

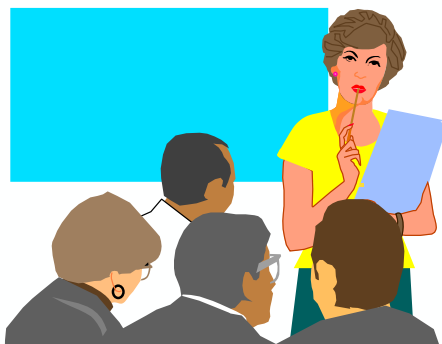
国家级精品课

国家级一流本科课程

国家级精品资源共享课MOOC

编译技术

编译原理及编译程序构造



史晚华

xhshi@buaa.edu.cn

新主楼G913

2023.9-2023.12

课程要求

★理论课时:48学时 (1 - 13周, 每周1、3)

实验机时: 48小时 (2 - 17周, 每周1)

★分为两部分:(分别计分)

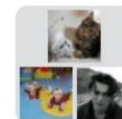
– 理论基础 (占60%) :课堂教学, 按时交作业。

• 按百分计: 作业10分(补交不超过6分);

• 3~5次随堂考试, 共计30分; (不提前通知, 不补)

• 期末闭卷考试, 60分

– 实践部分(占40%) : (2~17周上机)



群聊：编译技术-史晓华班 2023



该二维码7天内(9月11日前)有效，重新进入将更新

课堂要求

- 不许使用手机
- 不许使用笔记本电脑、平板电脑
 - 教室最后两排可以使用
- 不许交头接耳，影响他人听课

课程目的

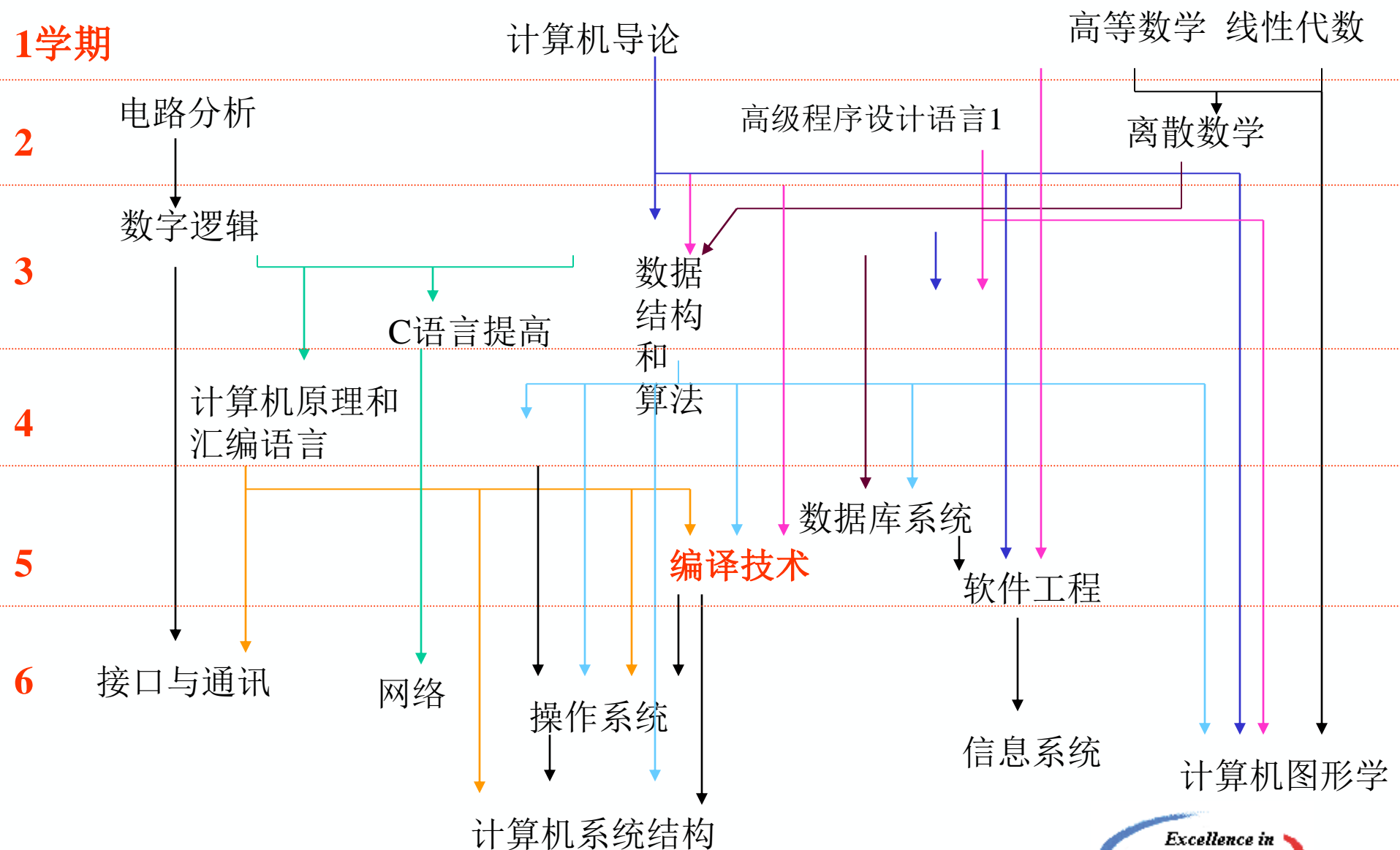
目的:

- 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（结构和原理）。
- 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。

为什么要掌握编译技术:

- 国家“**自主可控**”战略中，要解决“**卡脖子**”问题
- 如果我们有了自主可控的CPU、操作系统，以及各种关键软件，但是**编译器、虚拟机、编程语言标准等等，都没有掌握，被别人控制，谈何自主可控？**

