编译原理

第一章 引论

第一章 引论

- 什么是编译程序
- 编译过程
- 编译程序的结构
- 编译程序与程序设计环境
- 编译程序的生成

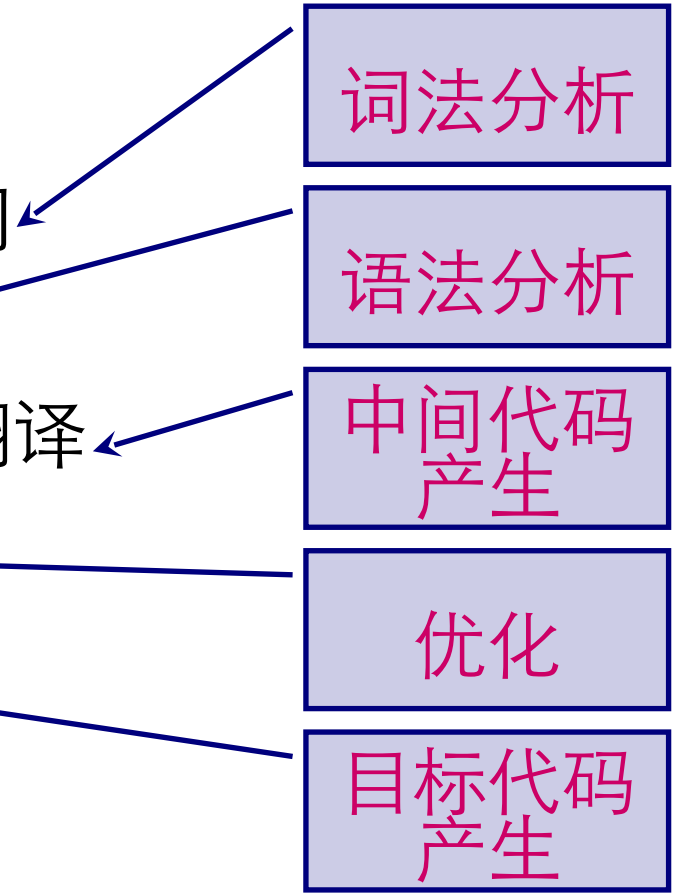
第一章 引论

- 什么是编译程序
- 编译过程
- 编译程序的结构
- 编译程序与程序设计环境
- 编译程序的生成

编译过程

- □□□□□□□□
  - 识别出句子中的一个个单词
  - 分析句子的语法结构
  - 根据句子的含义进行初步翻译
  - 对译文进行修饰
  - 写出最后的译文

编译程序工作的五个阶段



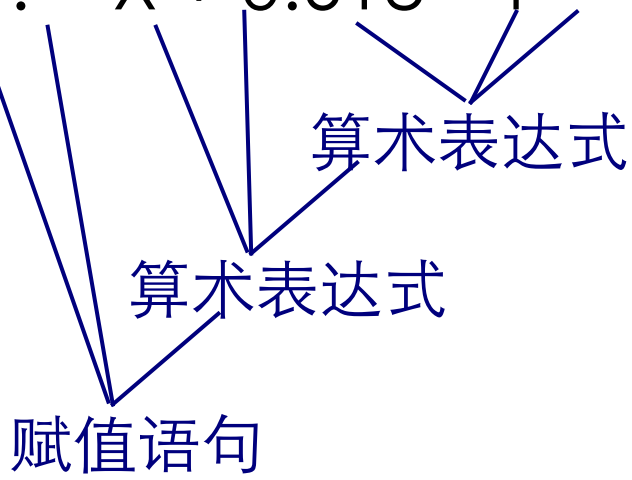
编译过程——词法分析

- 任务：输入源程序，对构成源程序的字符串进行扫描和分解，识别出一个个单词符号
- 依循的原则：构词规则
- 描述工具：有限自动机
- FOR　I　:=　1　TO　100　DO

  基本字　标识符　赋值号　整常数　基本字　整常数　基本字

编译过程——语法分析

- 任务：在词法分析的基础上，根据语言的语法规则把单词符号串分解成各类语法单位（语法范畴）
- 依循的原则：语法规则
- 描述工具：上下文无关文法
- Z := X + 0.618 * Y



编译过程——中间代码产生

- 任务：对各类不同语法范畴按语言的语义进行初步翻译
- 依循的原则：语义规则
- 中间代码：三元式，四元式，树，…
- Z:=X + 0.618 * Y 翻译成四元式为

(1) * 0.618 Y T1
(2) + X T1 T2
(3) := T2 _ Z

编译过程——优化

- 任务：对于前阶段产生的中间代码进行加工变换，以期在最后阶段产生更高效的目标代码
- 依循的原则：程序的等价变换规则

```
FOR K:=1 TO 100 DO
BEGIN
    X:=I+1;
    M := I + 10 * K;
    N := J + 10 * K;
END
```

编译过程：优化示例

```
FOR K:=1 TO 100 DO
BEGIN
      X:=I+1;
      M := I + 10 * K;
      N := J + 10 * K;
END
```

序号	OPR	OPN1	OPN2	RESULT	注释
(1)	:=	1		K	K:=1
(2)	j<	100	K	(10)	if (100<K)
					to
(10)					
(3)	+	I	1		=I+1
(4)	*	10	K	T1	T1:=10*K
(5)	+	I	T1	M	M:=I+T1
(6)	*	10	K	T2	T2:=10*K
(7)	+	J	T2	N	N:=J+T2
(8)	+	K	1	K	K:=K+1
(9)	j			(2)	goto (2)
(10)					

