



编译概论





- 程序设计语言
- 编译引论
- GCC和LLVM介绍



- 语言发展历史
- 语言分类
- Lab. 1
- 语言综述



- 程序设计语言的演化
 - 二十世纪四十年代(1940), 机器语言
 - 二十世纪五十年代(1950) 早期, 汇编语言
 - 二十世纪五十年代(1950)后期, 高级程序设计语言
 - 科学计算FORTRAN
 - 商业数据处理Cobol
 - 符号计算Lisp
 - 二十世纪六十年代以后, 更多的高级程序设计语言



■ 分类标准：程序设计语言的演化分代

■ 第一代

机器语言

■ 第二代

汇编语言

■ 第三代

高级程序设计语言，比如：Fortran、Cobol、Lisp、C、C++、C#、Java

■ 第四代

特定应用设计的语言，比如：NOMAD(生成报告)、SQL(数据库查询)、Postscript(文本排版)

■ 第五代

基于逻辑和约束的语言，比如：Prolog、OPS5



- **分类标准: 描述方式**
 - **命令(强制)式:** 编程告诉计算机如何完成工作
 - 面向过程语言
 - 面向对象语言
 - **声明式:** 编程告诉计算机要完成哪些工作
 - 函数式语言, 比如: ML, Haskell
 - 逻辑编程语言, 比如: Prolog

语言分类



■ 分类标准:数据类型1

■ 静态类型

- 运行前(编译时)类型检查

■ 动态类型

- 运行时检查类型

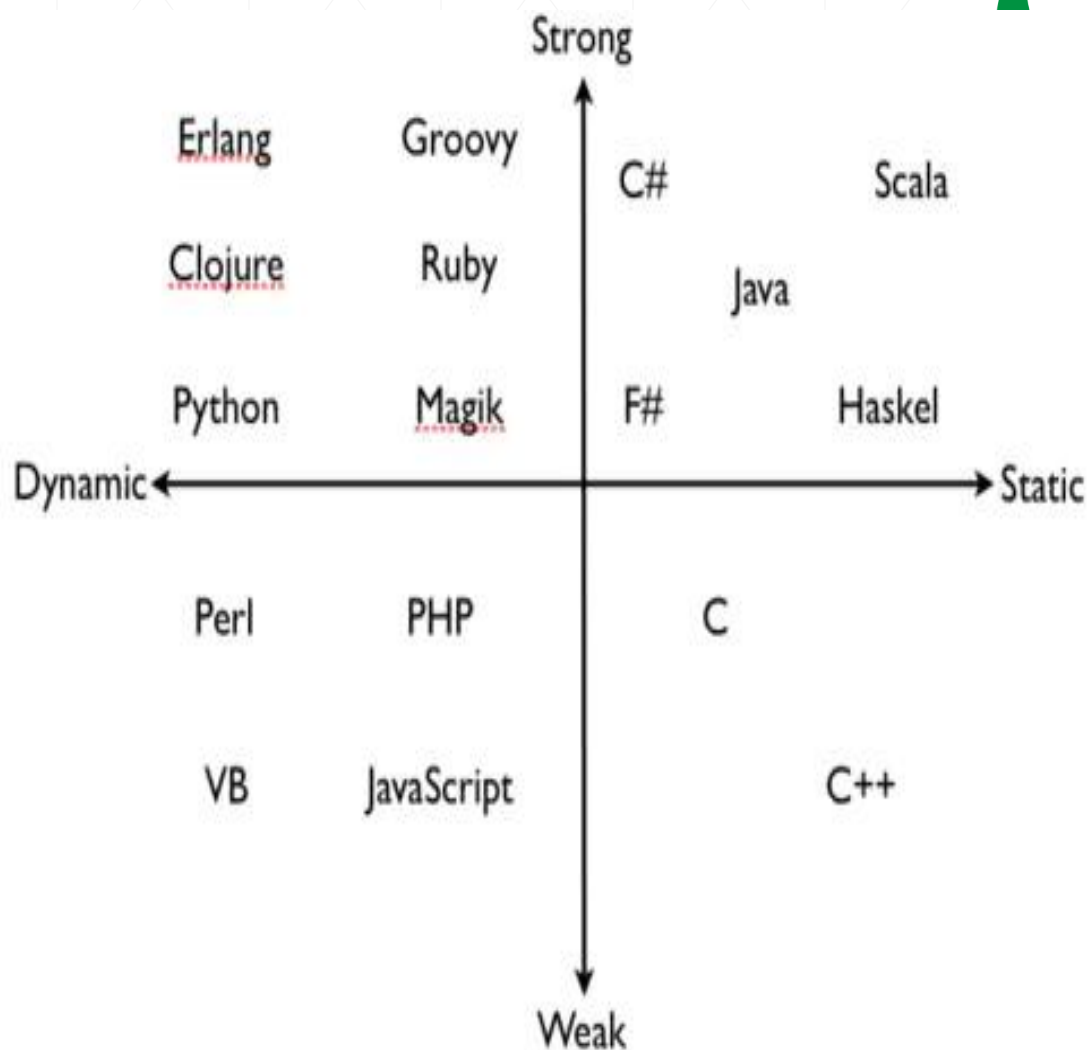
■ 分类标准:数据类型2

■ 强类型

- 不允许隐式数据类型转换

■ 弱类型

- 允许隐式数据类型转换





- 分类标准：实现方式
 - 编译型
 - 解释型
- 分类标准：按照应用领域
 - 科学计算(FORTRAN)
 - 商业应用(SQL)
 - 系统编程(C/C++)
- 分类标准：运算单元
 - 冯.诺依曼语言
 - 面向对象语言
 - 脚本语言



- 选定一个特定的功能，分别使用C/C++、Java、Python、Haskell和一种汇编语言(X86、MIPS、ARM或者RISC-V等)实现该功能，对采用这几种语言实现的编程效率，程序的规模，程序的运行效率进行对比分析。
- 具体要求见乐学平台上相关材料



- 不断有新的语言涌现
 - 为什么会有这么多种语言?
 - 什么是好的语言?



◆语言演化

不断发现有更好的方式做事情

◆特殊目的

为了特定的应用各领域和问题设计新的语言
大数据、人工智能

◆个人喜好

◆表达方式强大：在某一方面抽象程度高

◆新手容易学习

◆容易实现

◆小机器上也可以实现、容易移植

◆开源且社区活跃

◆强大的编译器支持

◆ ...



- 程序设计语言
- 编译引论
- GCC和LLVM介绍



第一章 编译引论

1.1 程序设计语言与编译程序



1.2 编译程序的表示与分类

1.3 编译程序的结构与编译过程

~~1.4 语言开发环境中的伙伴程序~~

1.5 编译程序结构的实例模型

1.6 编译程序的构造与实现

Compiler ?

Compiler ?

■ 定义1.1 :

将某一种程序设计语言写的程序翻译成等价的另一种语言的程序的程序，称之为编译程序或编译器(compiler)。

- ✓ 借助编译程序可以执行高级语言编写的程序；
- ✓ 编译程序是语言处理系统；
- ✓ 人、机交互的工具；
- ✓

源程序： (source program)

被翻译(编译) 的程序称为源程序。

源语言： (source language)

编写源程序的语言是源语言。

源语言一般是汇编语言或高级程序设计语言。

Compiler ?



源程序

■ 定义1.1:

将某一种程序设计语言写的程序翻译成等价的另一种语言的程序的程序，称之为编译程序或编译器(compiler)。

- ✓ 编译程序是语言处理系统；
- ✓ 编译程序是将高级语言翻译成机器语言的程序；
- ✓ 人、机交互的工具；
- ✓

目标程序: (target program)

源程序经过翻译（编译）后生成的程序称之为目标程序。

目标语言: (target language)

目标程序的描述语言称为目标语言。

Compiler ?

■ 定义1.1:

源程序



将某一种程序设计语言写的程序翻译成等价的另一种语言的程序的程序，称之为编译程序或编译器(compiler)

目标程序

- ✓ 编译程序是语言处理系统；
- ✓ 编译程序是将高级语言翻译成机器语言的程序；
- ✓ 人、机交互的工具；
- ✓

Compiler ?

■ 定义1.1:

将源程序翻译成等价的程序的目标程序，称之为编译程序或编译器(compiler)。

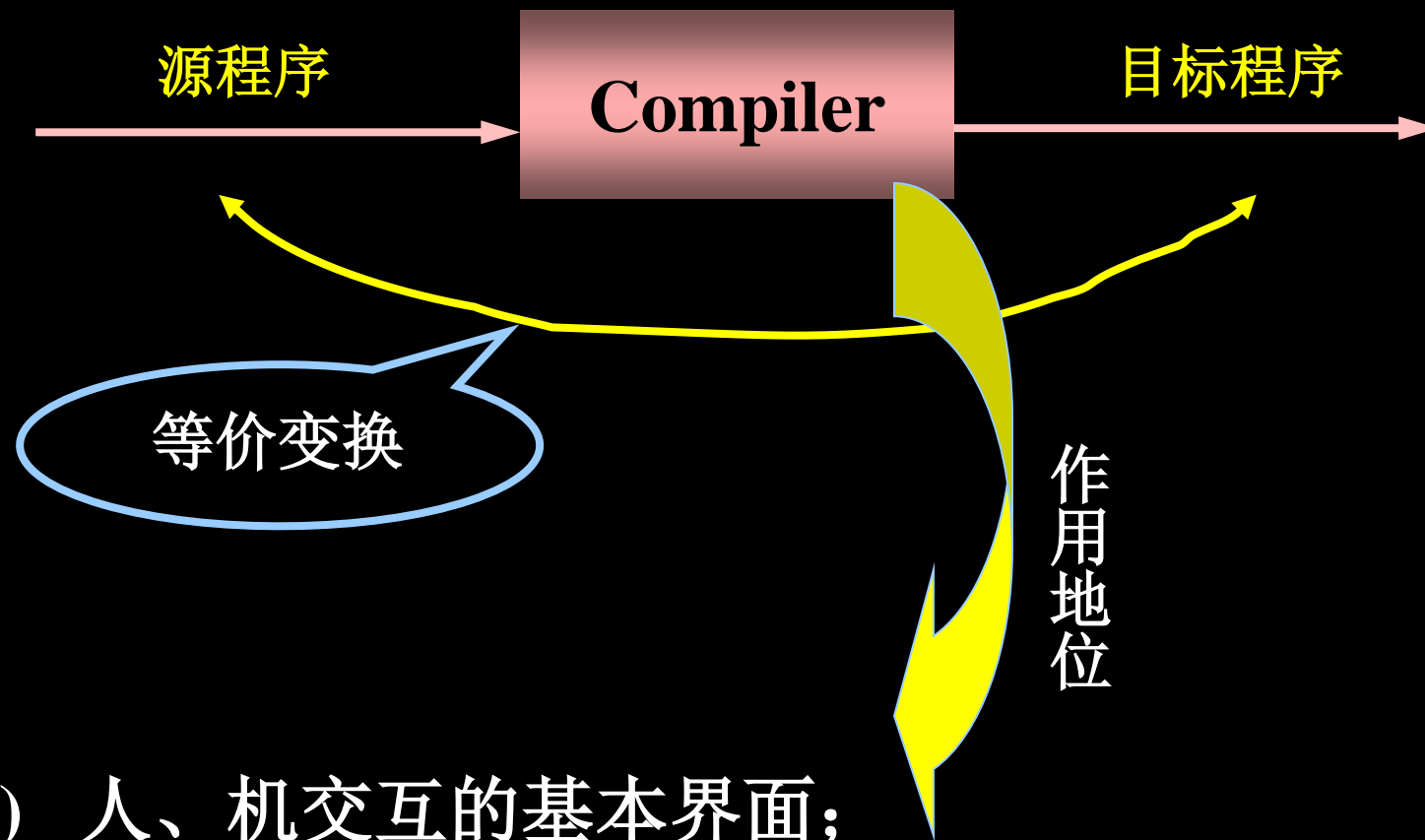
- ✓ 编译程序是语言处理系统；
- ✓ 编译程序是将高级语言翻译成机器语言的程序；
- ✓ 人、机交互的工具；
- ✓

