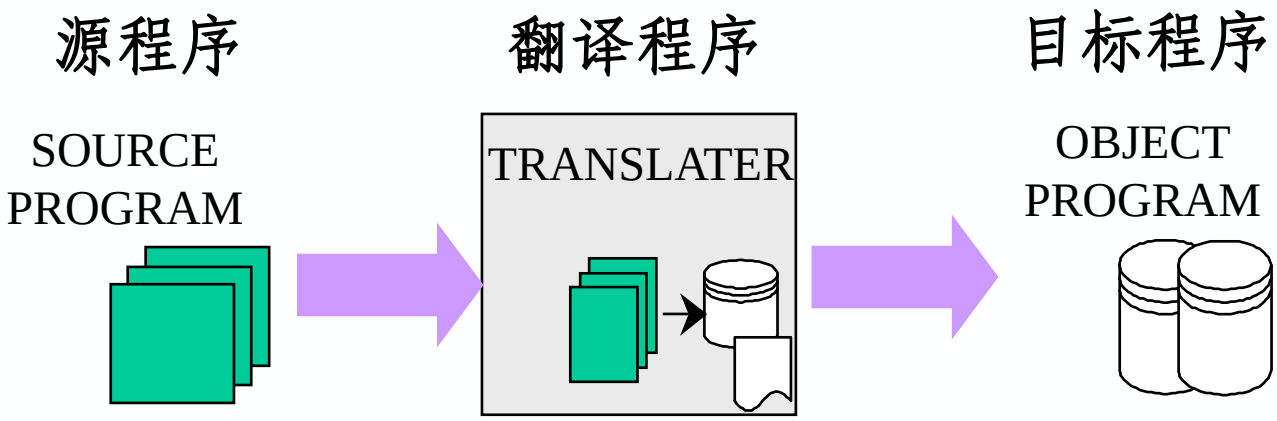
  用目标语言所表示的程序。

  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，也可以是某种机器的汇编语言。

- 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。

汇编语言 → 汇编程序 → 机器语言

高级语言 → 编译程序 → 目标程序

- **汇编程序**

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)。

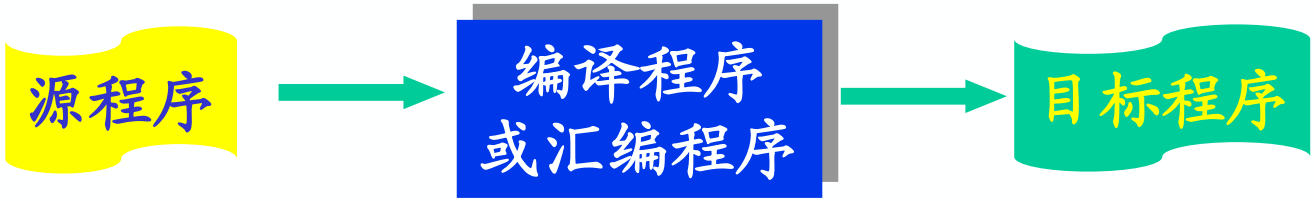
- **编译程序**

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)。

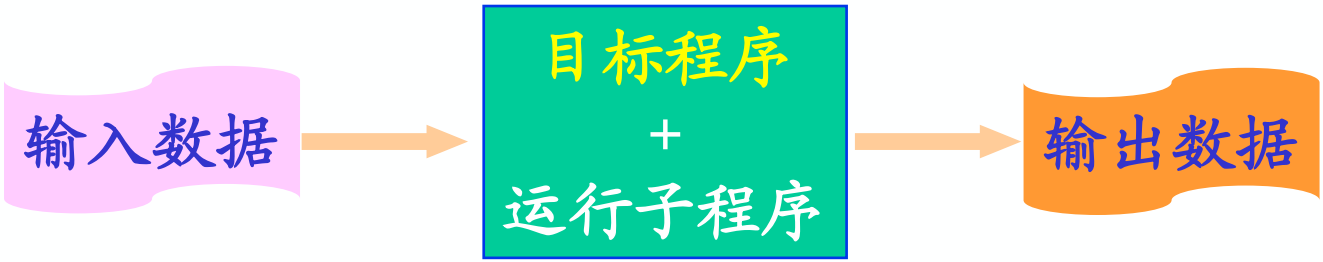
汇编程序与编译程序都是翻译程序，主要区别是加工对象的不同。由于汇编语言格式简单，常与机器语言之间有一一对应的关系，所以汇编程序所要做的翻译工作比编译程序简单得多。