400 次加法

200 次乘法

编译过程：
转换后的等价代码

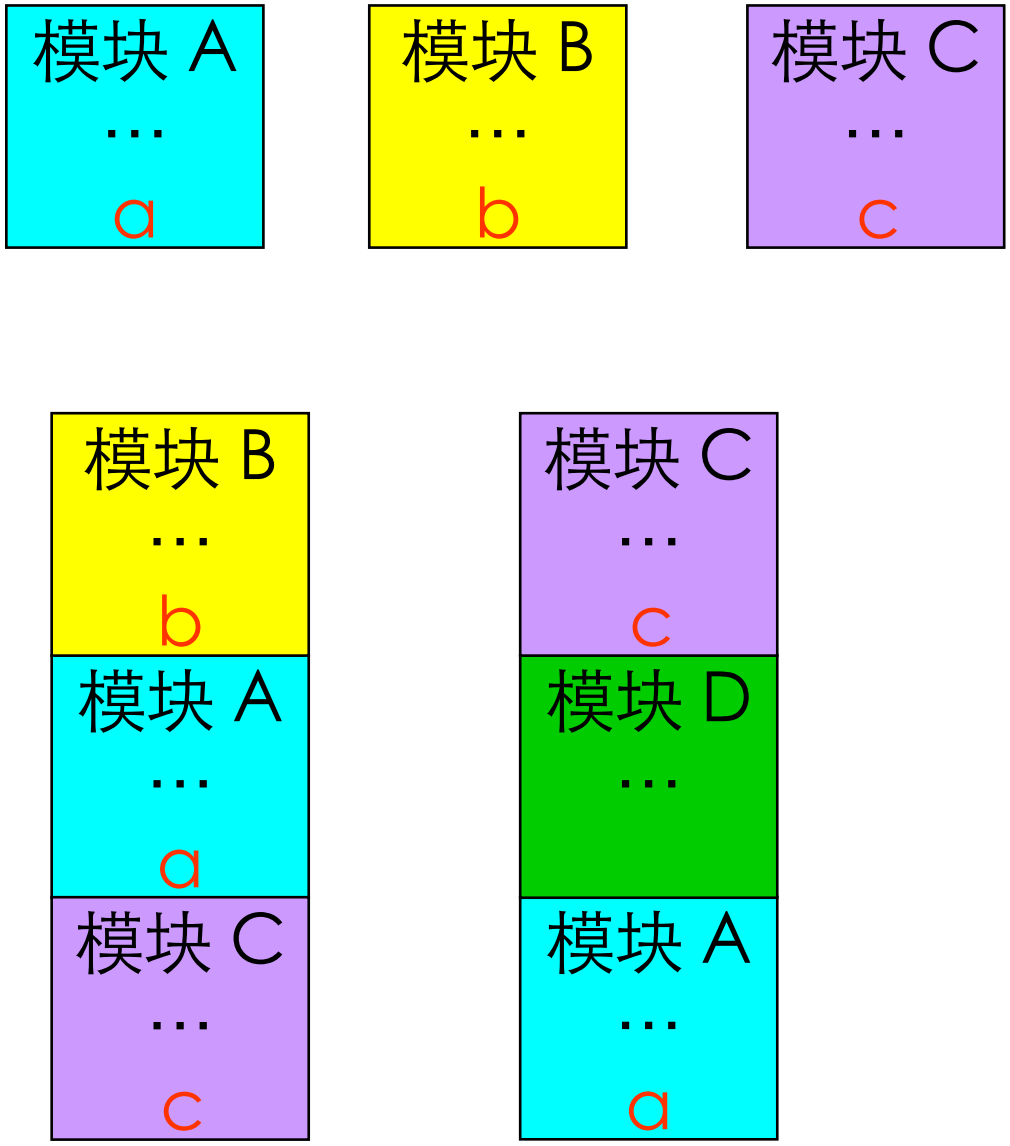
```
FOR K:=1 TO 100 DO
BEGIN
      X:=I+1;
      M := I + 10 * K;
      N := J + 10 * K;
END
```

序号	OPR	OPN1	OPN2	RESULT	注释
(1)	+	I	1	X	X:=I+1
(2)	:=	I		M	M:=I
(3)	:=	J		N	N:=J
(4)	:=	1		K	K:=1
(5)	j<	100	K	(10)	if (100<K) goto (10)
(6)	+	M	10	M	M:=M+10
(7)	+	N	10	N	N:=N+10
(8)	+	K	1	K	K:=K+1
(9)	j			(5)	goto (5)
(10)					

301 次加法

编译过程——目标代码产生

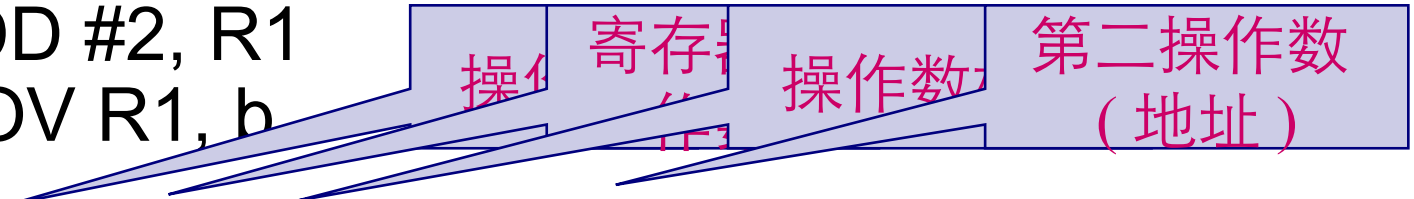
- 任务：　把中间代码变换成特定机器上的目标代码
- 依赖于硬件系统结构和机器指令的含义
- 目标代码三种形式
  - 绝对指令代码：可直接运行
  - 可重新定位指令代码：需要连接装配
  - 汇编指令代码：需要进行汇编