宿主语言：

编译程序的实现语言。

宿主机（目标机）：

编译程序的运行环境（运行编译程序的计算机）。

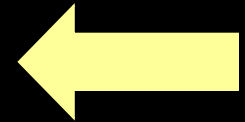


- (1) 人、机交互的基本界面;
- (2) 软件学科的重要分支;
- (3) 用户直接关心的工具。

第一章 编译引论

1.1 程序设计语言与编译程序

1.2 编译程序的表示与分类



1.3 编译程序的结构与编译过程

~~1.4 语言开发环境中的伙伴程序~~

1.5 编译程序结构的实例模型

1.6 编译程序的构造与实现

一. 编译程序的分类A



汇编程序： 接受汇编源程序，将其翻译成等价的机器语言表示的目标程序。

编译程序： 接受某语言的源程序，将其直接翻译成等价的目标代码。

一. 编译程序的分类B



解释程序： 接受某语言的源程序将其直接翻译成目标代码且执行。

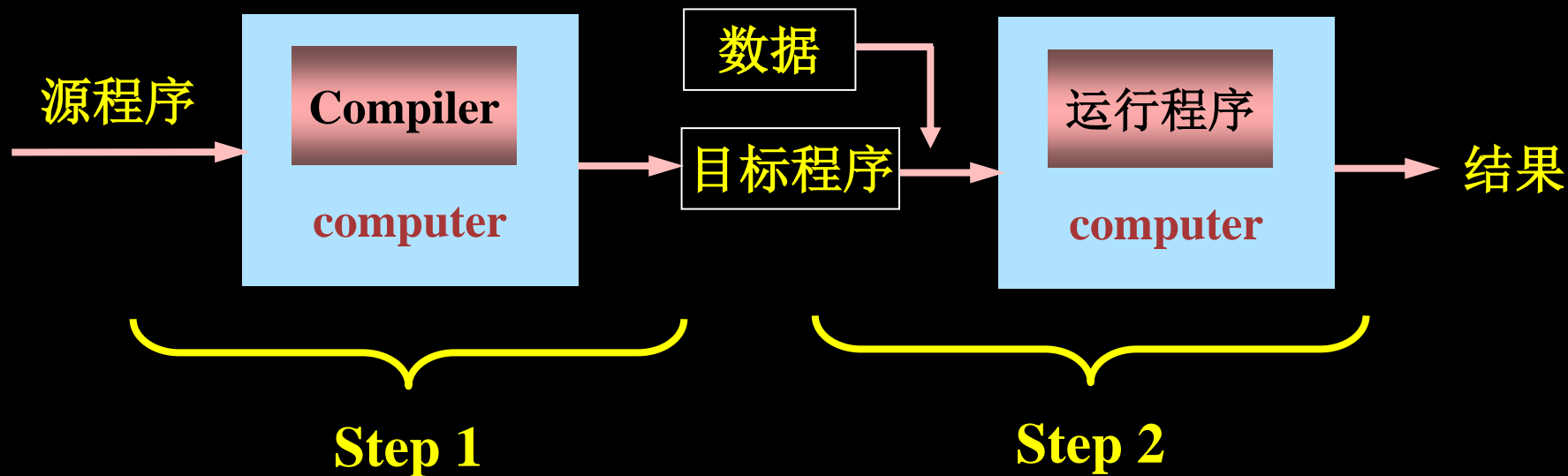
编译程序： 接受某语言的源程序，将其直接翻译成等价的目标代码。

编译执行和解释执行的区别：

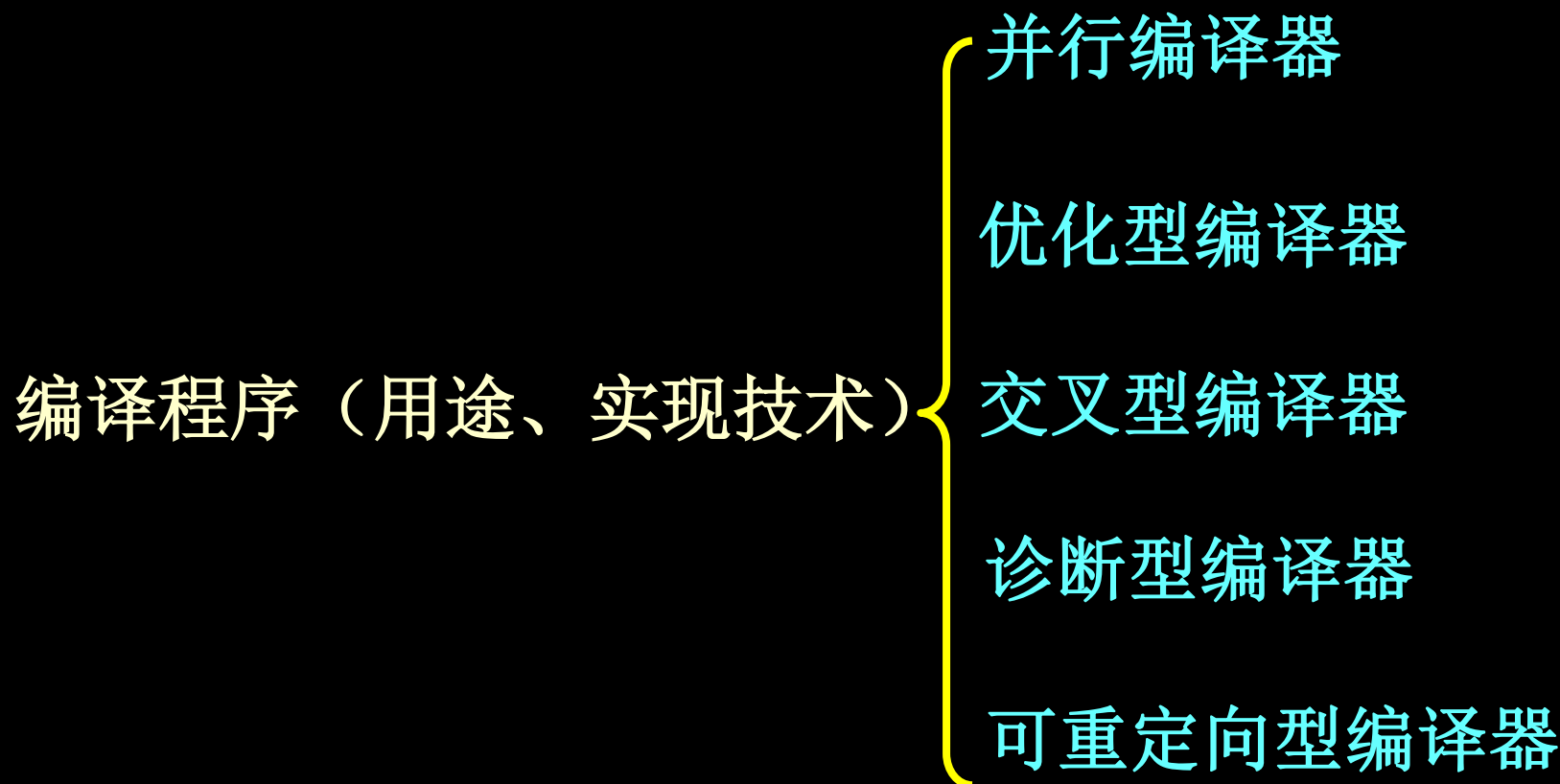
编译执行是由编译程序生成一个与源程序等价的
目标程序，它可以完全取代源程序，该目标程序
可以运行任意次。

解释执行没有输出等价的目标程序，每次执行
都需要重新处理源程序。

编译执行类似于自然语言的笔译；解释执行类似
自然语言的口译。

解释执行:编译执行:

一. 编译程序的分类C



二. 编译程序的表示 (体现编译程序的三个要素)

1) 函数表示 :