源程序的编译和运行

- 编译或汇编阶段（compile-time）



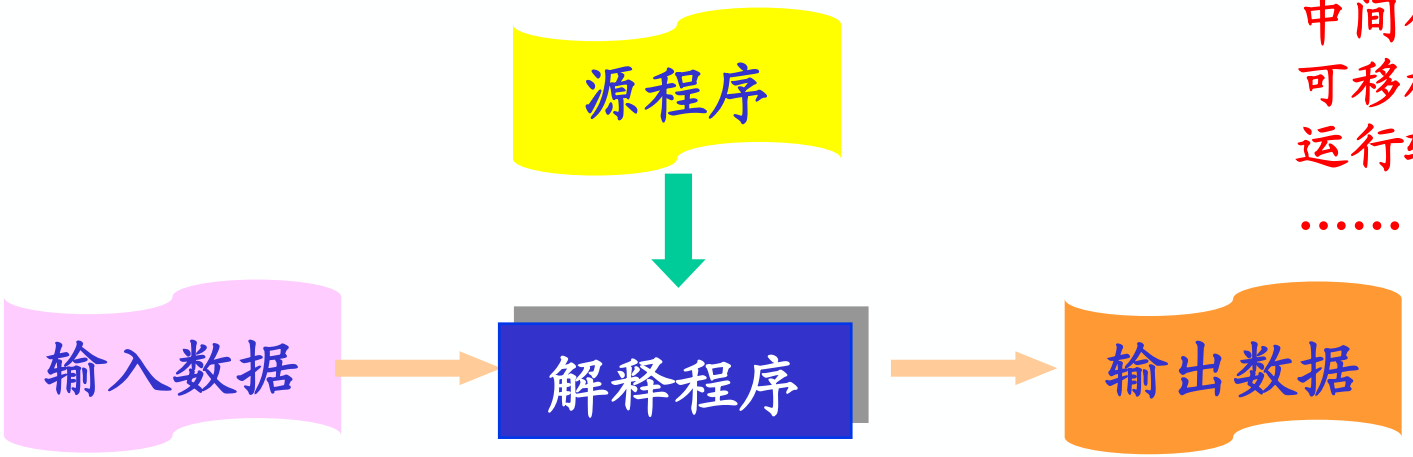
- 运行阶段（run-time）



•解释程序（Interpreter）

对源程序进行解释执行的程序。

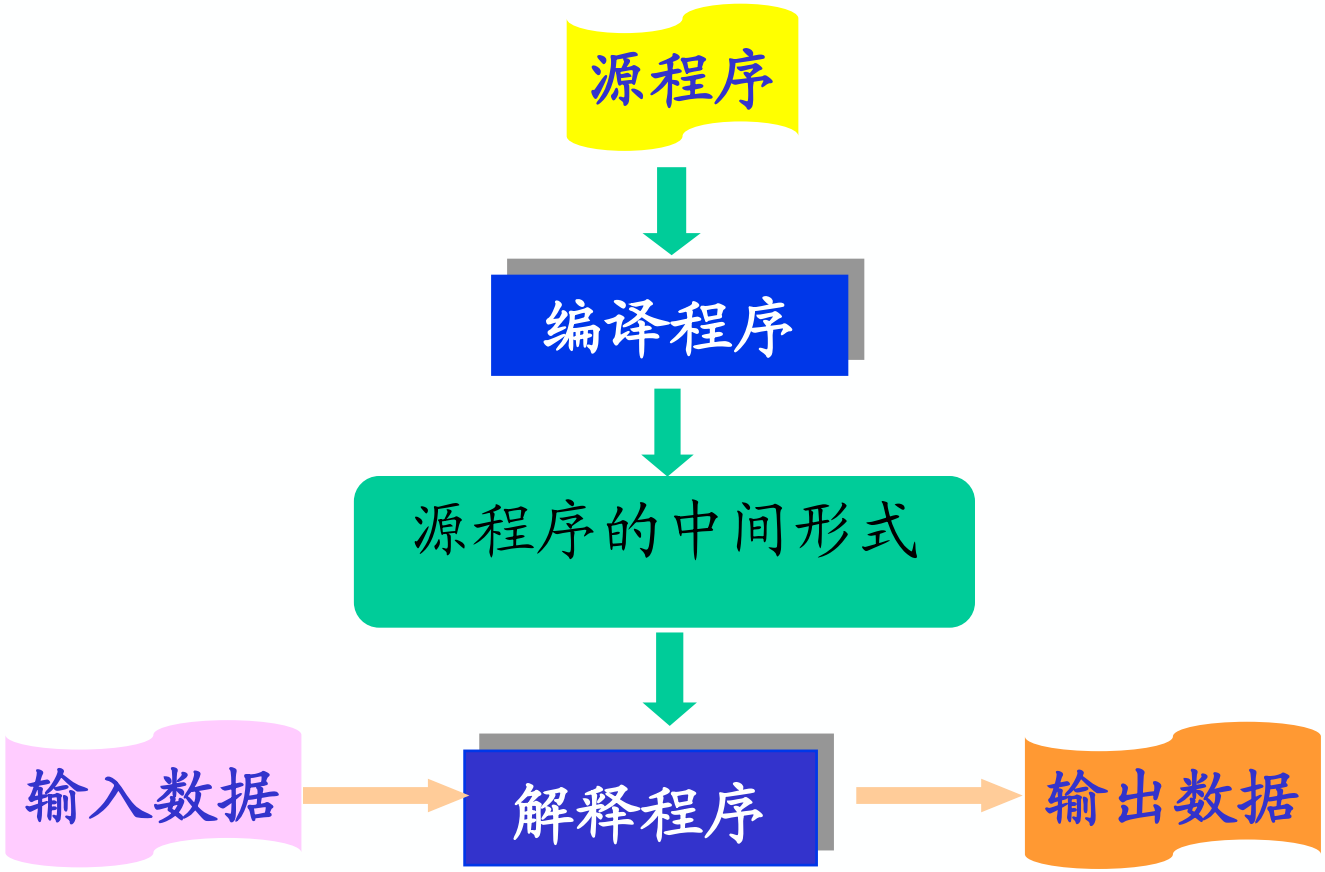
• 工作过程

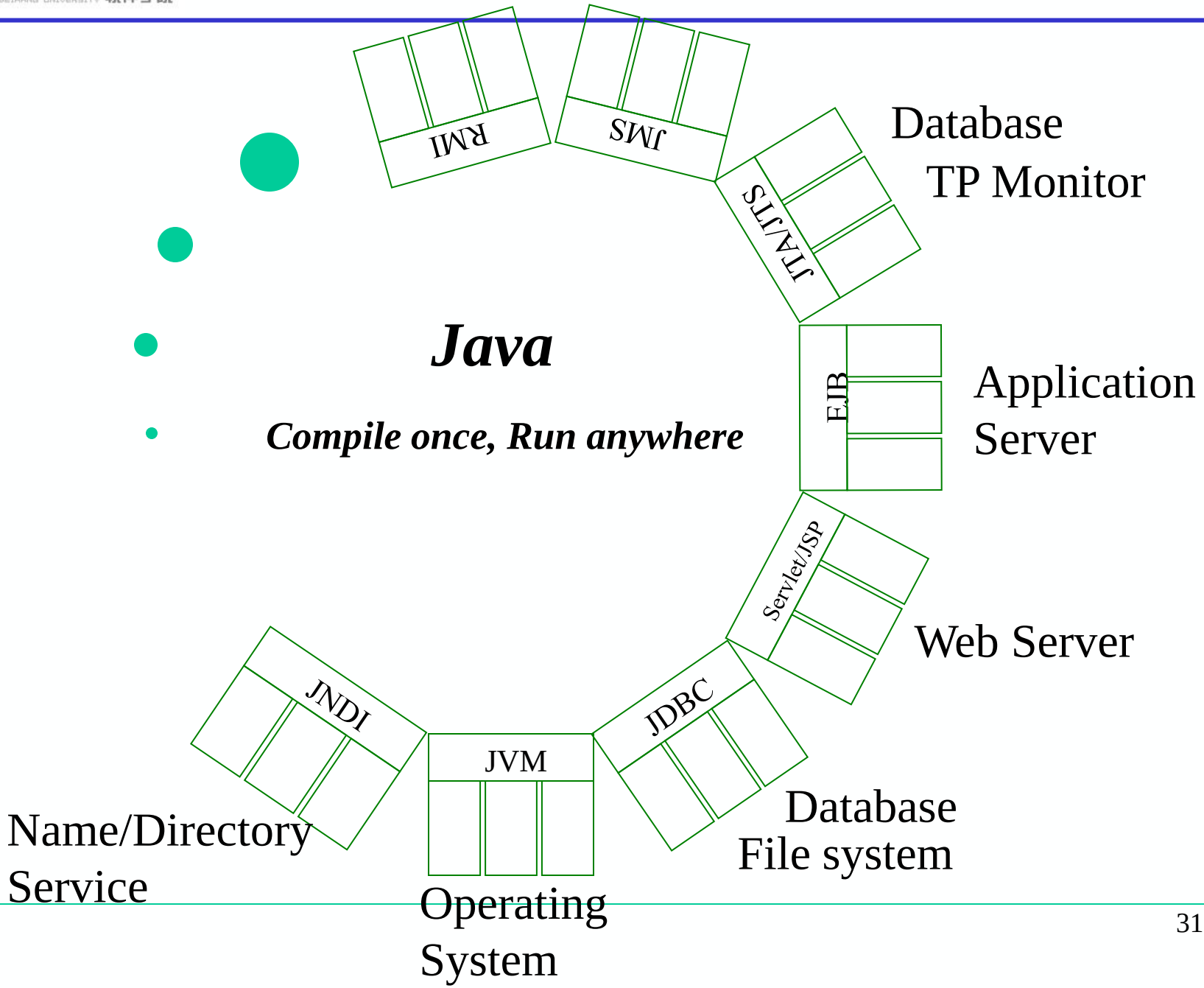


动态性强
中间代码存储量小
可移植性好
运行较慢
……

• 特点、与编译程序比较

“编译-解释执行”系统




RMI
JMS
JTA/JTS
Database
TP Monitor
Java
Compile once, Run anywhere
EJB
Application
Server
Servlet/JSP
Web Server
JNDI
JVM
JDBC
Database
File system
Name/Directory
Service
Operating
System

1.3 编译过程

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。

习惯上是将编译过程划分为5个基本阶段：

一、词法分析

任务：分析和识别单词。

源程序是由字符序列构成的，词法分析扫描源程序(字符串),根据语言的词法规则分析并识别单词，并以某种编码形式输出。

•单词：是语言的基本语法单位，一般语言有四大类单词

<1>语言定义的关键字或保留字（如BEGIN、END、IF）

对于如下的字符串,词法分析程序将分析和识别出9个单词:

X1	:=	(	2.0	+	0.8	)	*	C1
1	2	3	4	5	6	7	8	9

二、语法分析

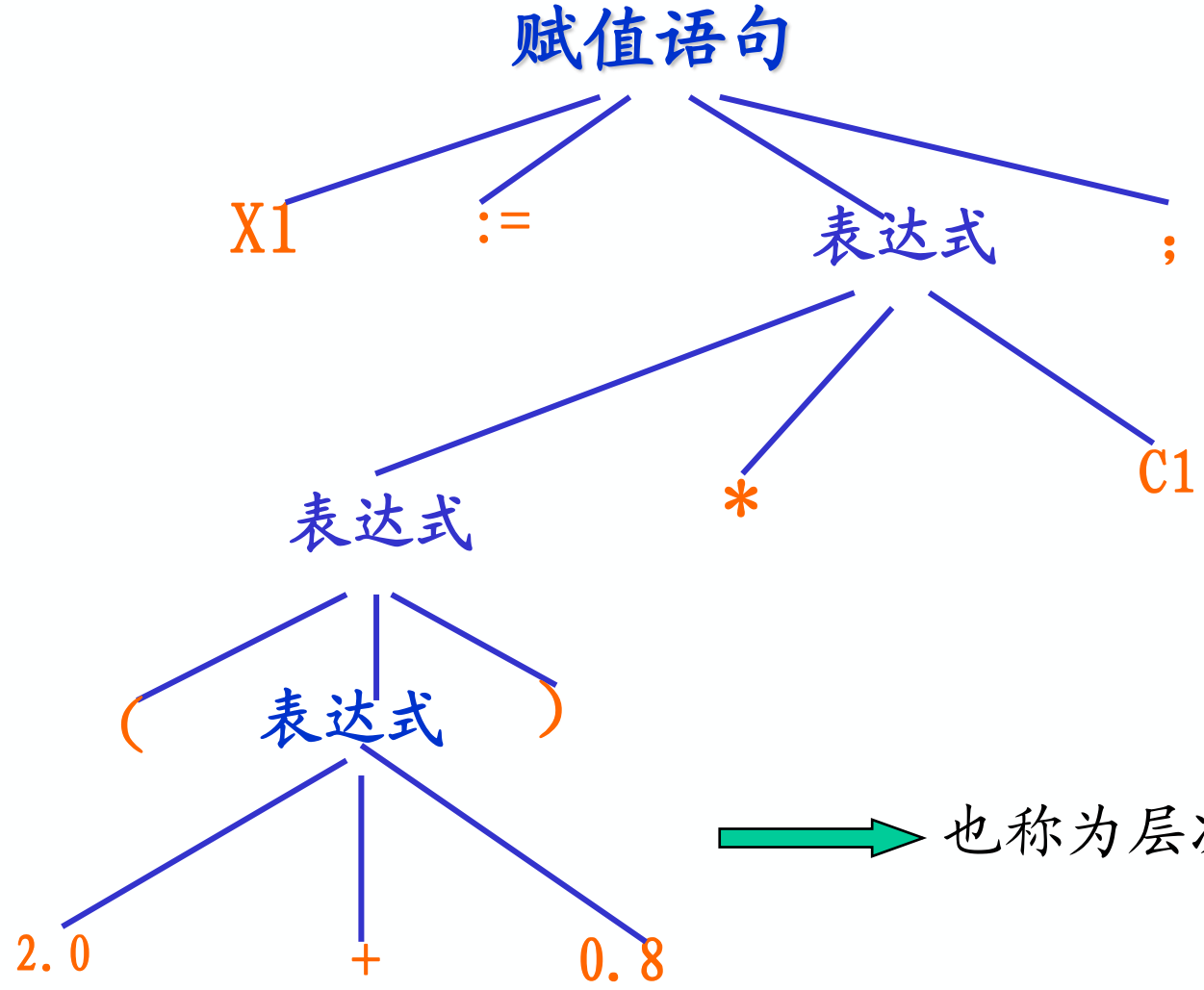
任务：根据语法规则（即语言的文法），分析并识别出各种语法成分，如表达式、各种说明、各种语句、过程、函数、程序等，并进行语法正确性检查。

例如，对于前面提到的例子X1 := (2.0 + 0.8) * C1 我们可以根据语言赋值语句的文法来分析和识别该语句（单词串）。首先给定文法：

<赋值语句>→<变量><赋值操作符><表达式>
<变量>→<简单标识符>
<赋值操作符>→:=
<表达式>→ ……

语法分析根据文法，将<变量>、<赋值操作符>、<表达式>识别出来，进而将赋值语句识别出来。在识别过程中进行语法检查；若有错误，则应输出出错信息。

X1:= (2.0 + 0.8) * C1;



→ 也称为层次分析。

三、语义分析、生成中间代码

任务：对识别出的各种语法成分进行语义分析，并产生相应的中间代码。

- 中间代码：一种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> 便于做优化处理；
  - <2> 便于编译程序的移植（中间代码不依赖于目标计算机）。
- 中间代码的形式：编译程序设计者可以自己设计，常用的有四元式、三元式、逆波兰表示等。

例： X1 := (2.0 + 0.8) * C1

由语法分析识别出为赋值语句，语义分析首先要分析语义上的正确性，例如要检查表达式中和赋值号两边的类型是否一致。

根据赋值语句的语义，生成中间代码。即用一种语言形式来代替另一种语言形式，这是翻译的关键步骤。（翻译的实质：语义的等价性）

下面介绍一种常用的中间代码来替换上述的赋值语句。

★ 四元式（三地址指令）

对于前面提到的例子 X1 := (2.0 + 0.8) * C1

	运算符	左运算对象	右运算对象	结果
(1)	+	2.0	0.8	T1
(2)	*	T1	C1	T2
(3)	:=	X1	T2	

其中T1和T2为编译程序引入的临时变量。

四元式的语义为：
$$2.0 + 0.8 \rightarrow T1$$
$$T1 * C1 \rightarrow T2$$
$$T2 \rightarrow X1$$

这样所生成的四元式与原来的赋值语句在语言的形式上不同，但语义上等价。

四、代码优化

任务：目的是为了得到高质量的目标程序。

例如：上面的四元式中第一个四元式是计算常量表达式值，该值在编译时就可以算出并存放在临时变量中，不必生成目标指令来计算，这样四元式可优化为：

编译时：　　**2.0 + 0.8 → T1**

运算符	左运算对象	右运算对象	结果
*	T1	C1	T2
:=	X1	T2	

优化前的四元式：

(1)	+	2.0	0.8	T1
(2)	*	T1	C1	T2
(3)	:=	X1	T2	

五、生成目标程序

由中间代码很容易生成目标程序（地址指令序列）。这部分工作与机器关系密切，所以要根据具体机器进行。在做这部分工作时（要注意充分利用累加器），也可以进行优化处理。

```
LOAD   2.0
ADD    0.8
STO    T1
LOAD   T1
MUL    C1
STO    T2
LOAD   T2
STO    X1
```