模块 A
…
a
模块 B
…
b
模块 C
…
c
模块 B
…
b
模块 A
…
a
模块 C
…
c
模块 C
…
c
模块 D
…
模块 A
…
a

编译过程——目标代码产生

- 例： b=a+2

```
MOV a, R1
ADD #2, R1
MOV R1, b
```




```
0001  01  00  00000000 *
0011  01  10  00000010
0100  01  00  00000100 *

L=00001111
0001  01  00  00001111
0011  01  10  00000010
0100  01  00  00010011
```

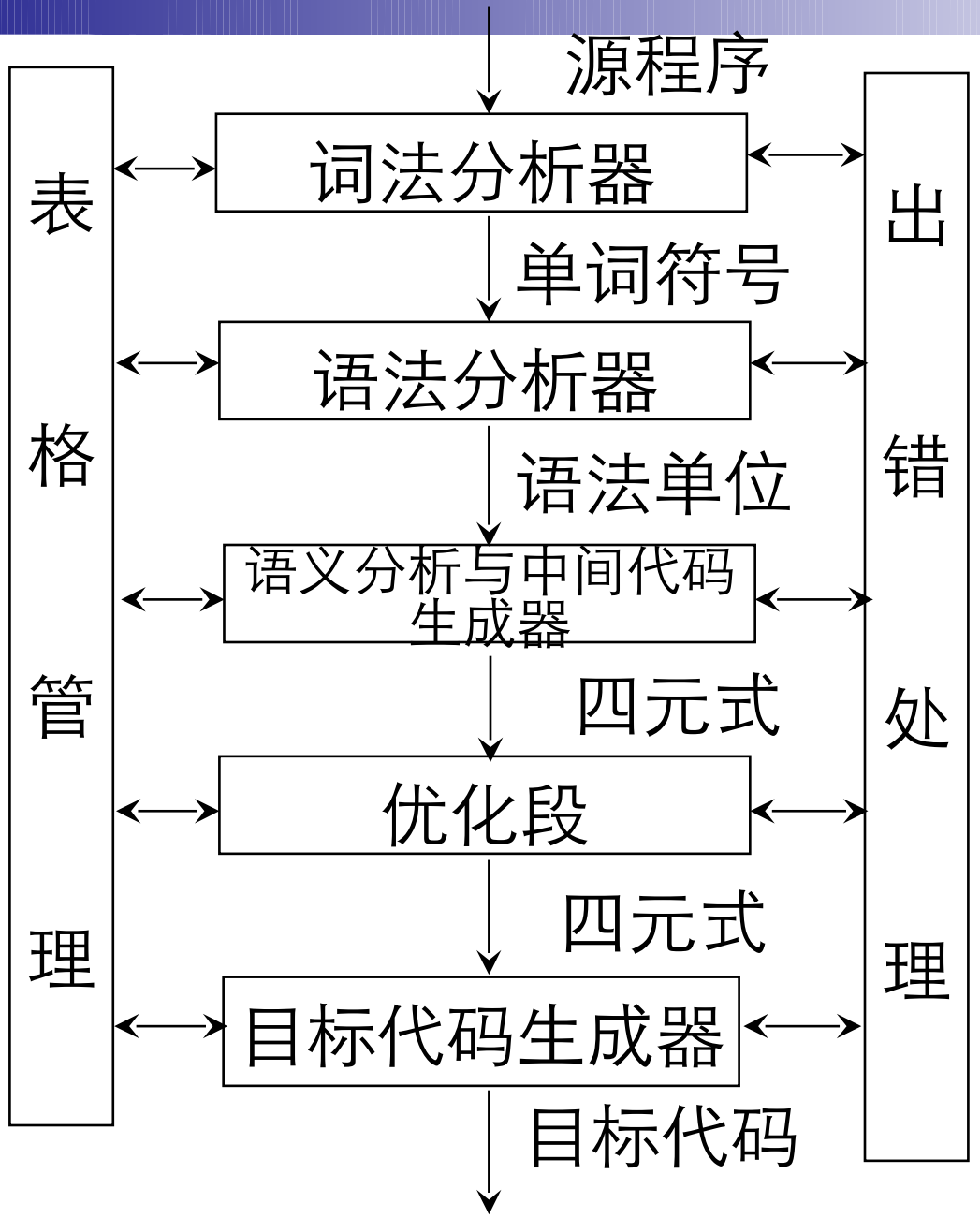
第一章 引论

- 什么是编译程序
- 编译过程
- 编译程序的结构
- 编译程序与程序设计环境