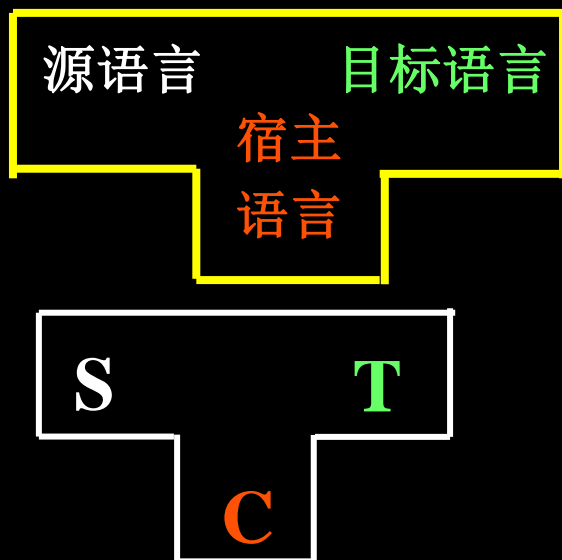
$$T = C(S)$$

T: 目标语言

C: 宿主语言

S: 源语言

2) T型图表示

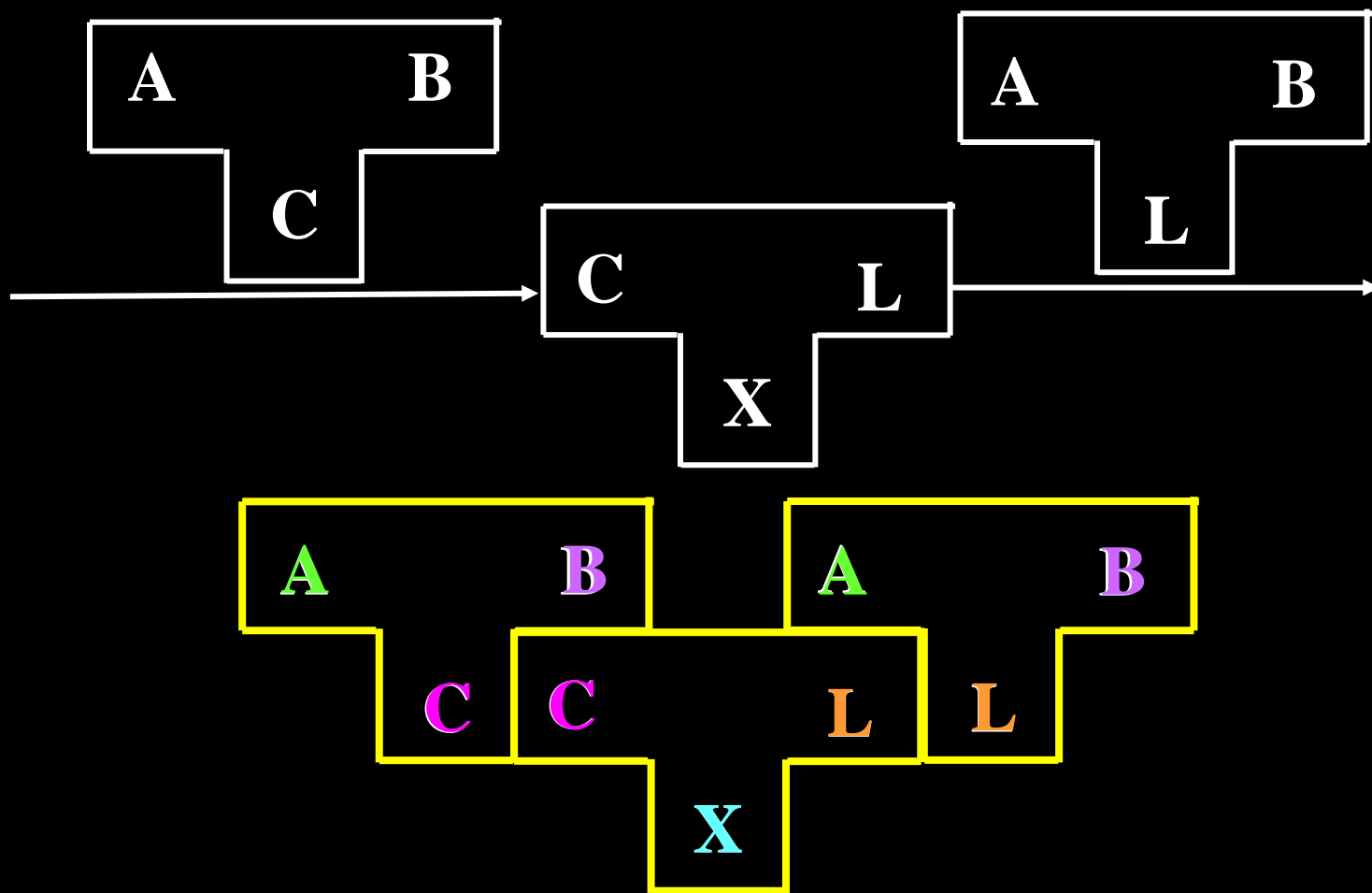


3) 符号表示

C^{源 目}
宿

C^{S T}
C

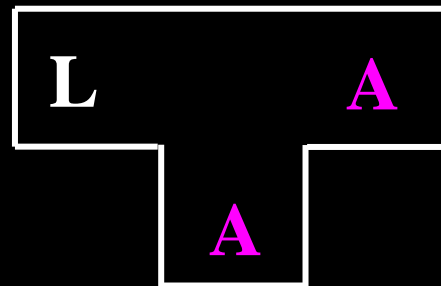
■ T型图的组合



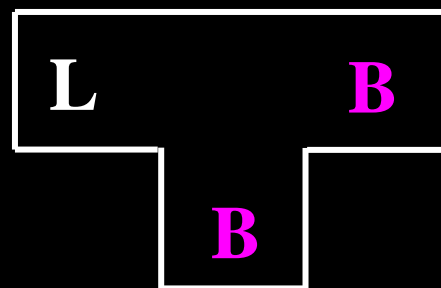
例1.1： 用**T型图**描述将**A**机器上已经运行的**L**语言的编译程序移植到**B**机器上的过程。

分析：

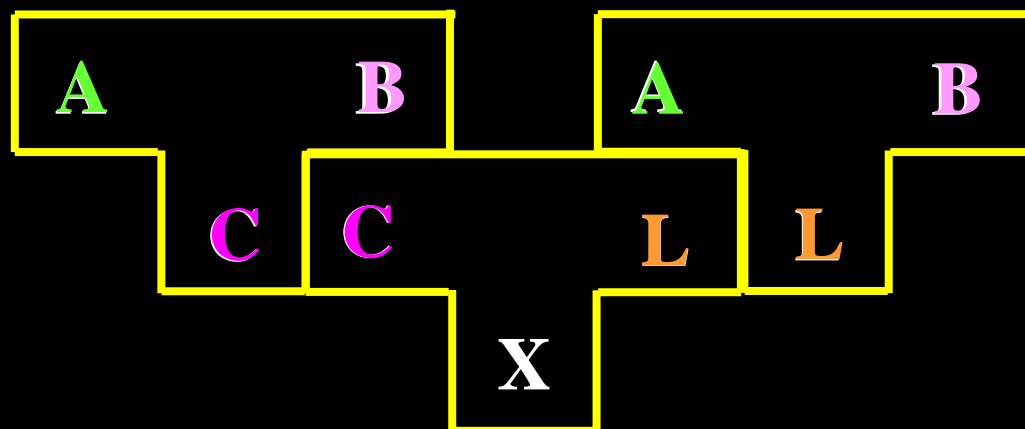
已知的编译器：



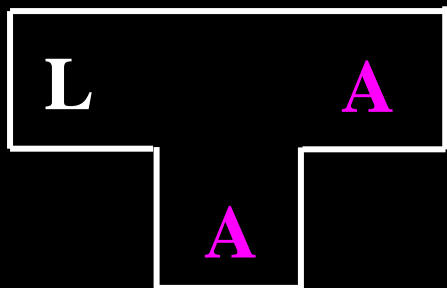
要得到的编译器：



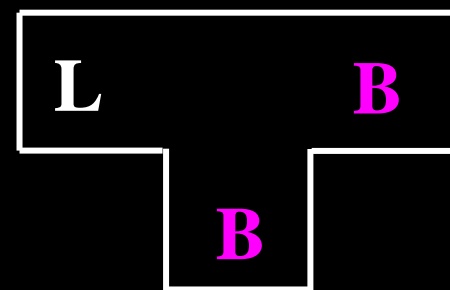
■ T型图的组合



已知的编译器:

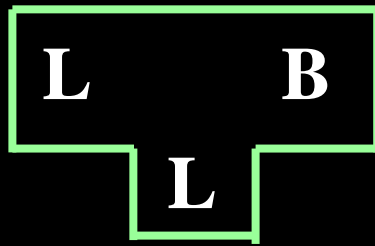


得到的编译器:

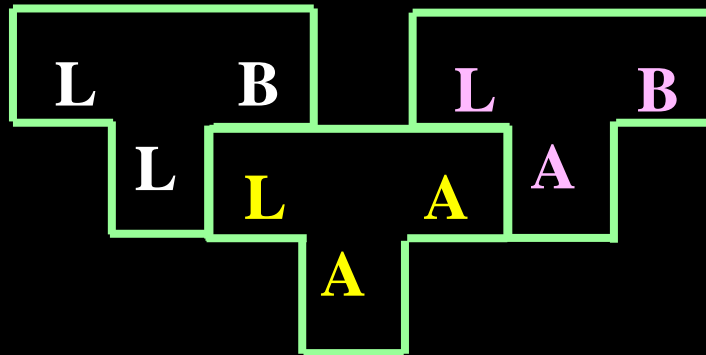


例1.1： 用T型图描述将A机器上已经存在的L语言的编译程序移植到B机器上的过程。

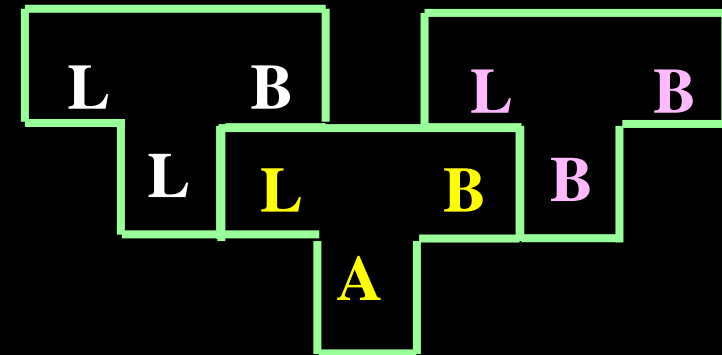
Step1:



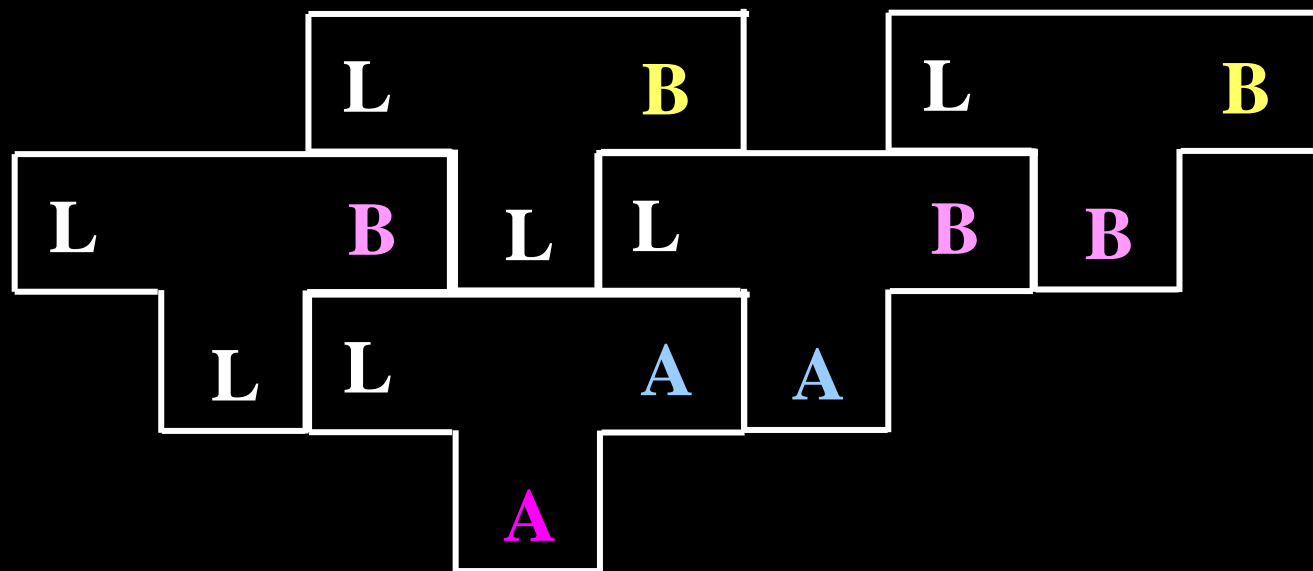
Step2:



Step3:

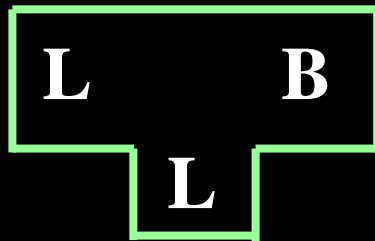


例1.1： 用**T型图**描述将**A**机器上已经存在的**L**语言的编译程序移植到**B**机器上的过程。

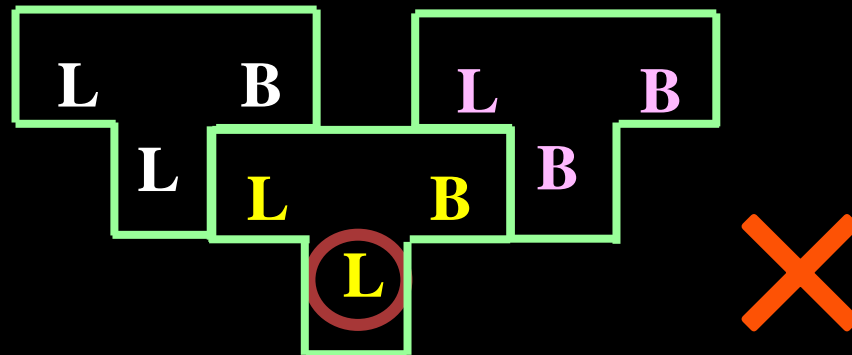


例1.1： 用T型图描述将A机器上已经存在的L语言的编译程序移植到B机器上的过程。

Step1:



Step2:

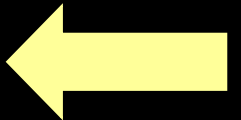


第一章 编译引论

1.1 程序设计语言与编译程序

1.2 编译程序的表示与分类

1.3 编译程序的结构与编译过程



~~1.4 语言开发环境中的伙伴程序~~

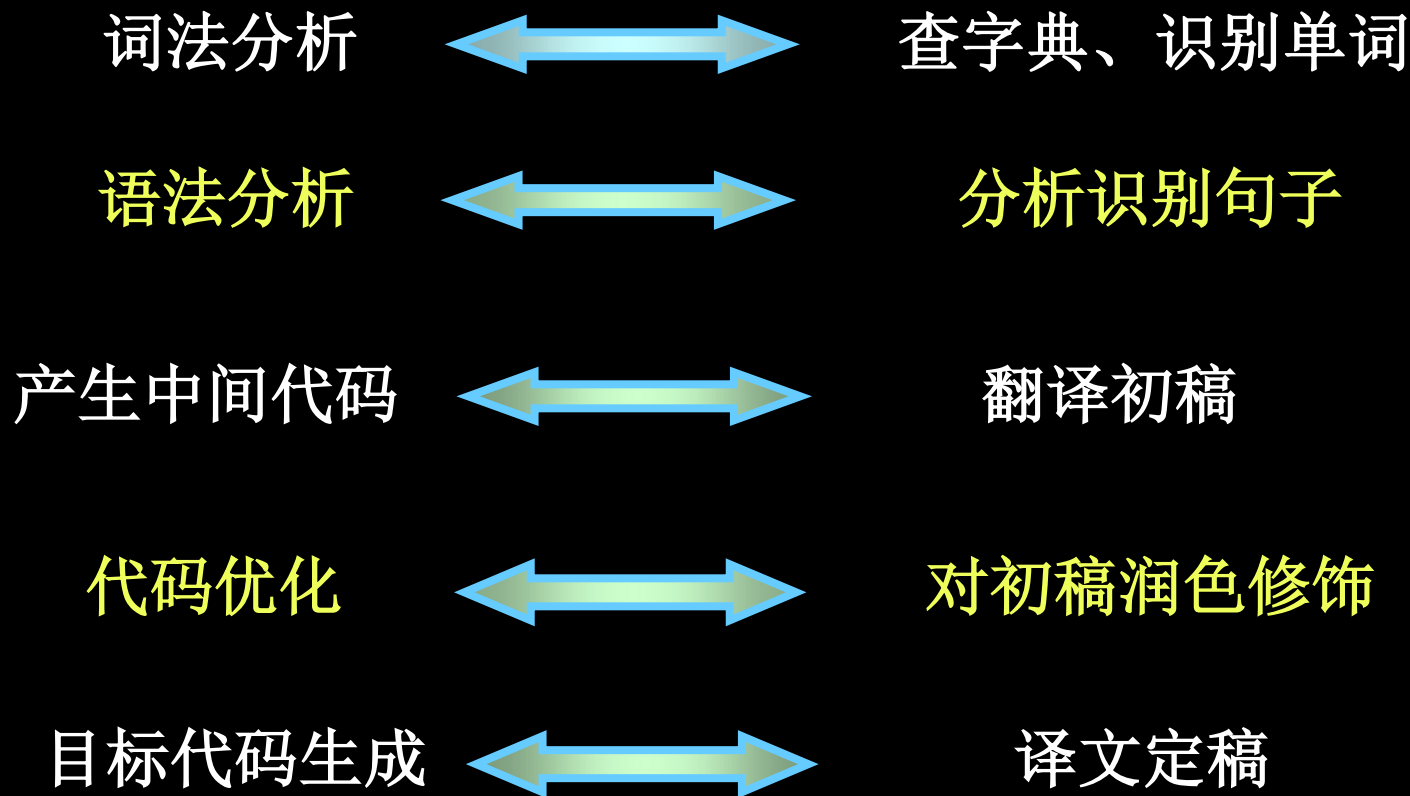
1.5 编译程序结构的实例模型

1.6 编译程序的构造与实现

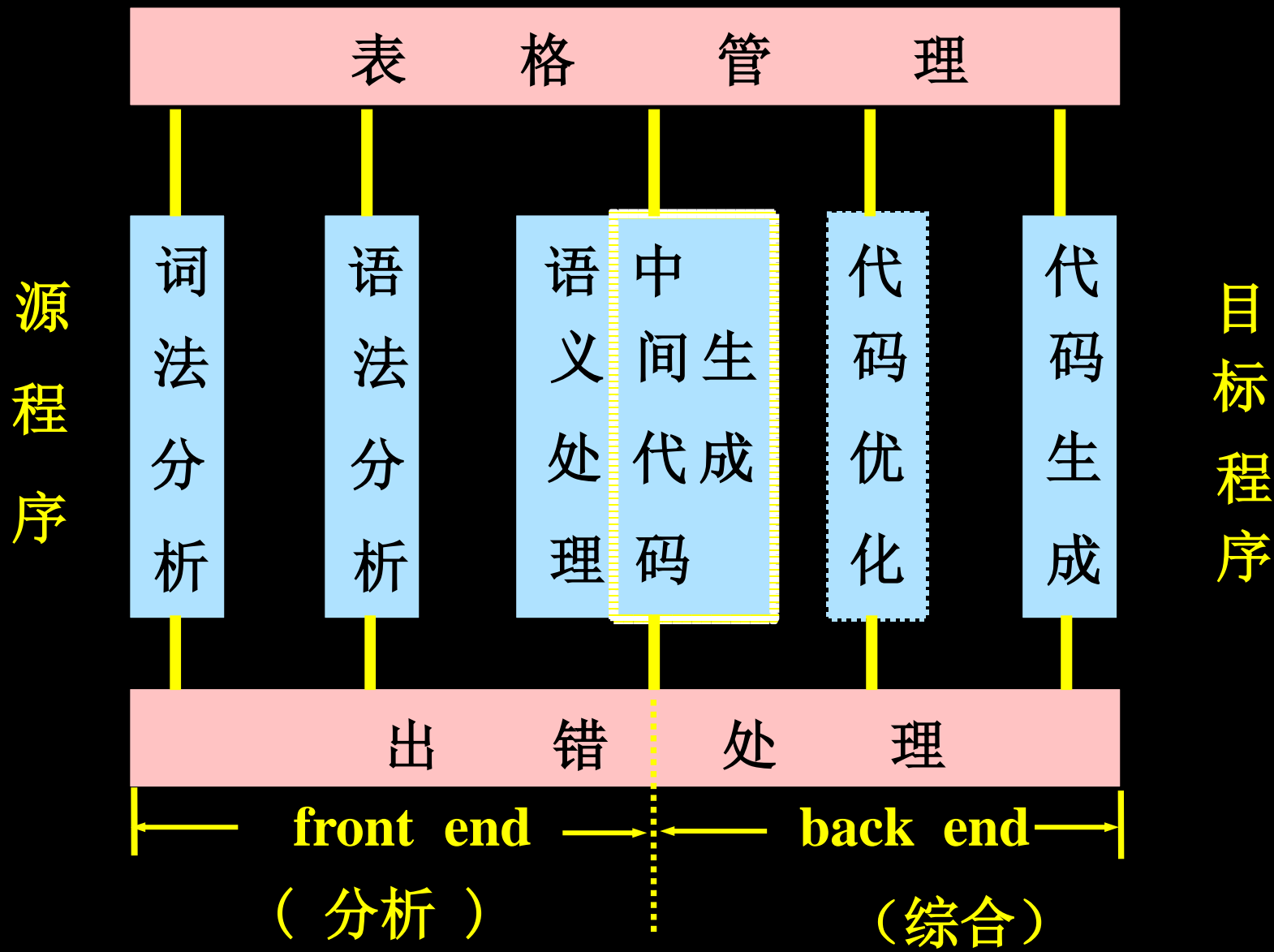
☺ 编译阶段与自然语言的翻译对比

编译程序

自然语言翻译



■ 编译程序的逻辑结构 (经典划分)