分类	课程名称	课程定位	备注
计算机基础	计算机导论	入门	
	算法和数据结构	基础	
	高级语言程序设计（1，2）	必备工具	
计算机理论 （离散数学1,2,3）	数理逻辑	计算机数学	
	集合论和图论		
	组合数学		
计算机硬件类课程	数字电路和数字逻辑	硬件基础课程	含实验
	计算机原理和汇编语言	部件原理	含实验
	计算机接口与通讯	部件间通讯	含实验
	计算机体系结构	体系结构	含实验
	计算机网络		
计算机软件类课程	编译技术	系统软件	含课程设计
	操作系统		含课程设计
	数据库系统原理		含课程设计
	软件工程		
	信息系统分析与设计	应用类	
	计算机图形学（多媒体技术）	应用类	





• 教材和参考书

- 张莉、史晓华、杨海燕等, 《编译技术》, 高等教育出版社
- **龙书**: Alfred V. Aho, Monica S.Lam, Ravi Sethi, Compilers—Principles, Techniques, and Tools. 机械工业出版社 (3rd版), 2009
- **鲸书**: Steven S. Muchnick, Advanced Compiler Design and Implementation
- A. W. Apple, J. Palsberg 著, 《Modern Compiler Implementation in Java》
- Kenneth C. Loudon 著, 《编译原理及实践》, 机械工业出版社, 2000,3。

要求:

1. 提前预习，上课认真听讲；
课后及时复习，独立认真完成作业。
2. 每周一交作业，小班汇总后直接交给助教。

助教及答疑:

申浩佳，王宇奇

地点：待定（**交作业,准备2个作业本，周一晚上交给助教**）

网址：<http://judge.buaa.edu.cn>

网上答疑为主

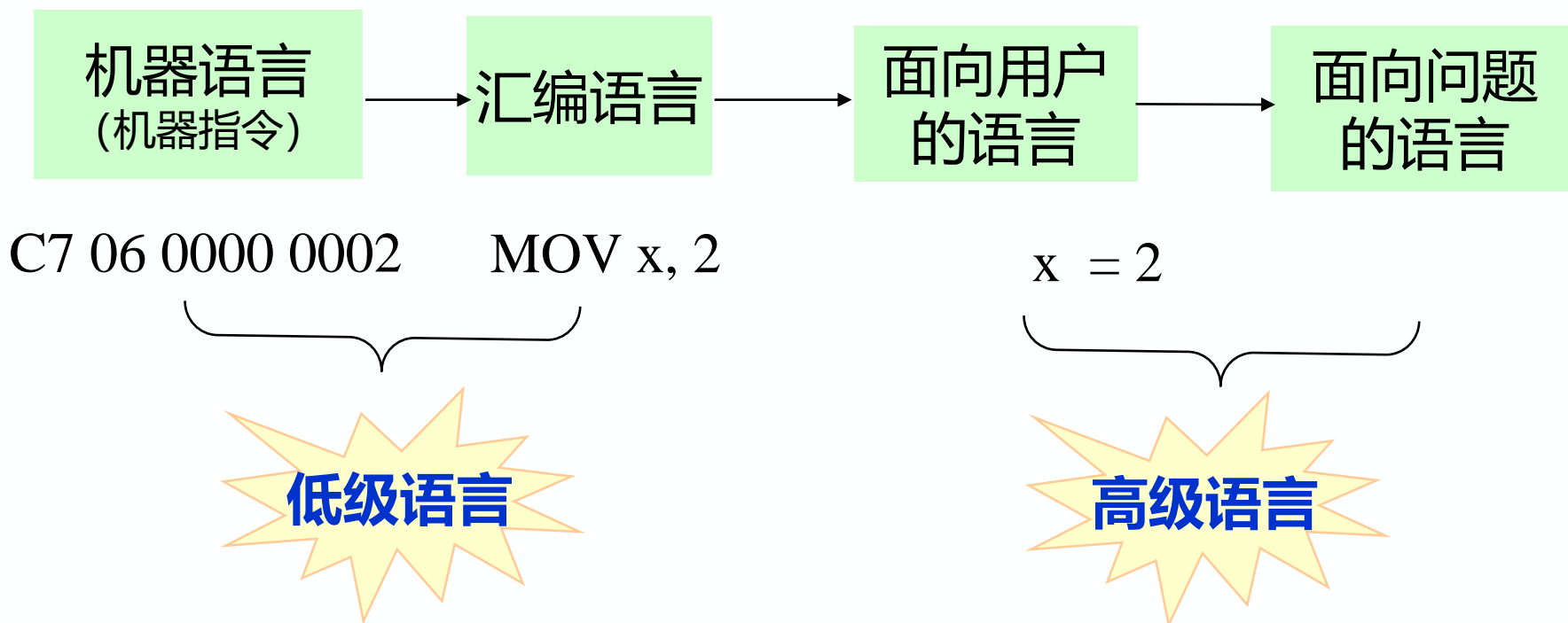
第一章 概论

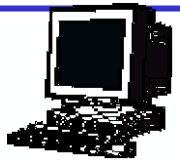
(介绍名词术语、了解编译系统的结构和编译过程)

- 编译的起源：程序设计语言的发展
- 基本概念
- 编译过程和编译程序构造
- 编译技术的应用



1.1 程序设计语言的发展





- **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，运行效率高，但使用和编程复杂、繁琐、费时、易出错**
- **高级语言**
 - Fortran、Pascal、C、Java 语言等
 - **特点：不依赖某个具体的硬件体系结构，可移植性好、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”。