- 编译程序的生成

编译程序的结构

- 编译程序总框
- 表格与表格管理
- 出错处理
- 遍
- 编译前端与后端

编译程序总框


源程序
词法分析器
单词符号
语法分析器
语法单位
语义分析与中间代码生成器
四元式
优化段
四元式
目标代码生成器
目标代码
表格管理
出错处理

表格和表格管理

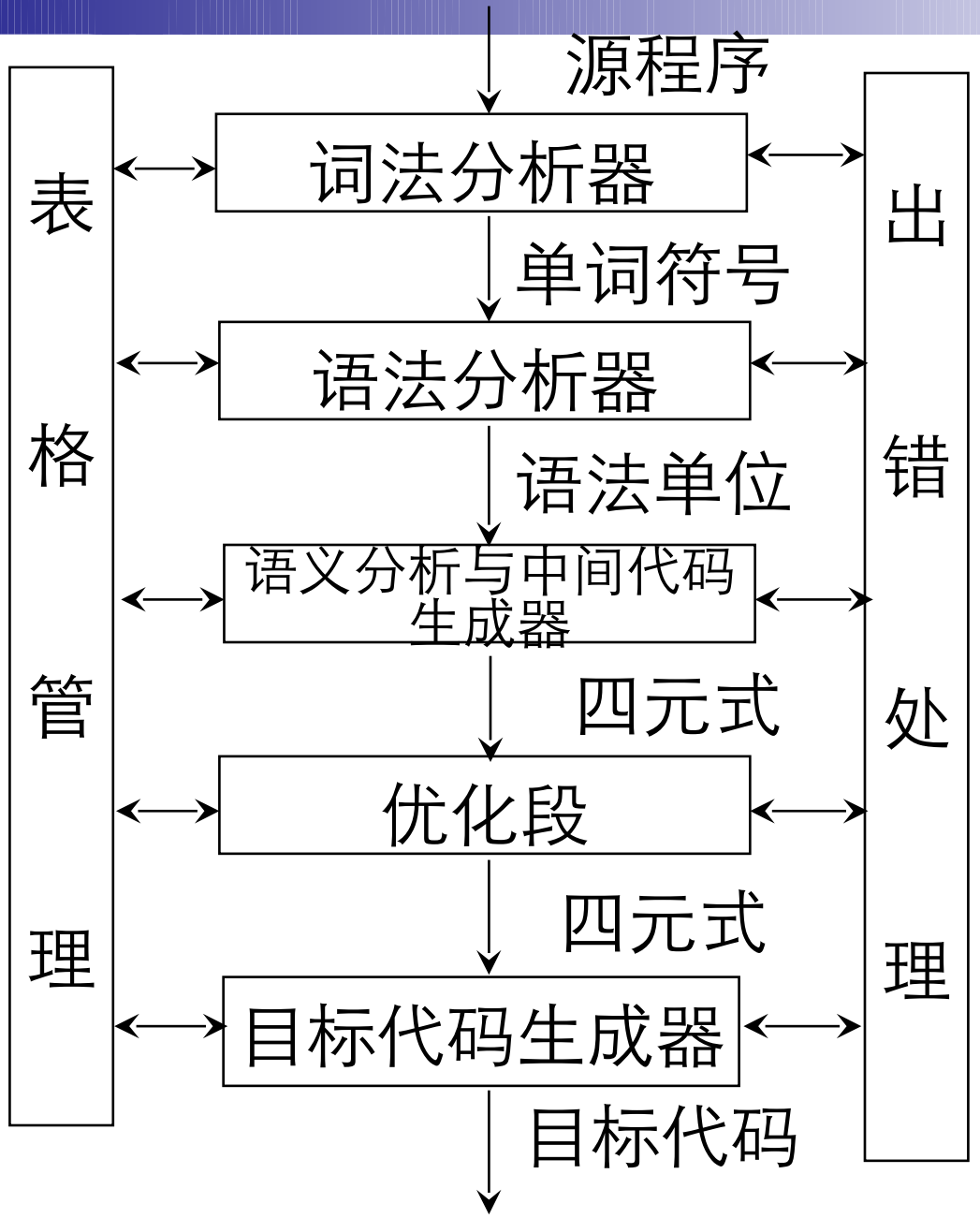
- 常见的表格
  - 符号名表，常数表，标号表，入口名表，过程引用表 ,...
- 格式

- 详见第 8 章符号表

出错处理

- 出错处理程序
  - 发现源程序中的错误，把有关错误信息报告给用户
- 语法错误
  - 源程序中不符合语法（或词法）规则的错误
  - 非法字符、括号不匹配、缺少；、 ...
- 语义错误
  - 源程序中不符合语义规则的错误
  - 说明错误、作用域错误、类型不一致、 ...