利用累加器的优化

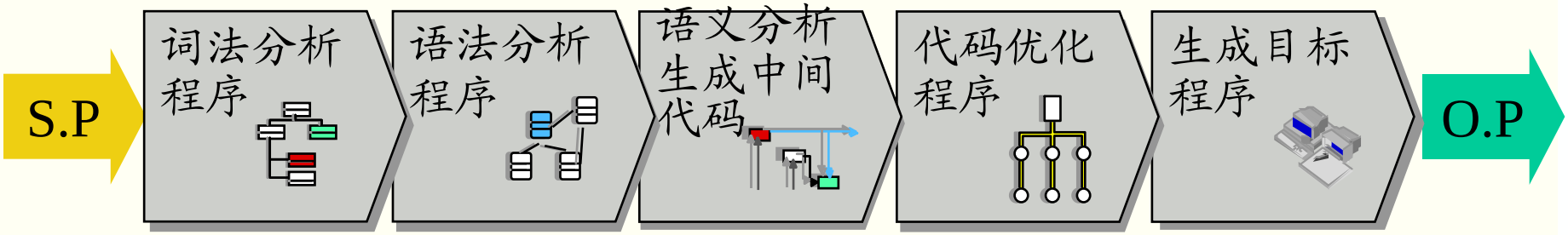
```
LOAD   2.0
ADD    0.8
MUL    C1
STO    X1
```

注意：在翻译成目标程序的过程中，要切记保持语义的等价性。

1.4 编译程序构造

1.4.1 编译程序的逻辑结构

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）。



在上列五个阶段中都要做两件事:

（1）建表和查表　（2）出错处理

所以编译程序中都要包括表格管理和出错处理两部分。

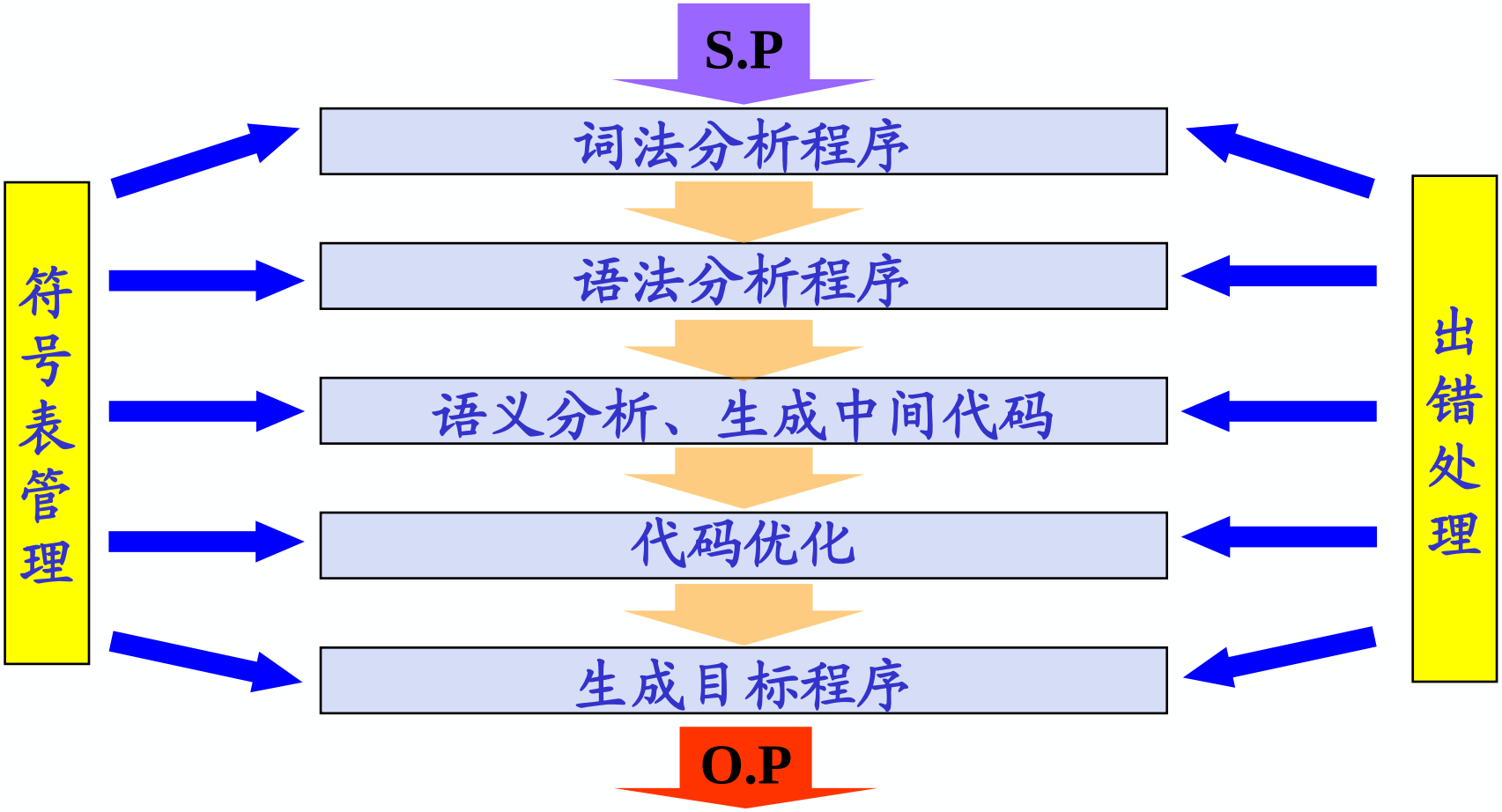
★ 表格管理（符号表组织）

在整个编译过程中始终都要贯穿着建表（填表）和查表的工作。即要及时地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，而在随后的编译过程中同时又要不断地查找这些表格中的信息。