1.2 基本概念

• 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

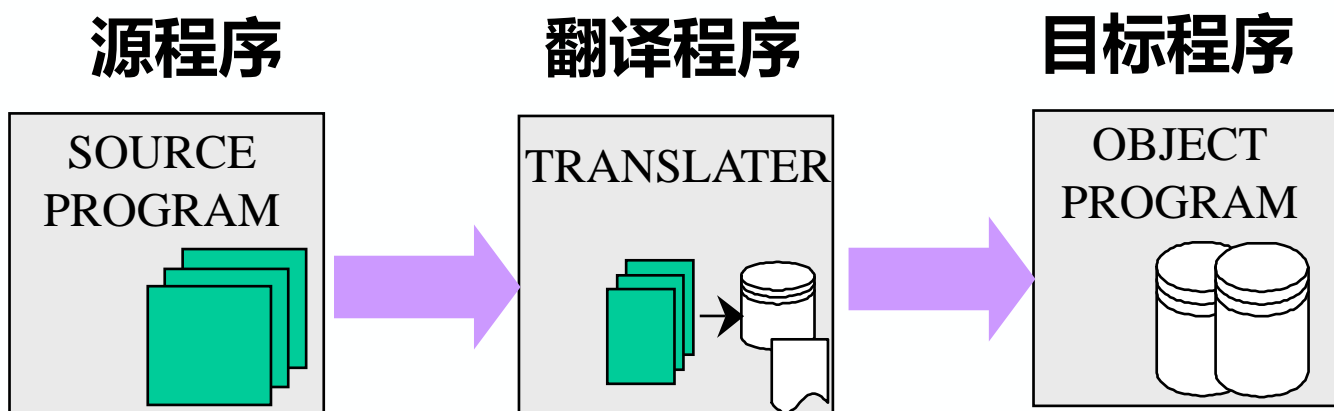
用**目标语言**所表示的程序。

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

源程序

汇编语言
高级语言

翻译程序

汇编程序
编译程序

目标程序

机器语言
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，这种翻译过程称“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得多。