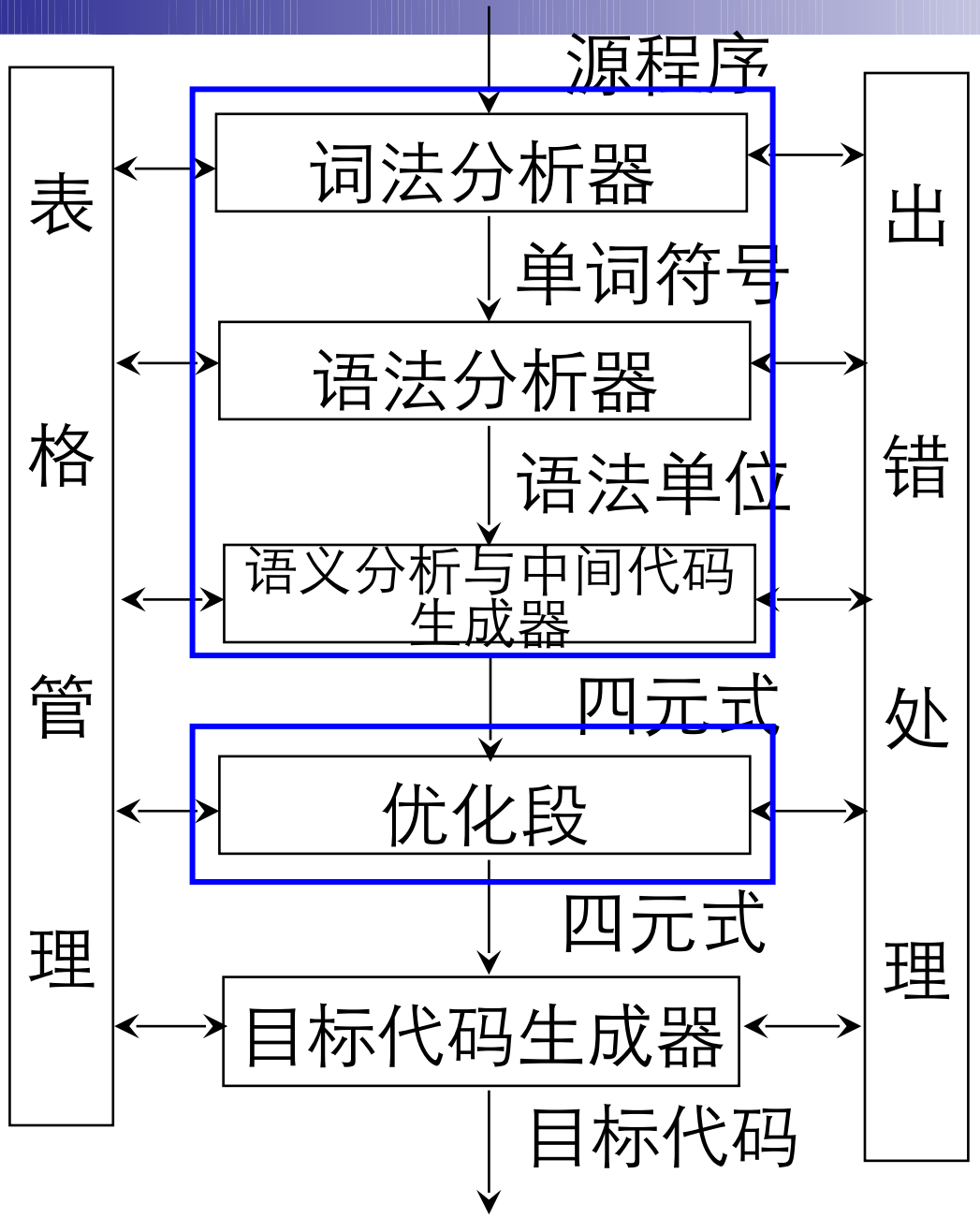
编译程序总框


源程序
词法分析器
单词符号
语法分析器
语法单位
语义分析与中间代码生成器
四元式
优化段
四元式
目标代码生成器
目标代码
表格管理
出错处理

遍 (pass)

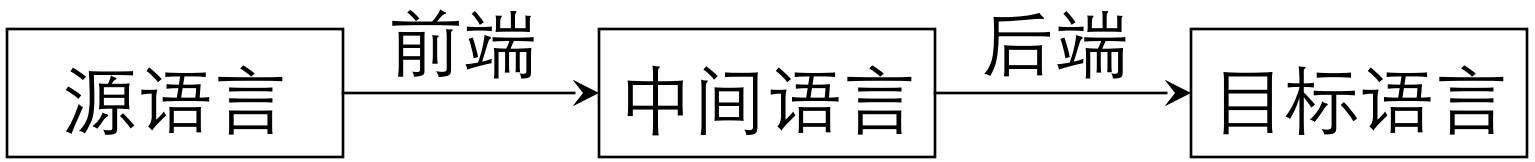
- 所谓"遍"， 就是对源程序或源程序的中间表示从头到尾扫描一次
- 阶段与遍是不同的概念
  - 一遍可以由若干段组成
  - 一个阶段也可以分若干遍来完成