1. 词法分析 (lexical analysis)

依据词法规则,

识别出读入的源程序中有独立意义的源语言单词,
用某种特定的数据结构(属性字或记号)进行表示。



对输入的符号串形式的源程序进行最初的加工处理。
一种线性分析。

属性字:类型标识和内码两个数据项。

例1.2: 如下C代码行 :

`a[index] = 12 * 3 ;`

词法分析:

(1) a
(2) [
(3) index
(4)]
(5) =
(6) 12
(7) *
(8) 3
(9) ;

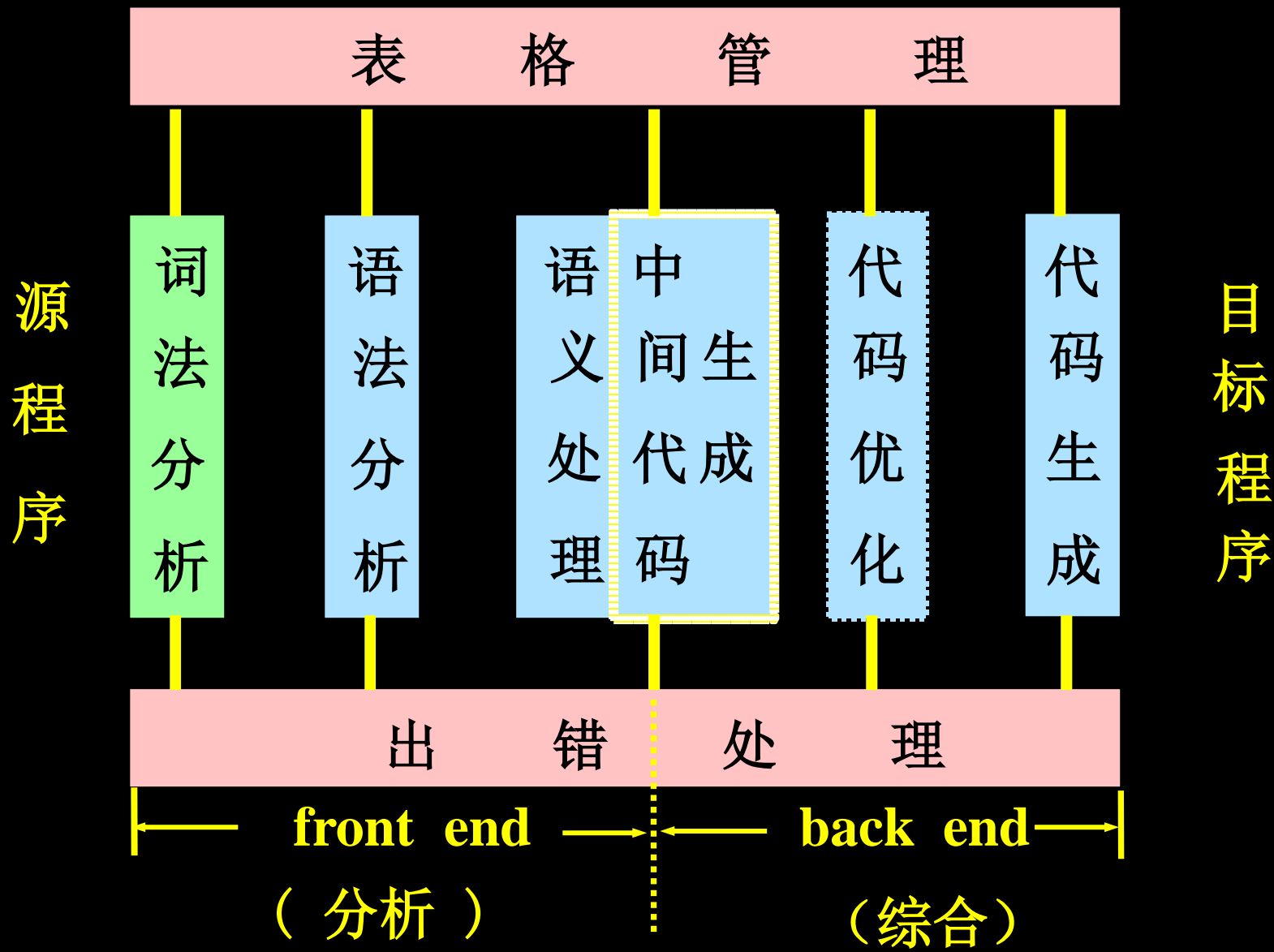
<标识符,1>
<左方括号,>
<标识符,2>
<右方括号,>
<赋值,>
<整常数,1>
<乘号,>
<整常数,2>
<分号,>

源程序

属性字流

识别出9个单词。

■ 编译程序的逻辑结构 (经典划分)



2. 语法分析 (Syntax analysis)