源程序的编译和运行

- 编译或汇编阶段

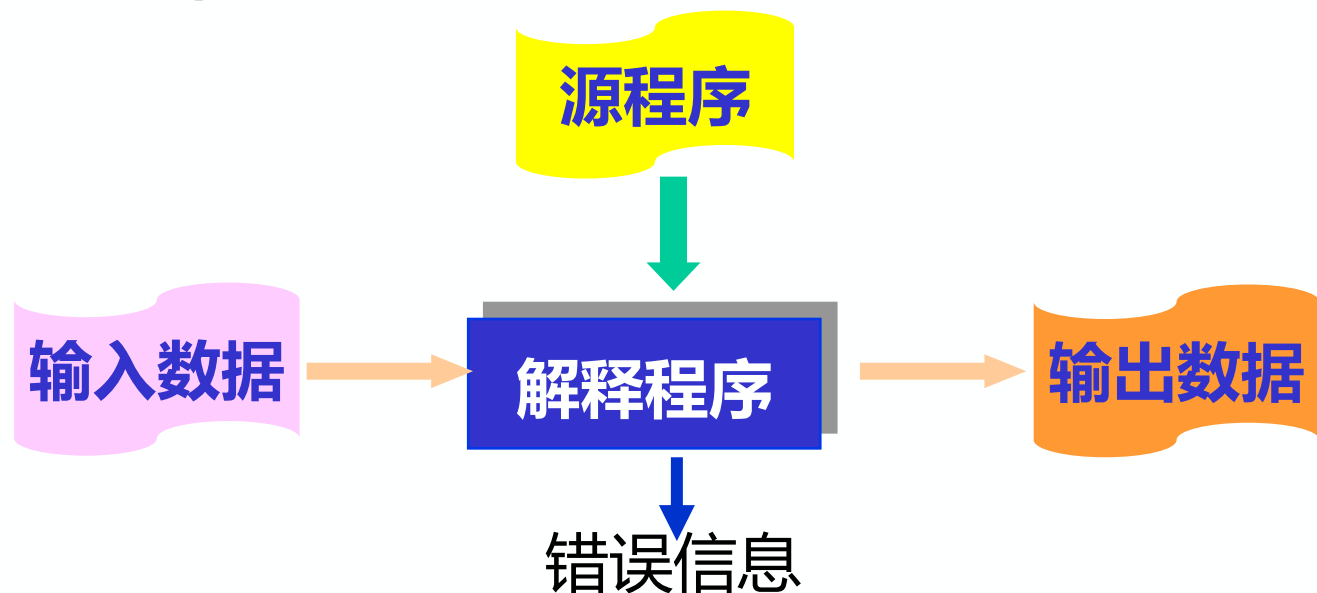


- 运行阶段



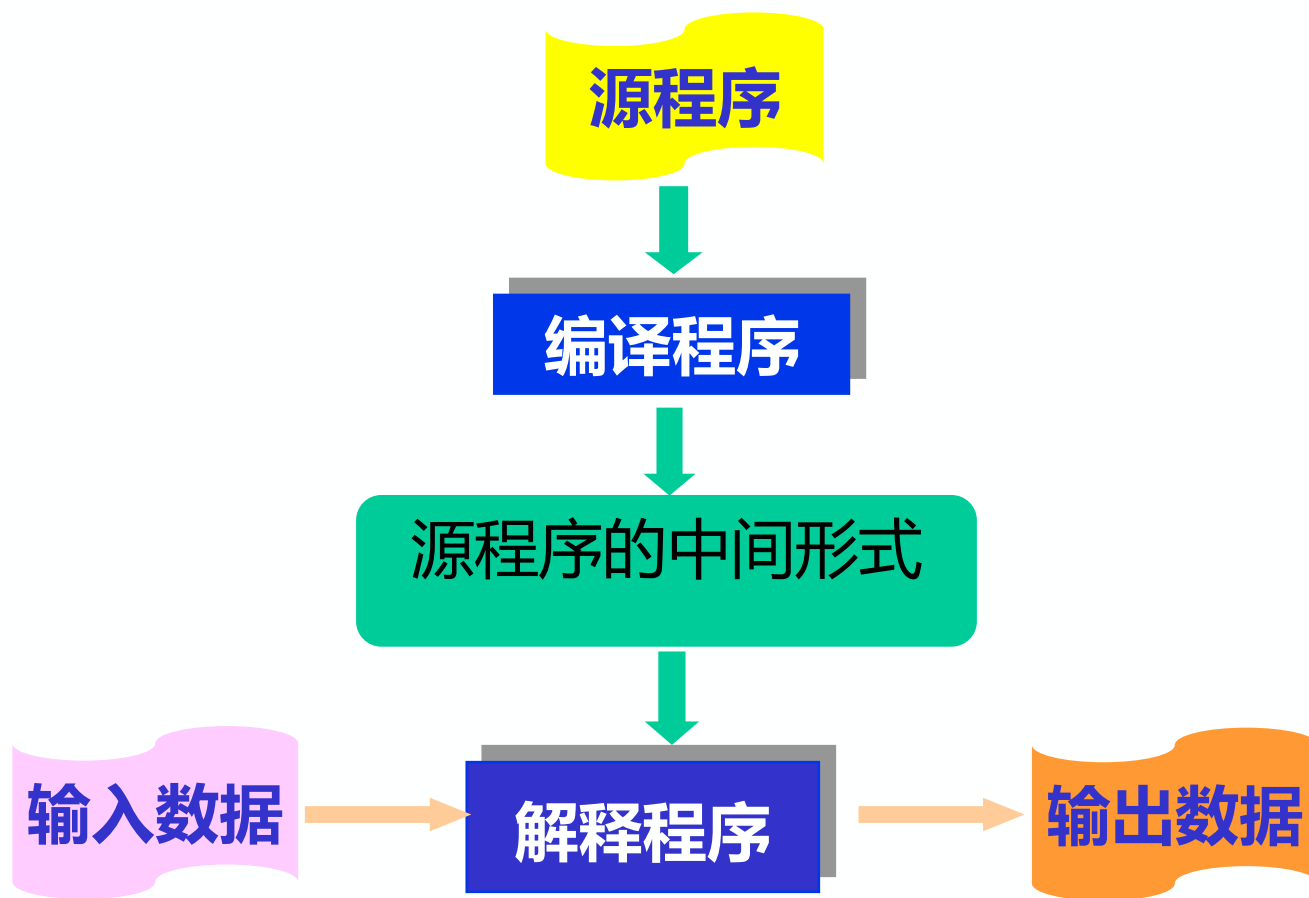
- **解释程序 (Interpreter)**
对源程序进行解释执行的程序。

- **工作过程**



- **特点、与编译程序比较**

“编译-解释执行”系统



1.3.1 编译过程



编译过程是指将**高级语言程序**翻译为语义等价的**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