编译程序总框


源程序
词法分析器
单词符号
语法分析器
语法单位
语义分析与中间代码生成器
四元式
优化段
四元式
目标代码生成器
目标代码
表格管理
出错处理

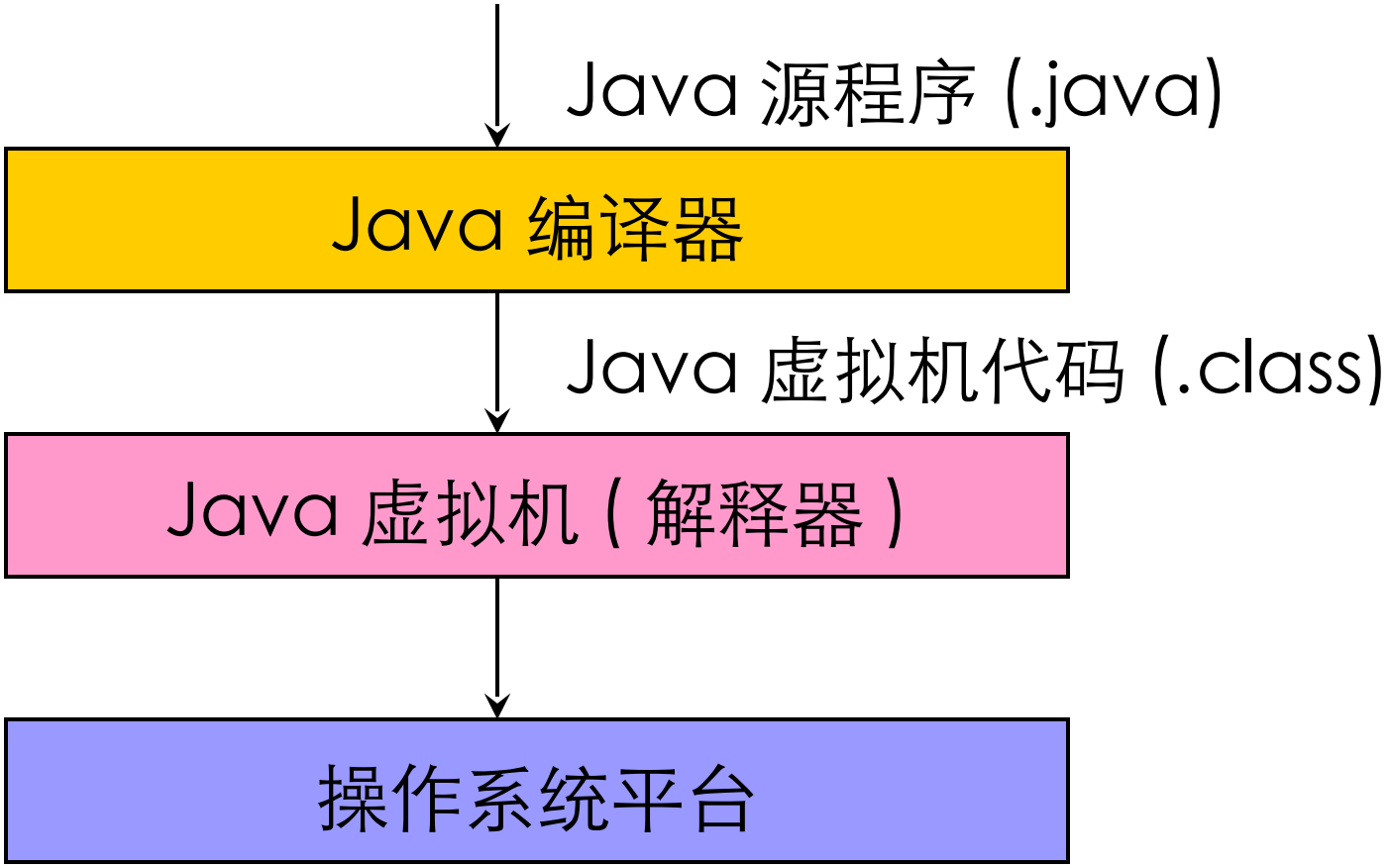
- 计算思维
  - 分解
  - 权衡

编译前端与后端



- 编译前端
  - 与源语言有关，如词法分析，语法分析，语义分析与中间代码产生，与机器无关的优化
- 编译后端
  - 与目标机有关，与目标机有关的优化，目标代码产生
- 带来的好处
  - 程序逻辑结构清晰
  - 优化更充分，有利于移植