依据语法规则

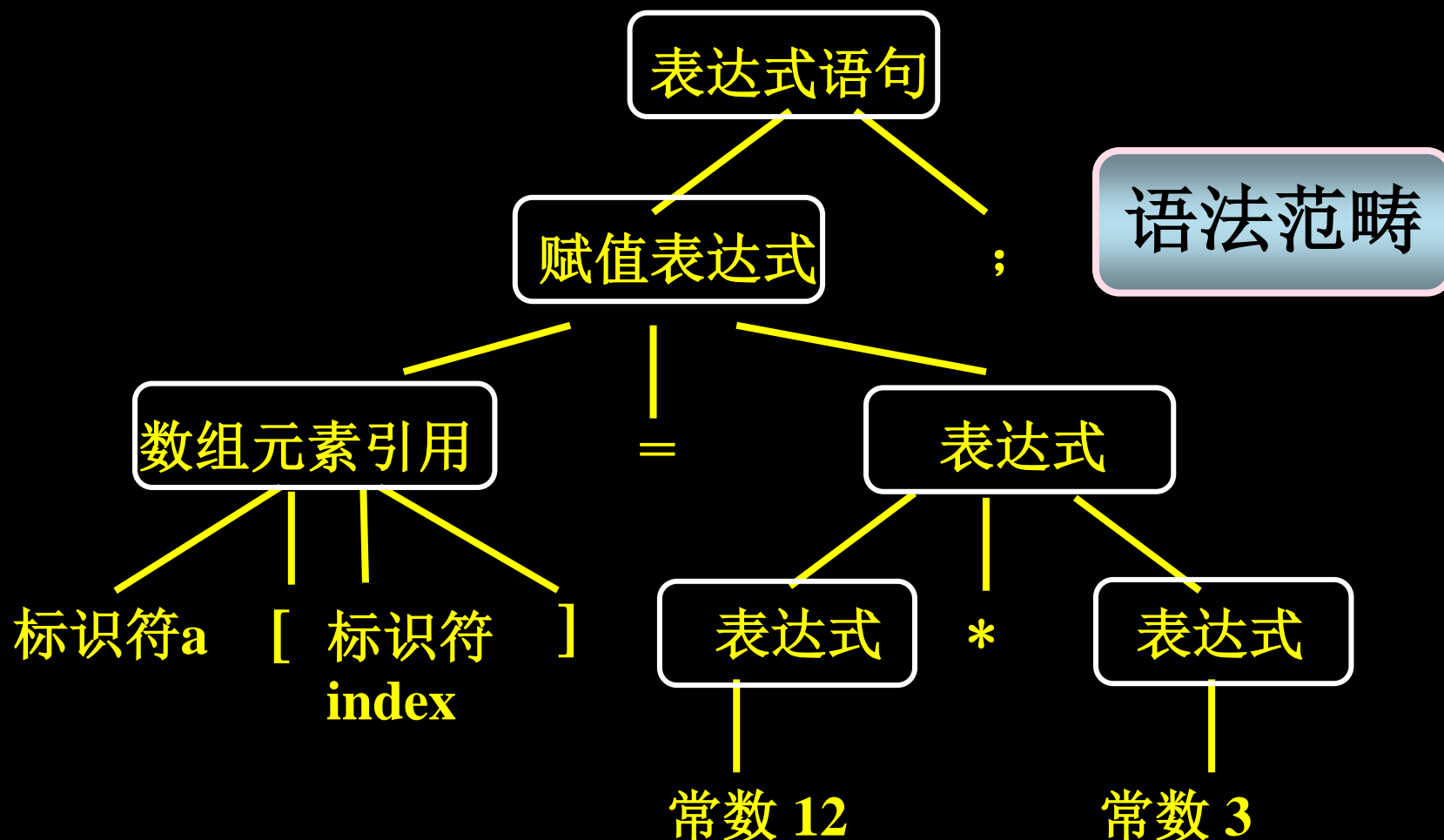
对属性字流进行语法检查,

识别出对应的语法范畴。

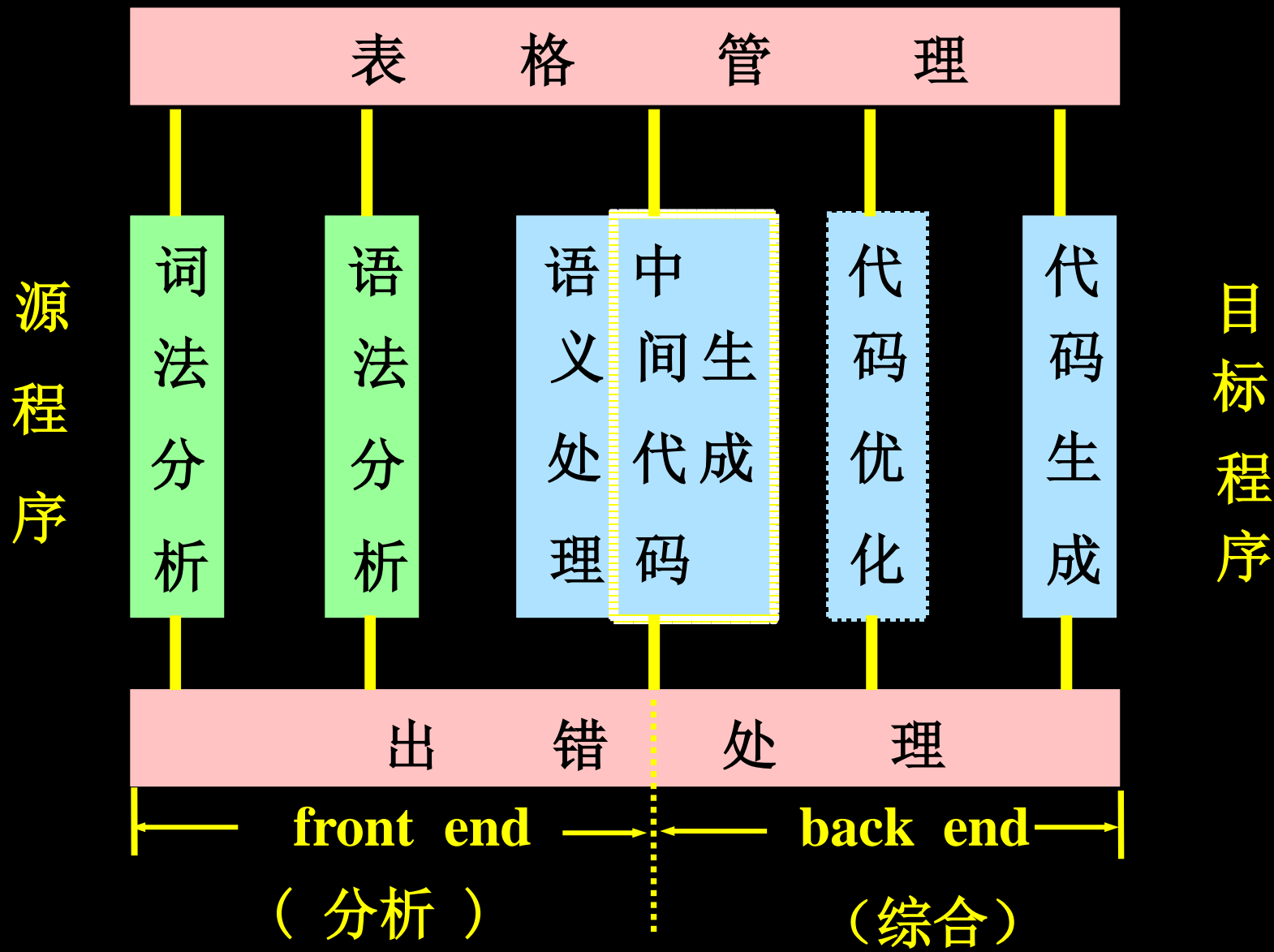
通常结果表示为语法分析树或抽象语法树。



线性结构⇒层次结构

C语句 $a[index] = 12 * 3$; 的语法分析树

■ 编译程序的逻辑结构 (经典划分)



3. 语义分析 (semantic analysis)与中间代码生成

依据语义规则

对识别出的语法范畴进行语义检查和处理,

翻译成**等价**的某种中间代码或目标代码。

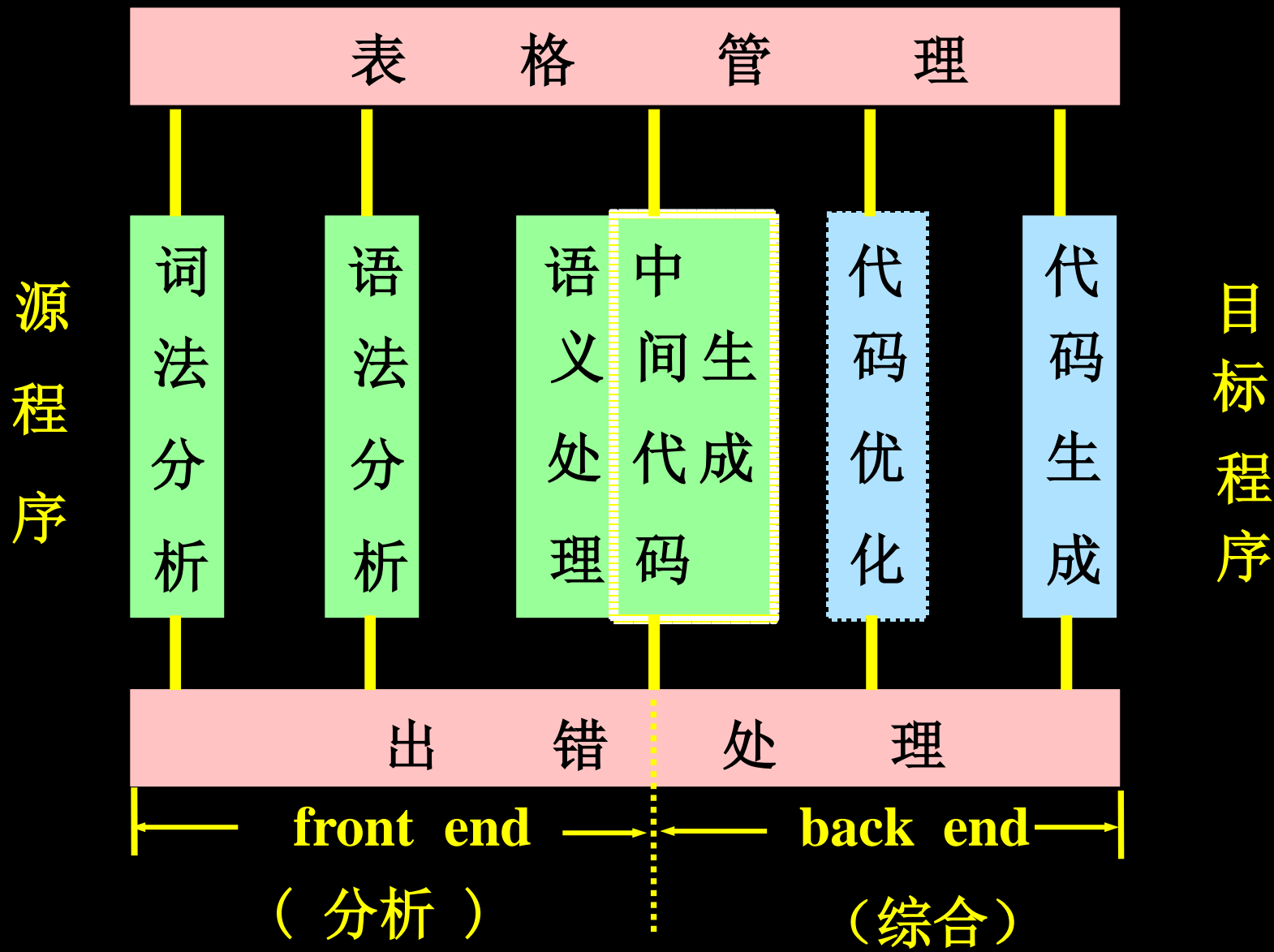
整个编译程序完成的最实质性的翻译任务。

赋值语句 $a[index] = 12 * 3$;

产生的四元式形式的中间代码:

*	12	3	T_1
[]=	T_1	-	$a[index]$

■ 编译程序的逻辑结构 (经典划分)



4. 代码优化 (optimization)

不改变源程序语义的基础上

对当前程序加工变换,

以期获得高效的~~目标~~代码。

中间代码或目标代码

运行时间的缩短和存
贮空间的节省。

赋值语句 $a[index] = 12 * 3$ 的四元式代码

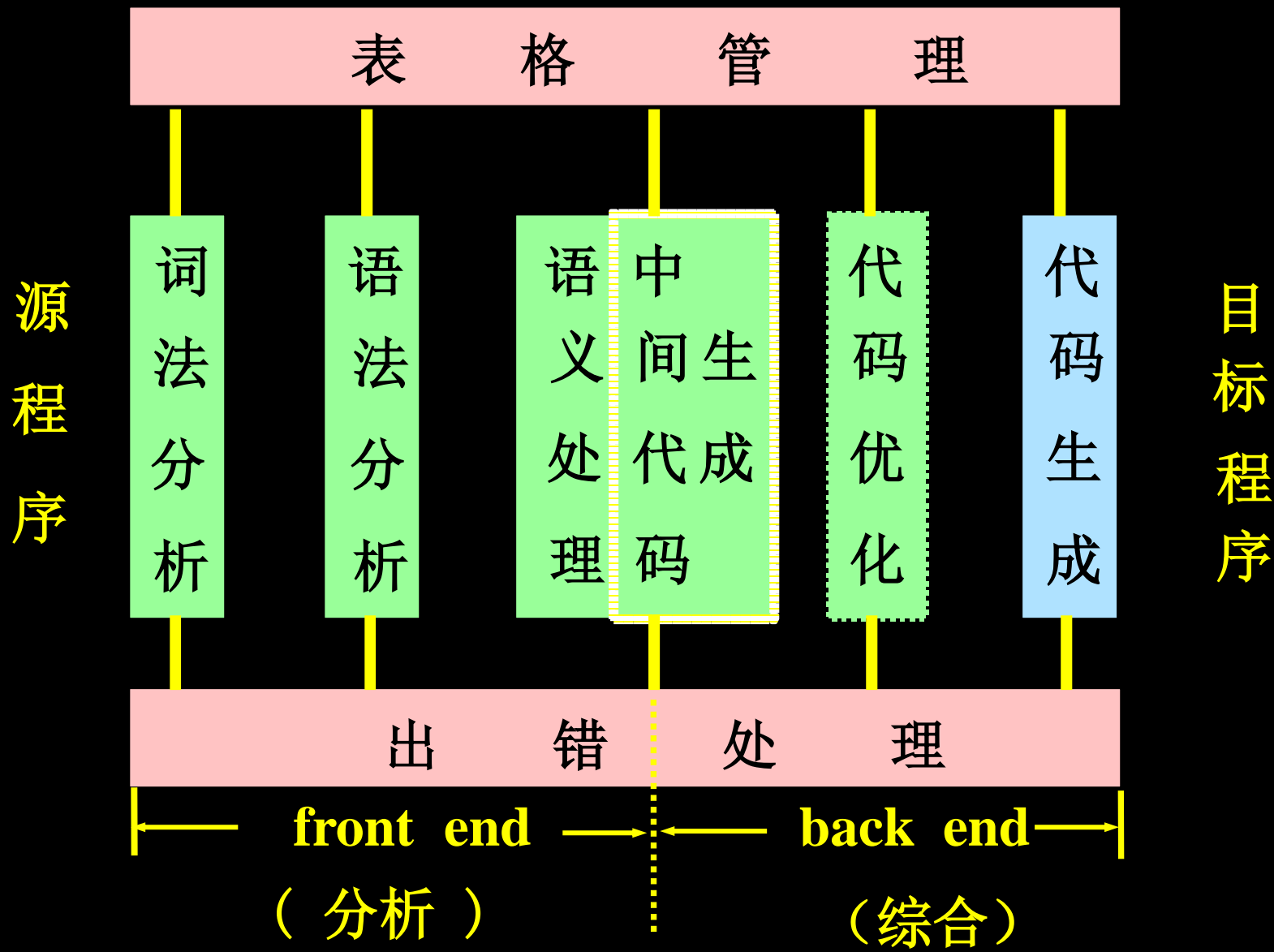
*	12	3	T_1
[]=	T_1	-	$a[index]$

[]=	36	-	$a[index]$
------	----	---	------------



优化后

■ 编译程序的逻辑结构 (经典划分)