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：

$$\frac{X1}{1} := \frac{(\frac{2.0}{2} + \frac{0.8}{5})}{3} * \frac{C1}{9}$$

————→ 也称为线性分析。

二、语法分析

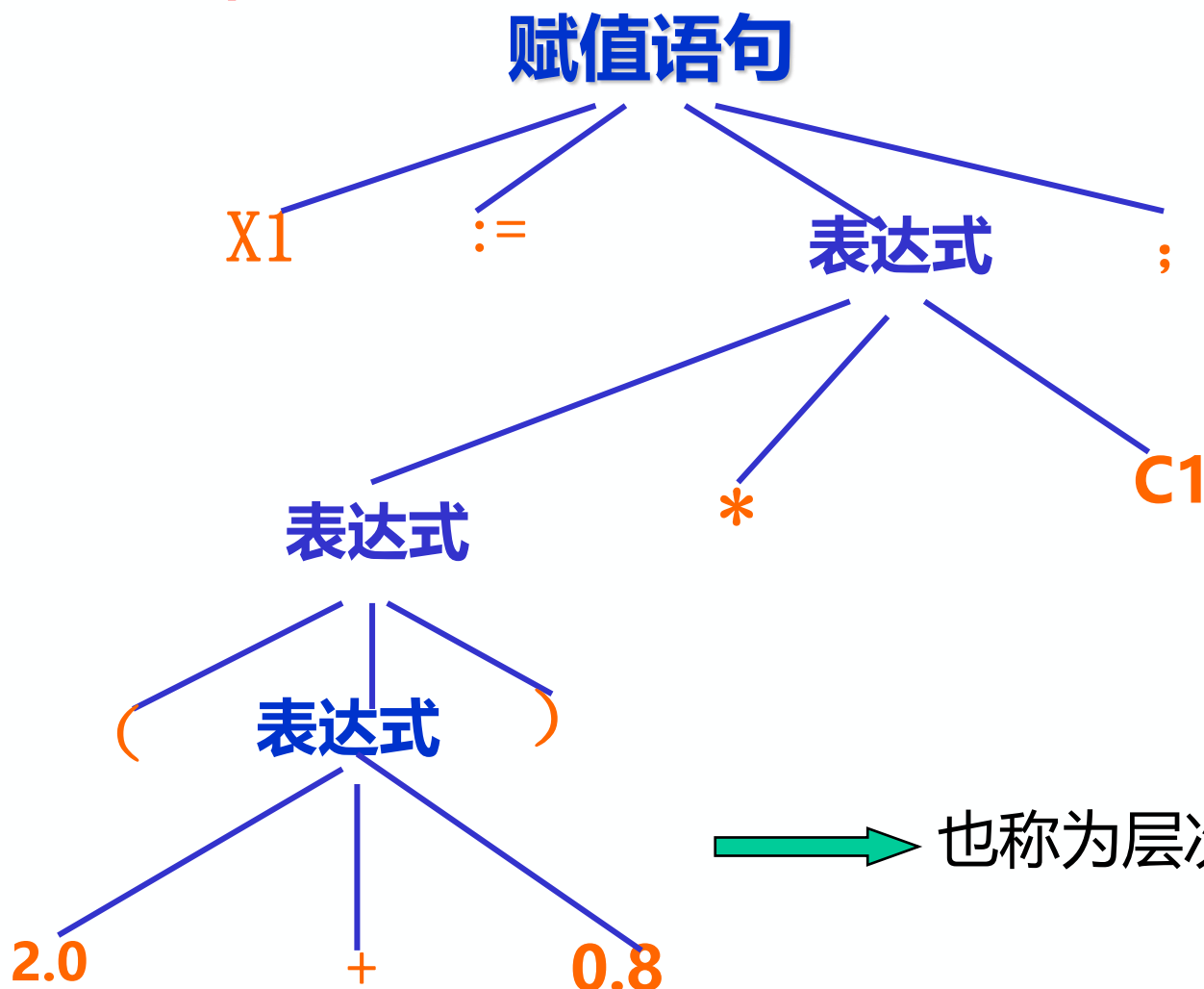
任务：根据**语法规则**（即语言的文法），分析并识别出各种语法成分，如表达式、各种说明、各种语句、过程、函数等，并进行语法正确性检查。