★ 出错处理

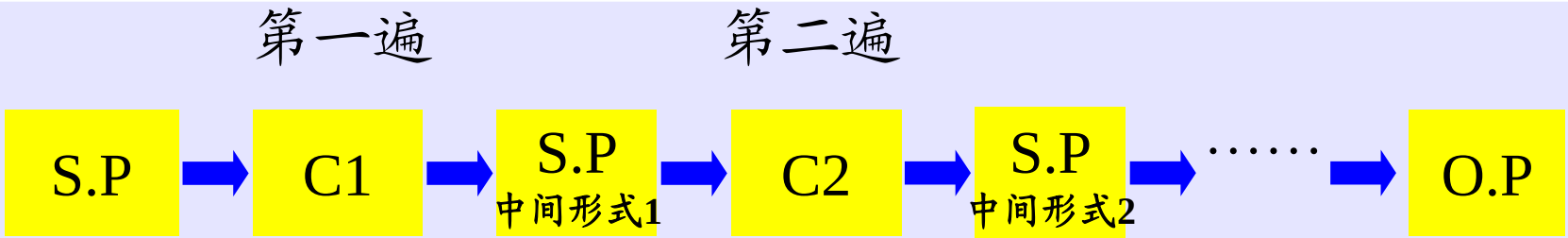
规模较大的源程序难免有多种错误。编译程序必须要有出错处理的功能，即能诊察出错误，并向用户报告错误性质和位置，以便用户修改源程序。出错处理能力的优劣是衡量编译程序质量好坏的一个重要指标。