5. 目标代码生成 (code generator)

根据中间代码及编译过程中产生的各种表格的有关信息,

生成所期望的目标代码程序。

一般为特定机器的机器语言代码或汇编语言代码。

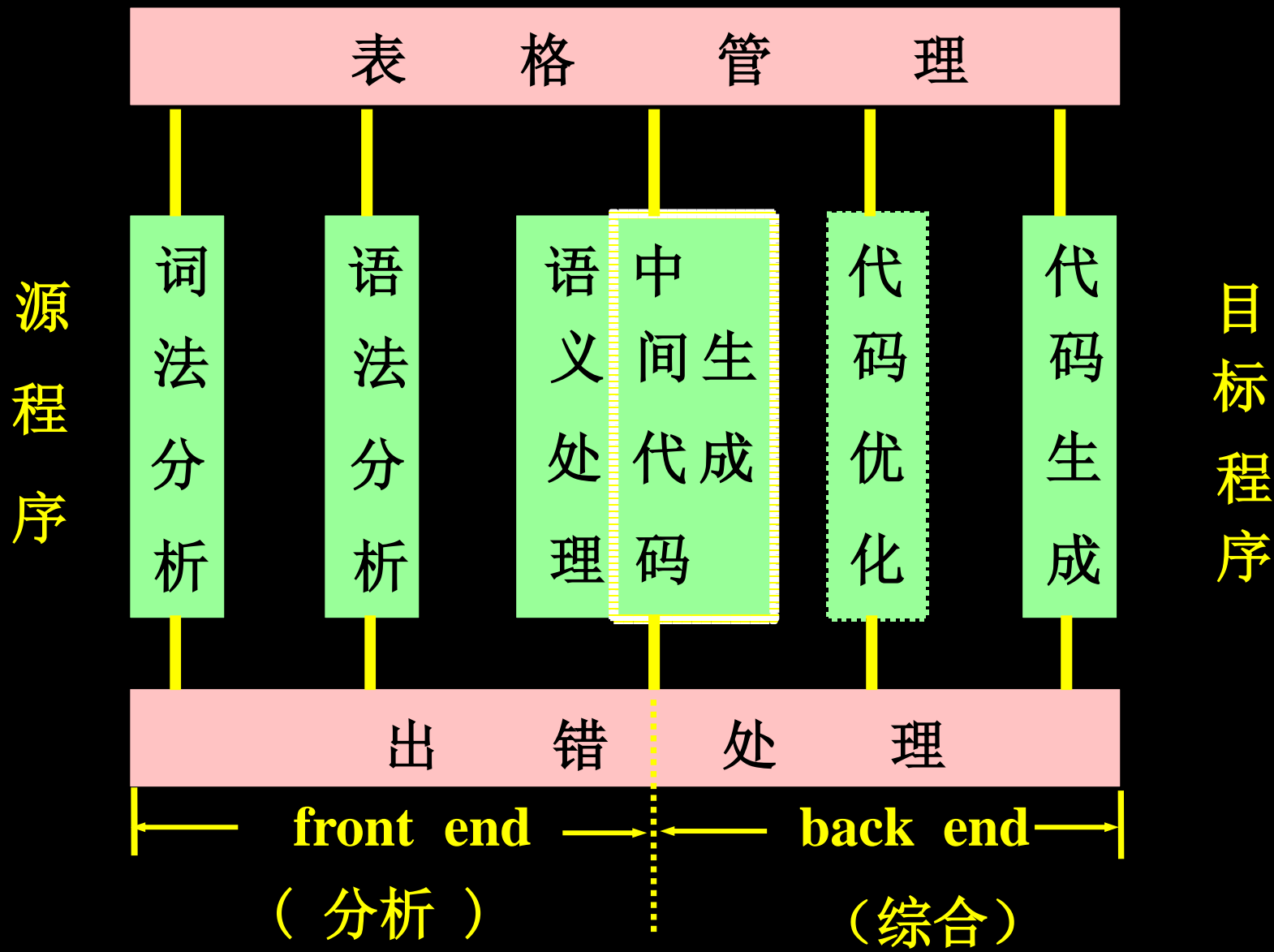
要充分考虑计算机硬件和软件所提供的资源,

生成较高质量的目标程序。

对语句 “ $a[index] = 12 * 3$; ” 生成目标代码。
要考虑怎样存储整型数来为数组元素的引用生成
目标代码，以假设的汇编语言描述如下：

MOV	R₀, index	;; 索引值赋给寄存器R₀
MUL	R₀, 2	;; 存储按字节编址，整型数 ;; 占2个字节
MOV	R₁, &a	;; &a表示数组a的基地址
ADD	R₁, R₀	;; 计算a[index]的地址
MOV	*R₁, 36	;; a[index] = 36

■ 编译程序的逻辑结构 (经典划分)



■ 表格管理

公共辅助部分。

对各种量进行管理，登记在相应的表格。

处理时通过查表得到所需的信息。

■ 出错处理

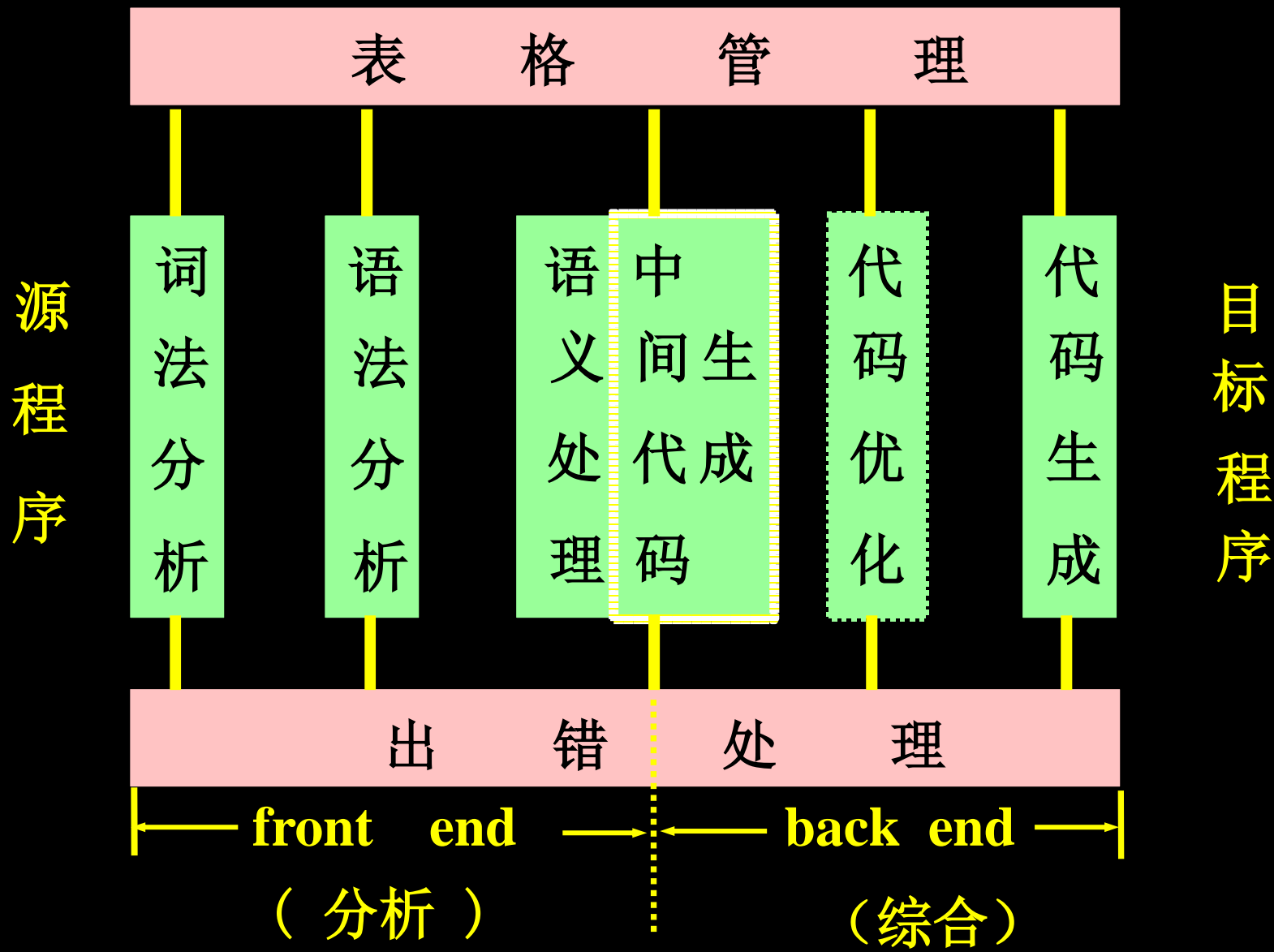
公共辅助部分。

对源程序中的各种错误检查、定位、定性、报告，

尽可能将错误限制在尽可能小的范围内，

保障编译继续。

■ 编译程序的逻辑结构 (经典划分)



■ 定义1.2 :遍(Pass趟)

对源程序或源程序的中间形式从头到尾扫描一遍，并做有关的分析加工，生成新的与源程序等价的中间形式或生成目标程序。

■ 遍数多的优点：

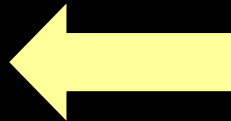
- 1) 编译程序逻辑结构清晰；
- 2) 减少对主存容量的要求；
- 3) 优化准备充分；

.....