X1 := (2.0 + 0.8) * C1

赋值语句的文法：

<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> →

$X1 := (2.0 + 0.8) * C1;$



→ 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, 变量是否已被声明等。
- 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**)



★ 四元式 (三地址指令)

$X1 := (2.0 + 0.8) * C1$

	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工作单元

四元式的语义为: $T1 = 2.0 + 0.8$

$T2 = T1 * C1$

$X1 = T2$

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。



任务：目的是为了得到高质量的目标程序。

例如：前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工作单元中，不必生成目标指令来计算，这样四元式可优化为：

优化前的四元式：

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

优化为： $2.0 + 0.8 \rightarrow 2.8$

(1)	*	2.8	C1	X1
-----	---	-----	----	----

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，所以要根据机器进行。在做这部分工作时（要注意充分利用目标机特性），也可以进行优化处理。

$X1 := (2.0 + 0.8) * C1$

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

作利用累
加器的优化



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

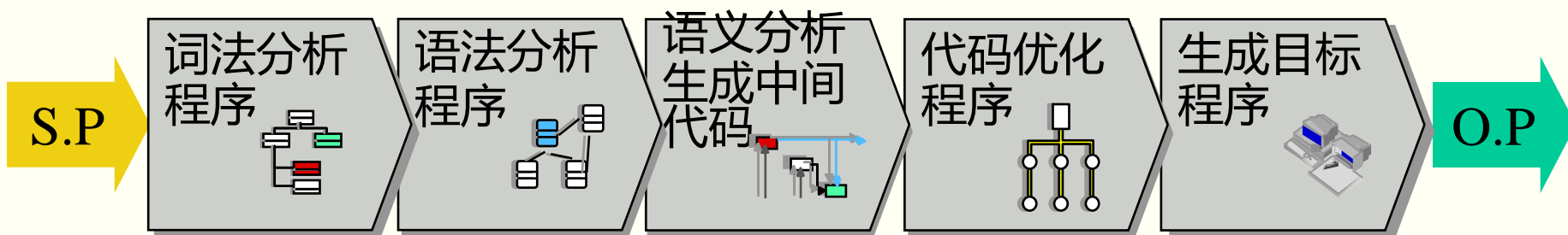
注意：在翻译成目标程序的过程中，
要切记保持语义的等价性。

1.3.2 编译程序构造



一、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符号表管理和出错处理两部分

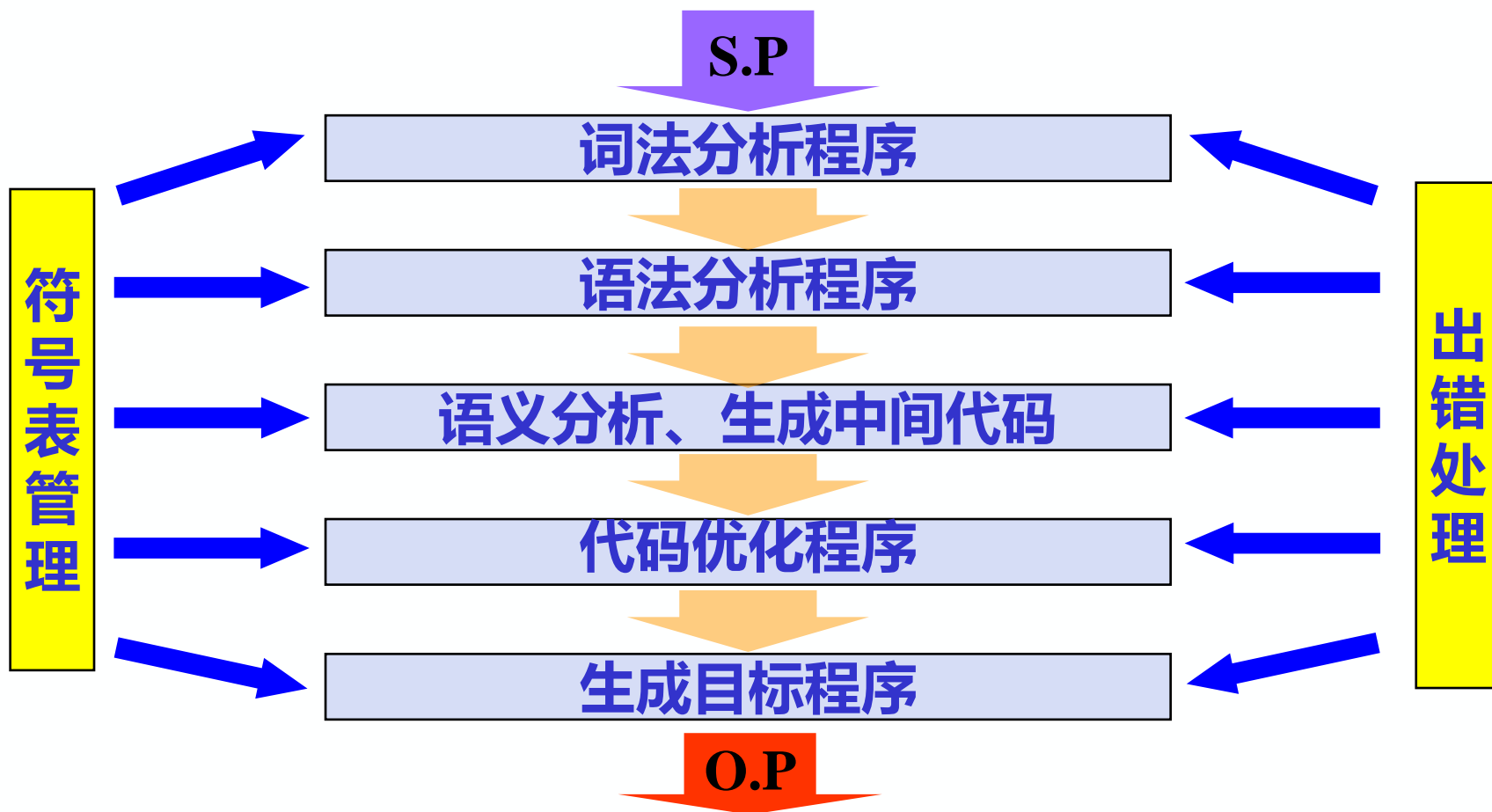
★ 符号表管理

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

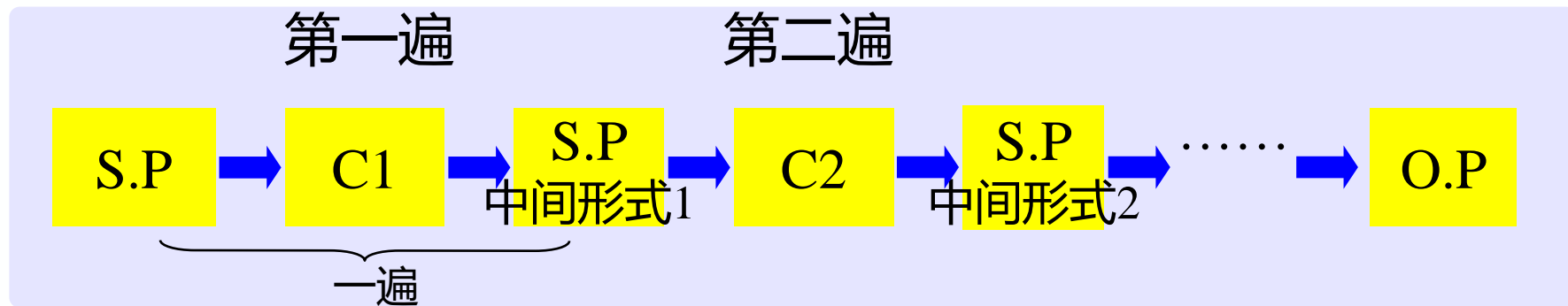
规模较大的源程序难免有多种错误，编译程序必须要有出错处理的功能。即能诊察出错误，并能报告用户错误的性质和位置，以使用户修改源程序。出错处理能力的大小是衡量编译程序质量好坏的一个重要指标。