JAVA 语言



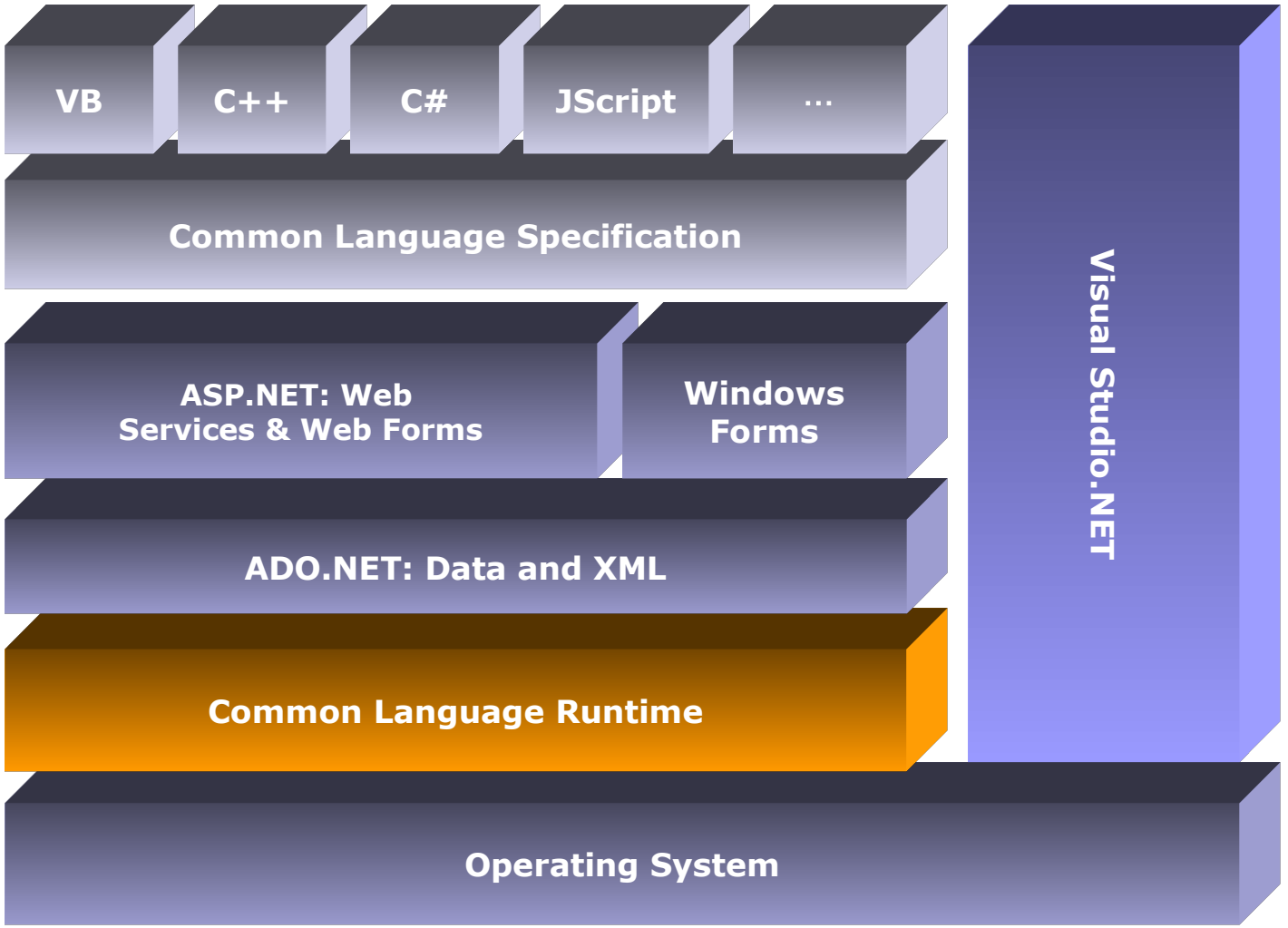
第一章 引论

- 什么是编译程序
- 编译过程
- 编译程序的结构
- 编译程序与程序设计环境
- 编译程序的生成

编译程序与程序设计环境

- 程序设计环境
  - 编辑程序
  - 编译程序
  - 连接程序
  - 调试工具
- 集成化的程序设计环境

.NET Framework 与 VS.NET



第一章 引论

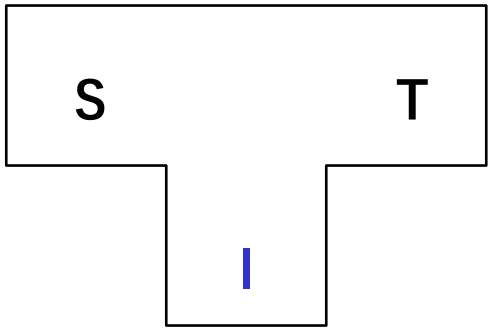
- 什么是编译程序
- 编译过程
- 编译程序的结构
- 编译程序与程序设计环境
- 编译程序的生成

编译程序生成

- 以汇编语言和机器语言为工具
  - 优点： 可以针对具体的机器，充分发挥计算机的系统功能；生成的程序效率高
  - 缺点： 程序难读、难写、易出错、难维护、生产的效率低