典型的编译程序具有7个逻辑部分



1.4.2 遍（PASS）

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。



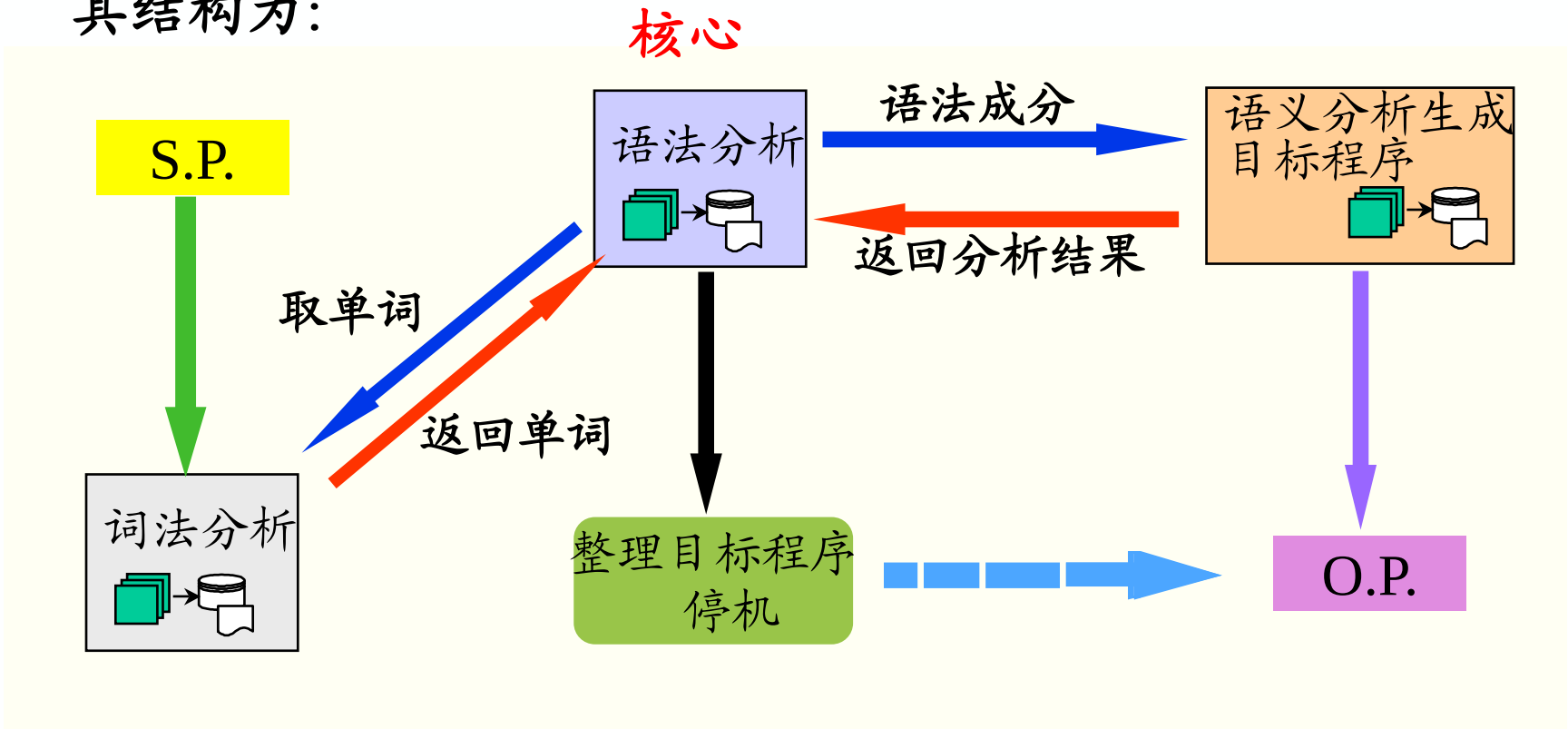
★要注意遍与基本阶段的区别

五个基本阶段：是将源程序翻译为目标程序在逻辑上要完成的工作。

遍：是指完成上述5个基本阶段的工作，要经过几次扫描处理。

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。

其结构为：

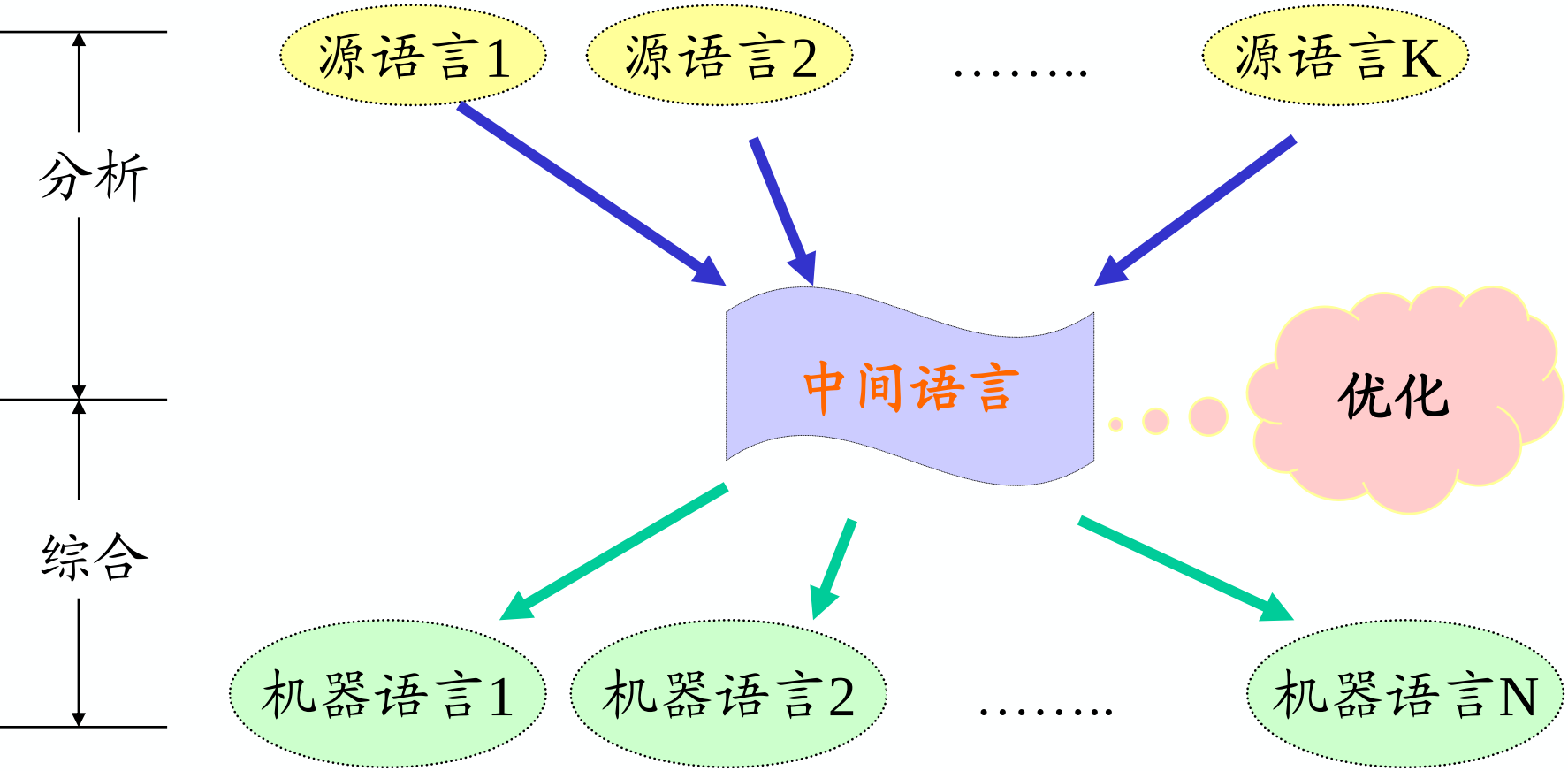


1.3.3 前端和后端

根据编译程序各部分功能，将编译程序分成前端和后端。

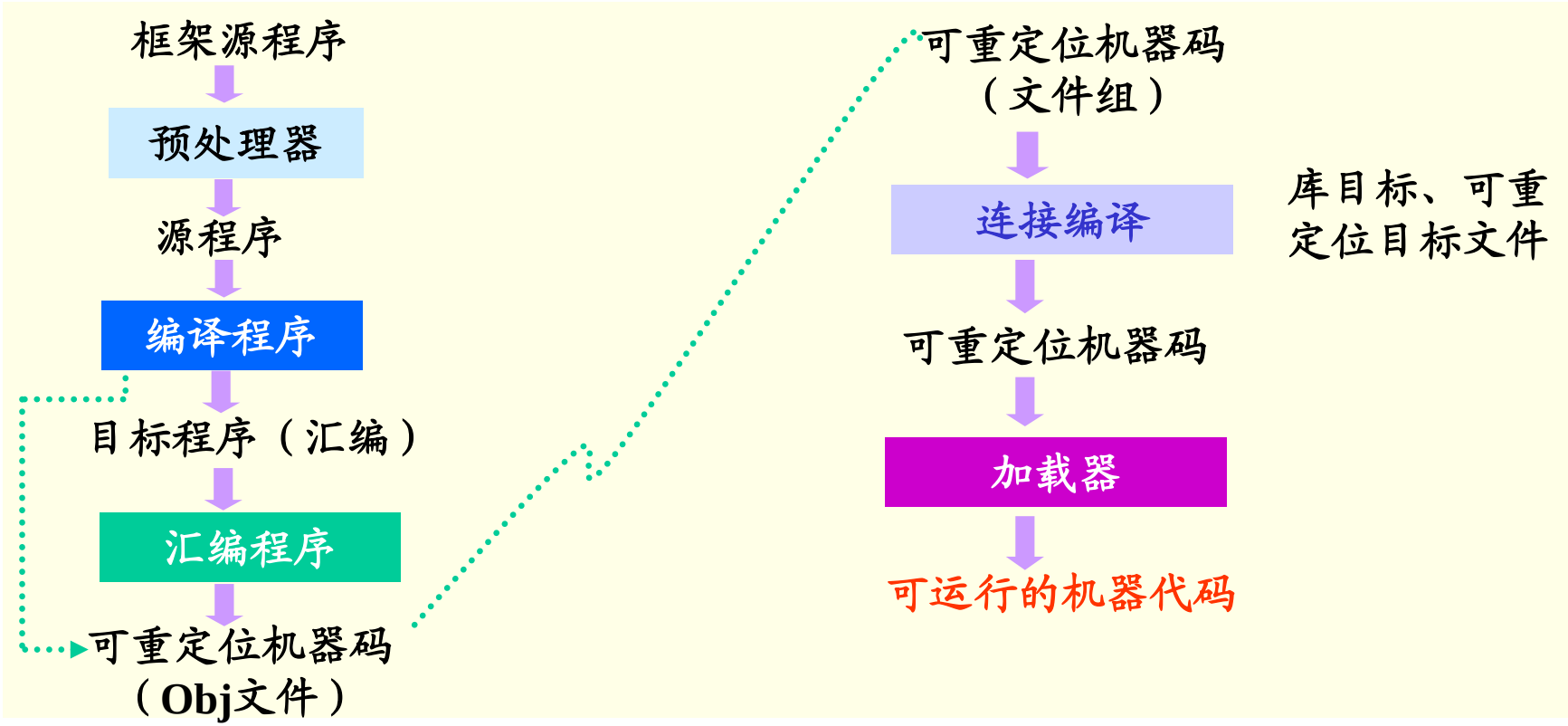
前端：通常将与源程序有关的编译部分称为前端。
词法分析、语法分析、语义分析、中间代码生成、代码优化 ————————分析部分
特点：与源语言有关

后端：与目标机有关的部分称为后端。
目标程序生成（与目标机有关的优化）
————————综合部分
特点：与目标机有关


分析
综合
源语言1
源语言2
……..
源语言K
中间语言
优化
机器语言1
机器语言2
……..
机器语言N

1.4 编译程序的前后处理器

源程序：多文件、宏定义和宏调用，包含文件
目标程序：一般为汇编程序或可重定位的机器代码



1.5 编译技术的应用

- 语法制导的结构化编译器
- 程序格式化工具
- 软件分析与测试工具
- 高级语言的翻译工具
- 程序理解工具
- 等等

1.5 编译技术的应用

程序理解工具：辅助理解别人写的程序

- 提取有价值的信息，展示给用户，辅助用户理解软件
- 准确提取的前提是设计自动分析软件，提供分析能力，进而准确分析

1.5 编译技术的应用

软件逆向工程工具：分析抽象，获取软件体系结构

软件 → **软件逆向工程** → 软件体系结构

- 逆向抽取模块设计，模块间接口
- 理解软件架构设计，功能划分、模块调用关系
- 维护现有软件
- 二次开发新软件

1.5 编译技术的应用