典型的编译程序具有7个逻辑部分



二、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**。



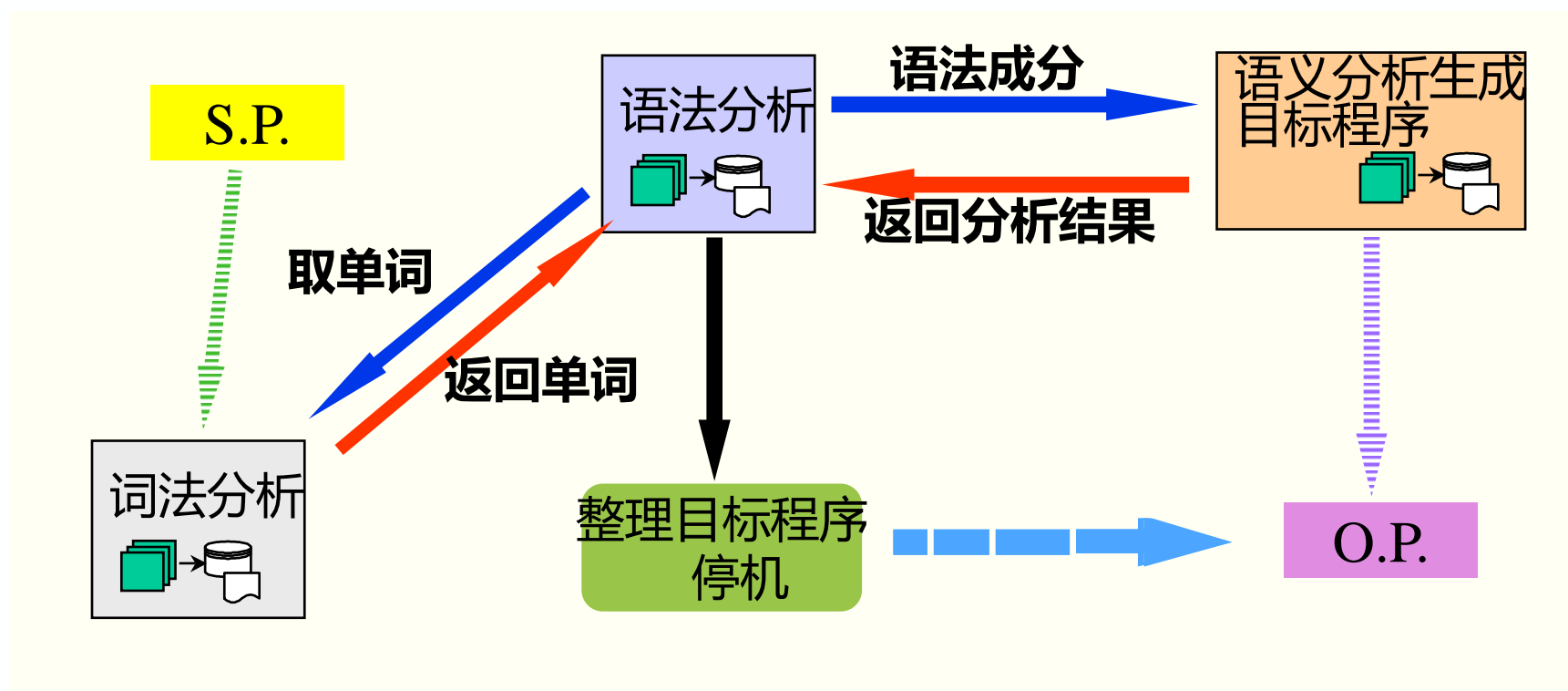
★ 要注意遍与基本阶段的区别

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：



三、前端和后端

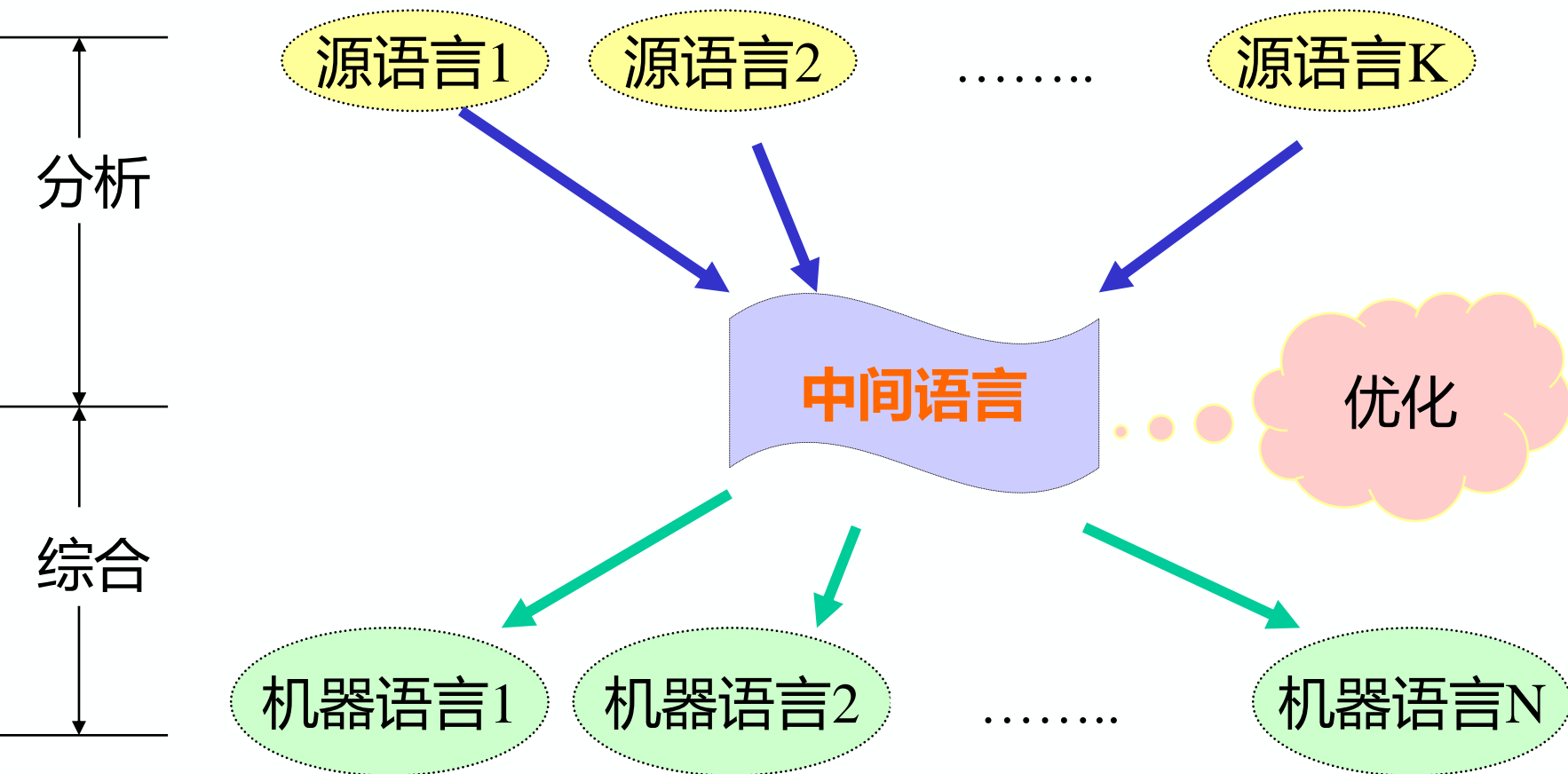
根据编译程序各部分功能，将编译程序分成**前端**和**后端**。

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

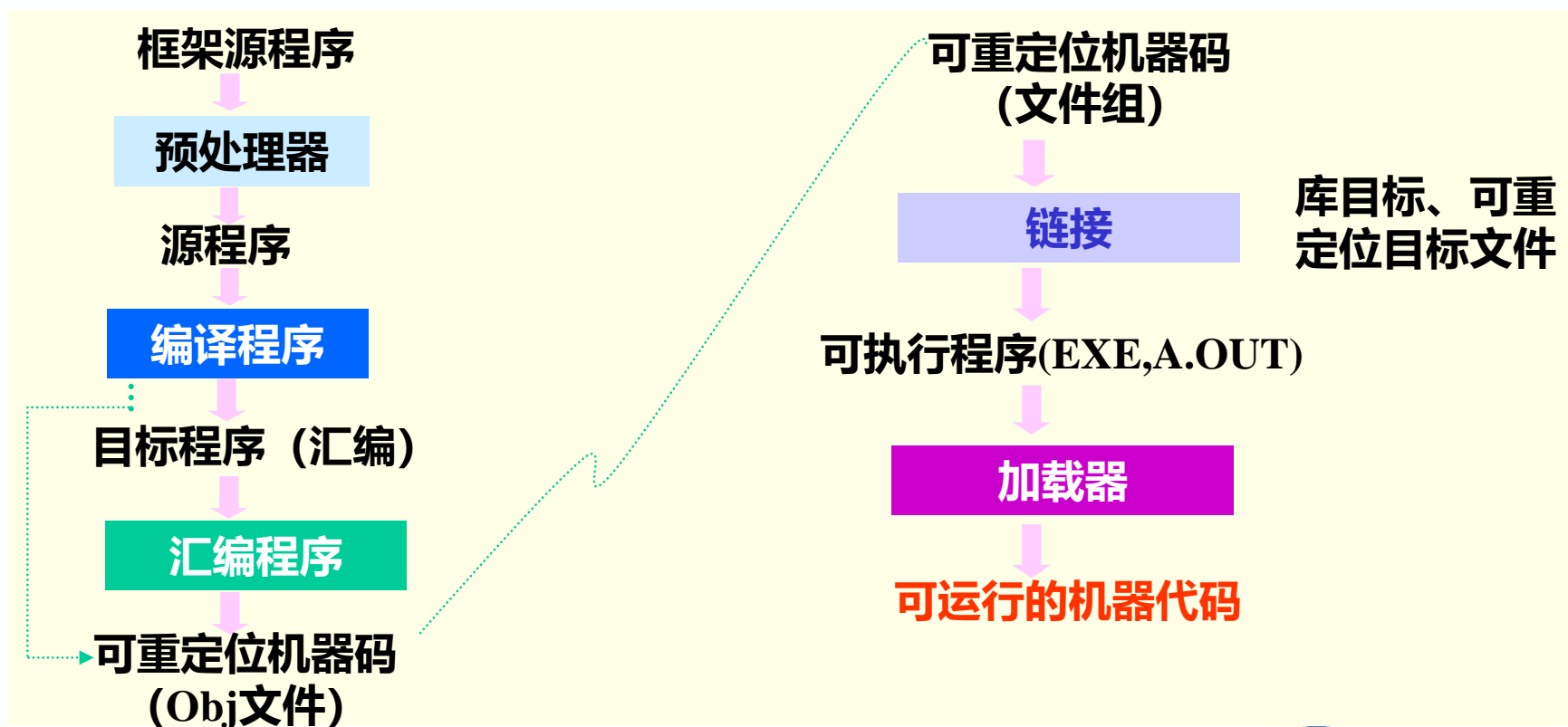
后端：与目标机有关的部分称为后端。
代码优化、目标代码生成
-----综合部分

特点：与**目标机**有关



四、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



2020~2023年全国大学生编译技术大赛

- 2023年北航获得特等奖1项（全国共2名），一等奖1项（全国共4名），二等奖多项
- 2020年全国编译大赛开赛至今，北航每年都至少获得1项1等奖（或特等奖）和多项2等奖、3等奖

样例	7.23 通过全部 样例	强度削弱、 死代码删 除等	冗余代码 删除	常量传播、 小数组常 量化	图着色、 公共子表 达式删除	函数内联、 强化死代 码删除	循环优化	gcc-O2
bitset3	18.44	9.65	7.43	6.88	6.33	6.16	1.96	3.669
mm1	12.68	9.49	7.85	7.35	7.36	6.67	5.62	4.958
mv1	8.45	7.39	6.12	6.01	5.87	5.50	4.36	1.171
sort2	65.34	24.84	22.22	19.1	13.69	13.29	11.29	13.274
spmv1	6.31	4.29	3.68	3.38	3.36	3.29	3.67	2.77
	186.95s	102.12s	79s	75s	69s	65s	49s	

1.4 编译技术的应用

- ≈ 语法制导的结构化编辑器
- ≈ 程序格式化工具
- ≈ 软件测试工具
- ≈ 程序理解工具
- ≈ 高级语言的翻译工具
- ≈ 等等。

作业：第21页1、2、3、4题

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

教学意义——《编译原理》是一门非常好的课程

- **Alfred V.Aho**: 编写编译器的原理和技术具有十分普遍的意义, 以至于在每个计算机科学家的研究生涯中, 本书中的原理和技术都会反复用到。
- 涉及的是一个比较适当的抽象层面上的数据变换 (既抽象, 又实际) 。
- 一些具体的表示和变换算法。
- “自顶向下的方法” 和 “自底向上的方法” 系统设计方法 (思想、方法、实现全方位讨论)
- 一个相当规模的系统的设计 (含总体结构) 。
- 计算机专业最为恰当、有效的知识载体之一。

- 掌握编译程序**总体结构**
- 在**系统级**上认识算法、系统的设计
 - 具有把握系统的能力