编译程序生成

- 高级语言书写

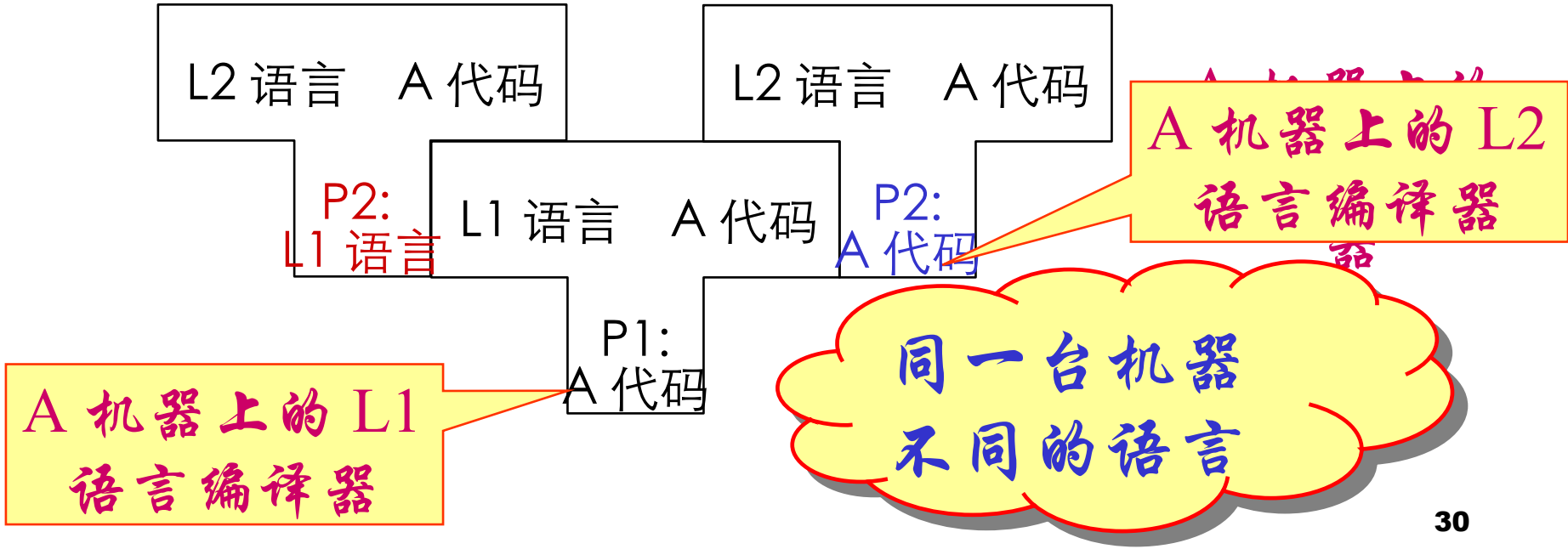


S 源程序　T 目标程序　I 实现语言

  - 程序易读、易理解、容易维护、生产的效率高

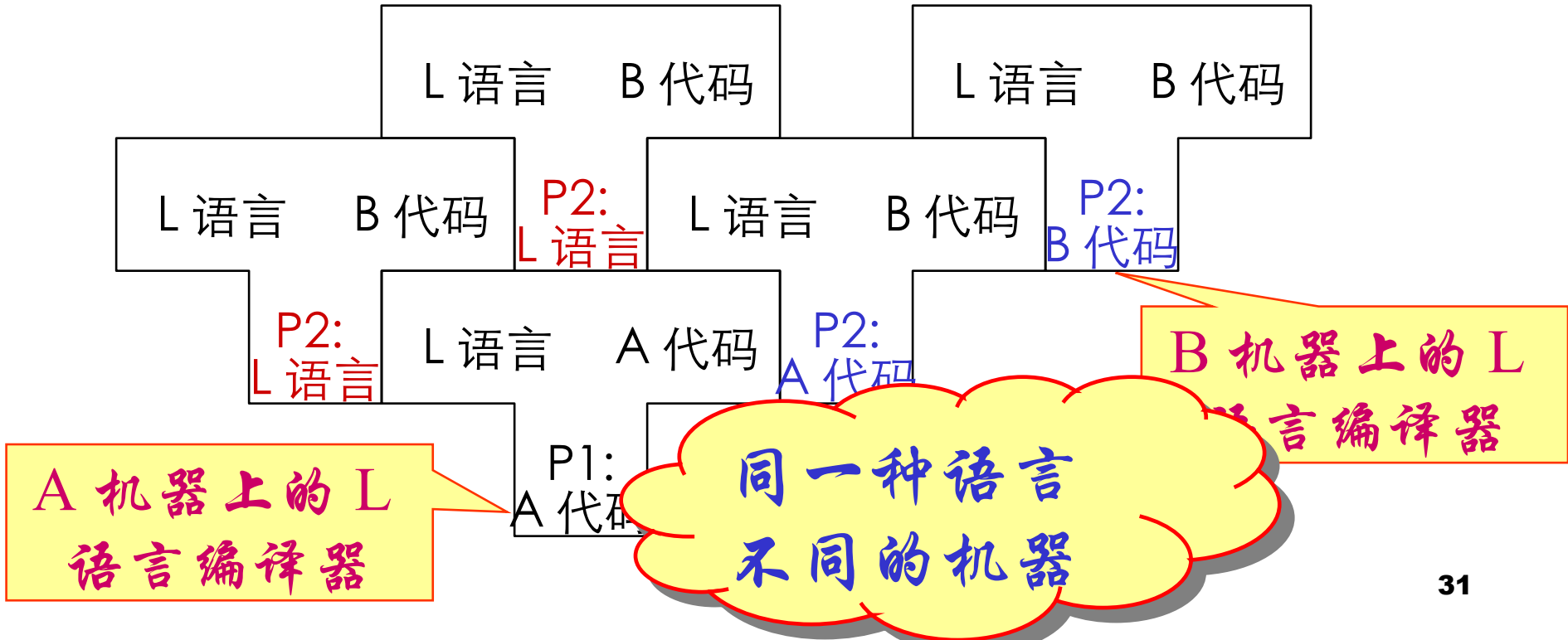
编译程序生成

- 高级语言书写
  - 利用已有的某种语言的编译程序实现另一语言的编译程序



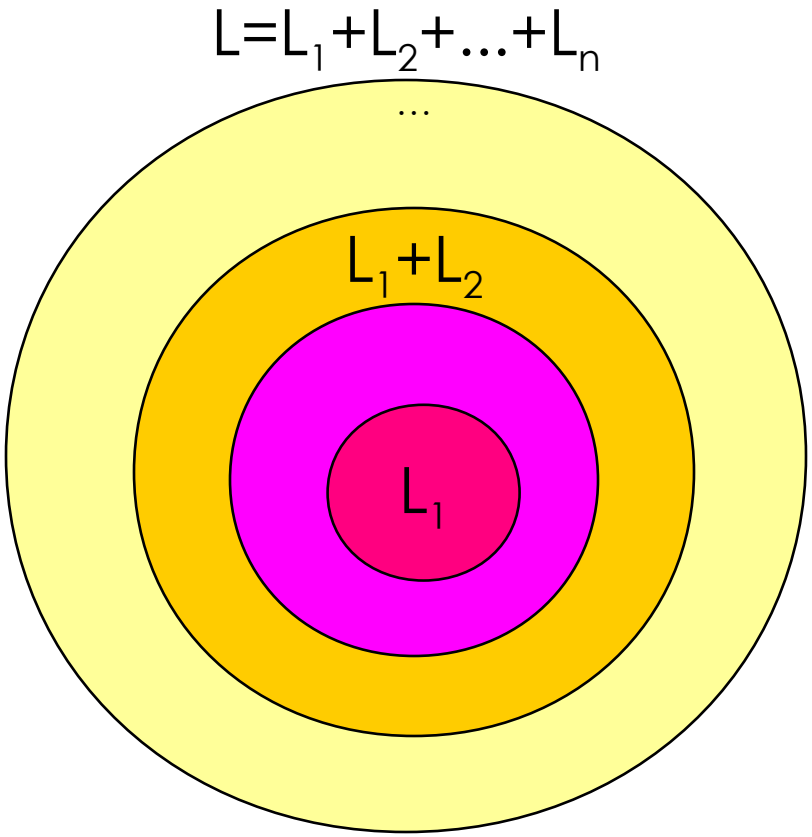
编译程序生成

- 移植方法
  - 把一种机器上的编译程序移植到另一种机器上



编译程序生成

- 自展技术



编译程序生成

- 编译程序自动产生
  - 编译程序 - 编译程序，编译程序书写系统

LEX 词法分析程序产生器
YACC 语法分析程序产生器

第一章 引论

- 课程概述
  - 内容、意义、安排
- 什么是编译程序
  - 翻译、编译、解释
- 编译基本过程
- 编译程序的结构
  - 阶段、遍、前端 / 后端
- 编译程序生成的几种方法