软件演化分析工具：分析软件演化规律

多个版本软件 → **软件演化分析** → 演化规律

- 分析软件，从结构、变量等多个角度分析
- 维护现有软件
- 捕捉规律，利用规律，开发新软件

1.5 编译技术的应用

软件自动并行化工具：并行化提升软件执行效率

分析软件，识别并行计算代码段

- 通过软件自动并行化工具，自动分析、找到并行程序段，分配到不同的处理器上，并行执行
- 什么时候可以并行，什么时候只能串行，还有很多待研究问题

1.5 编译技术的应用

软件错误定位工具：定位到有缺陷的程序语句

有缺陷的软件 → 软件错误定位 → 软件缺陷根因

提升开发效率

输入： 软件系统源码 + 一组测试，至少一个没有通过

输出： 一个可能有错误的程序元素列表，根据出错概率排序

程序元素的级别： 表达式/语句/方法/类/文件

1.5 编译技术的应用

软件自动修复工具：修复有缺陷的软件

提升开发效率

输入：一个程序和其正确性约束，并且程序不满足正确性约束

输入：一个补丁，可以使程序满足约束

研究和实践中考虑最广泛的正确性约束：软件项目中的测试

1.5 编译技术的应用

软件自动补全工具：补全不完整的代码

不完整的代码 → 软件自动补全 → 完整的代码

提升开发效率

- 基于智能化算法
- 简单的语句补全
- API自动补全（与功能相关）

第21页1、2、3、4、5题（新，高等教育出版社）

1. 解释下列名词术语：源程序、目标程序、翻译程序、汇编程序、编译程序、解释程序、遍。

2. 典型的编译程序可划分为哪几个主要的逻辑部分？各部分的主要功能是什么？

3. 什么是编译的前端和后端？为什么要把编译程序分成前端和后端？

4. 通过查资料和调研列举5个目前常用的高级程序设计语言，简述其特点，并指出是编译型还是解释性或编译解释型。

5. 介绍一个自己熟悉的编译系统，描述其外部特征（功能、性能等）。

回顾：

编译的起源

概念：源程序 目标程序 翻译程序 汇编程序 编译程序

编译过程：5个基本阶段

编译程序：7个逻辑部分

遍 前端 后端 前后处理器

- 教育：被教育者在这个过程中获得自信。
- 如何获得自信：参与教育、证明自己很棒---自信！！
- “痛并快乐着!!!”