- 学习有关的原理、实现方法和技术，了解计算学科的基本方法、思想
 - **掌握典型方法。** “在每一个计算机科技工作者的职业生涯中，这些原理和技术都被反复用到。”
- 兼顾语言的描述方法、设计、应用——**形式化**
 - 能形式化就能自动化
- 进一步培养 “**计算机思维能力**”
 - 软件系统的非物理性质

学习成果_以学生为中心

- 理解和掌握编译过程各个阶段的工作原理
- 理解标准编译器各个组成部分的任务
- 熟悉编译过程各阶段所要解决的问题及其采用的方法和技术
- 应用所学的技术解决编译器构造过程中所产生的相关问题
- 理解编译器在生成代码时如何充分利用特定处理器的特征

教、学方法

- **教学方法**

- **整体性——从系统的角度**
- **启发式——以学生为中心**
- **应用驱动——技术、方法的应用背景**

- **学习方法**

- **源程序是源泉，实践是手段。**
- **把每个阶段放到整个编译程序背景中学习**
- **认真做作业**
- **编程序**

2010~2020编程语言的发展

- 2010年TIOBE排名

- TIOBE排行榜是根据互联网上有经验的程序员、课程和第三方厂商的数量，并使用搜索引擎（如Google、Bing、Yahoo!）以及Wikipedia、Amazon、YouTube统计出排名数据，只是反映某个编程语言的热门程度，并不能说明一门编程语言好不好，或者一门语言所编写的代码数量多少
- 2010年居于榜首的十大编程语言分别为：Java、C、C++、PHP、Visual Basic、C#、Python、Objective-C、Perl和Ruby

2020年8月TIOBE编程语言排名

Aug 2020	Aug 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	16.98%	+1.83%
2	1	▼	Java	14.43%	-1.60%
3	3		Python	9.69%	-0.33%
4	4		C++	6.84%	+0.78%
5	5		C#	4.68%	+0.83%
6	6		Visual Basic	4.66%	+0.97%
7	7		JavaScript	2.87%	+0.62%
8	20	▲	R	2.79%	+1.97%
9	8	▼	PHP	2.24%	+0.17%
10	10		SQL	1.46%	-0.17%
11	17	▲	Go	1.43%	+0.45%
12	18	▲	Swift	1.42%	+0.53%
13	19	▲	Perl	1.11%	+0.25%
14	15	▲	Assembly language	1.04%	-0.07%
15	11	▼	Ruby	1.03%	-0.28%
16	12	▼	MATLAB	0.86%	-0.41%
17	16	▼	Classic Visual Basic	0.82%	-0.20%
18	13	▼	Groovy	0.77%	-0.46%
19	9	▼	Objective-C	0.76%	-0.93%
20	28	▲	Rust	0.74%	+0.29%

https://blog.csdn.net/linet_14521559

1. C
2. Java
3. Python
4. C++
5. C#
6. VB
7. JavaScript
8. R
9. PHP
10. SQL