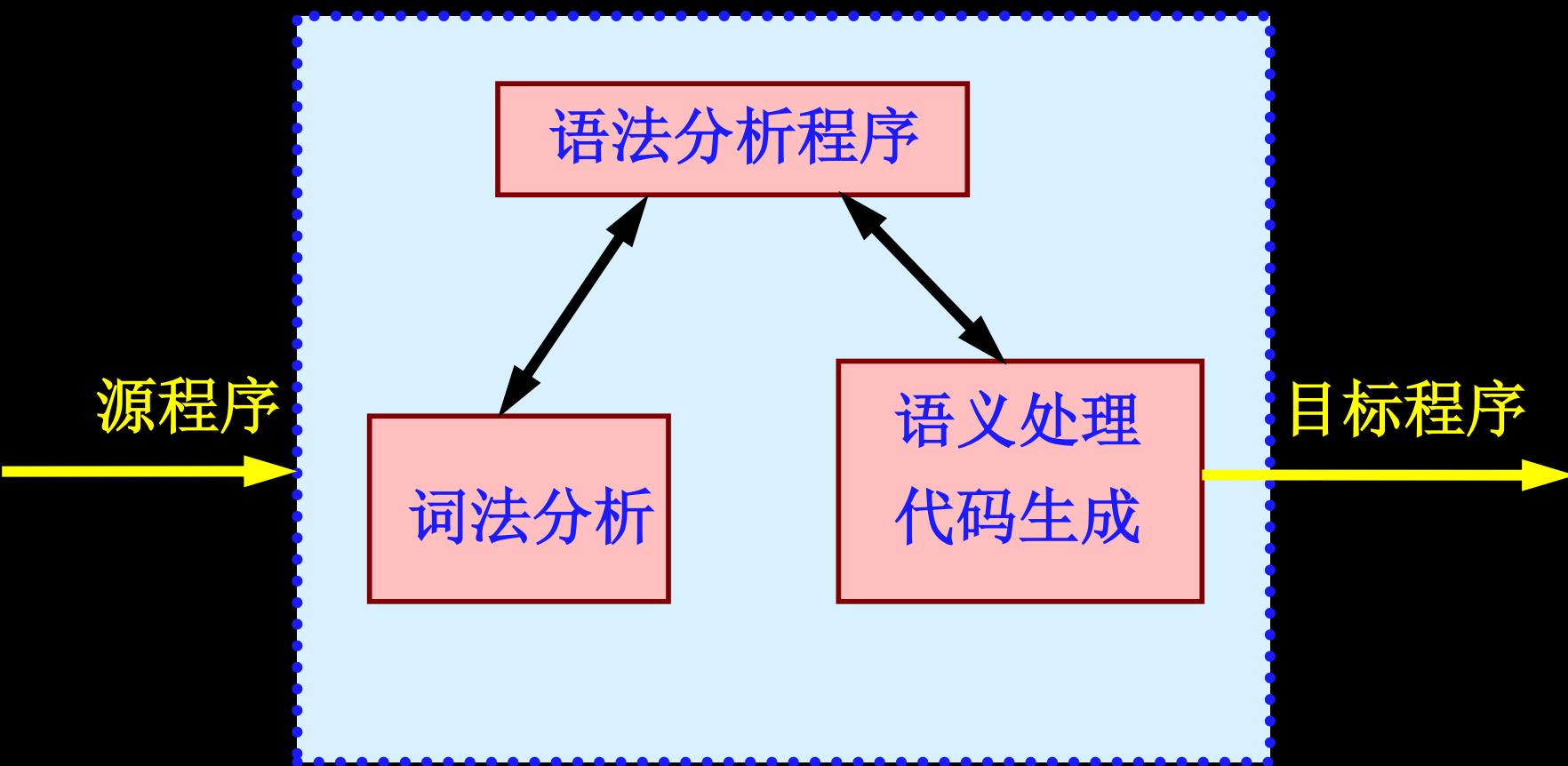
■ 遍设置的考虑因素

- 1) 宿主机存储容量;
- 2) 编译程序功能的强弱;
- 3) 源语言的繁简及约束;
- 4) 优化因素;
- 5) 设计、实现的环境、工具及人员因素等。

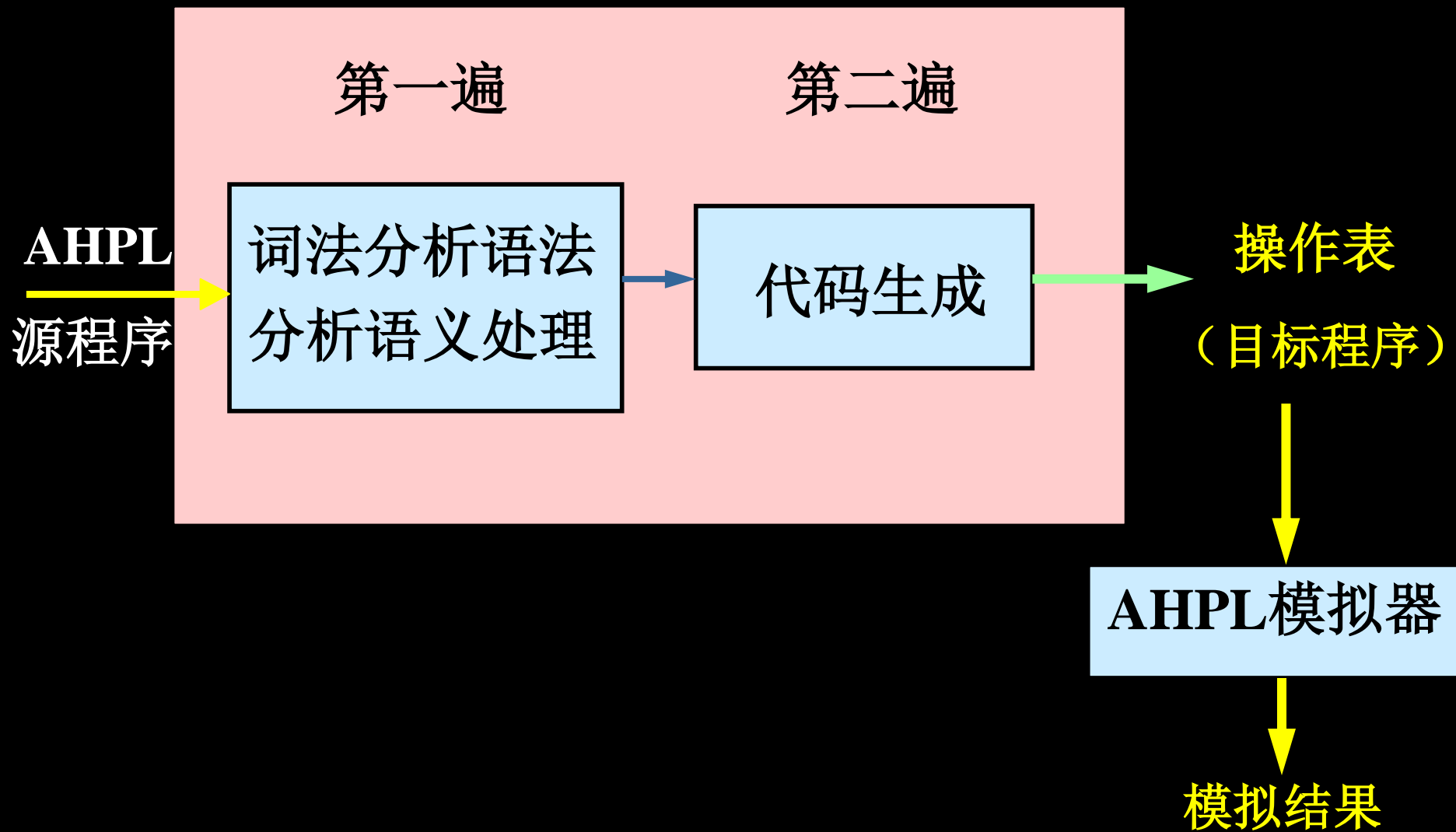
第一章 编译引论

- 1.1 程序设计语言与编译程序
- 1.2 编译程序的表示与分类
- 1.3 编译程序的结构与编译过程
- ~~1.4 语言开发环境中的伙伴程序~~
- 1.5 编译程序结构的实例模型 
- 1.6 编译程序的构造与实现

■ 一遍扫描的编译程序结构模型(Pascal编译器)



■ AHPL模拟器（两遍扫描的编译程序结构模型）



■ PDP-11 C编译器结构模型（三遍扫描的编译程序结构模型）

第一遍

词法分析语法分析
语义处理（中间代码生成）

中间代码

第二遍

代 码 生 成

汇编代码

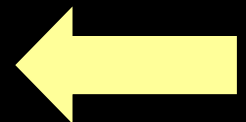
第三遍

代 码 优 化

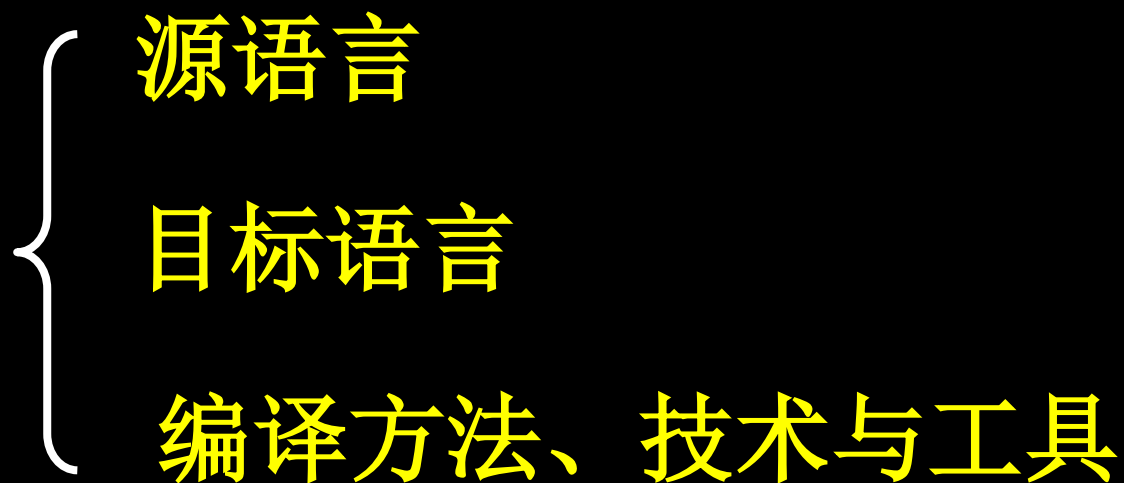
目标代码（汇编）

第一章 编译引论

- 1.1 程序设计语言与编译程序
- 1.2 编译程序的表示与分类
- 1.3 编译程序的结构与编译过程
- ~~1.4 语言开发环境中的伙伴程序~~
- 1.5 编译程序结构的实例模型
- 1.6 编译程序的构造与实现

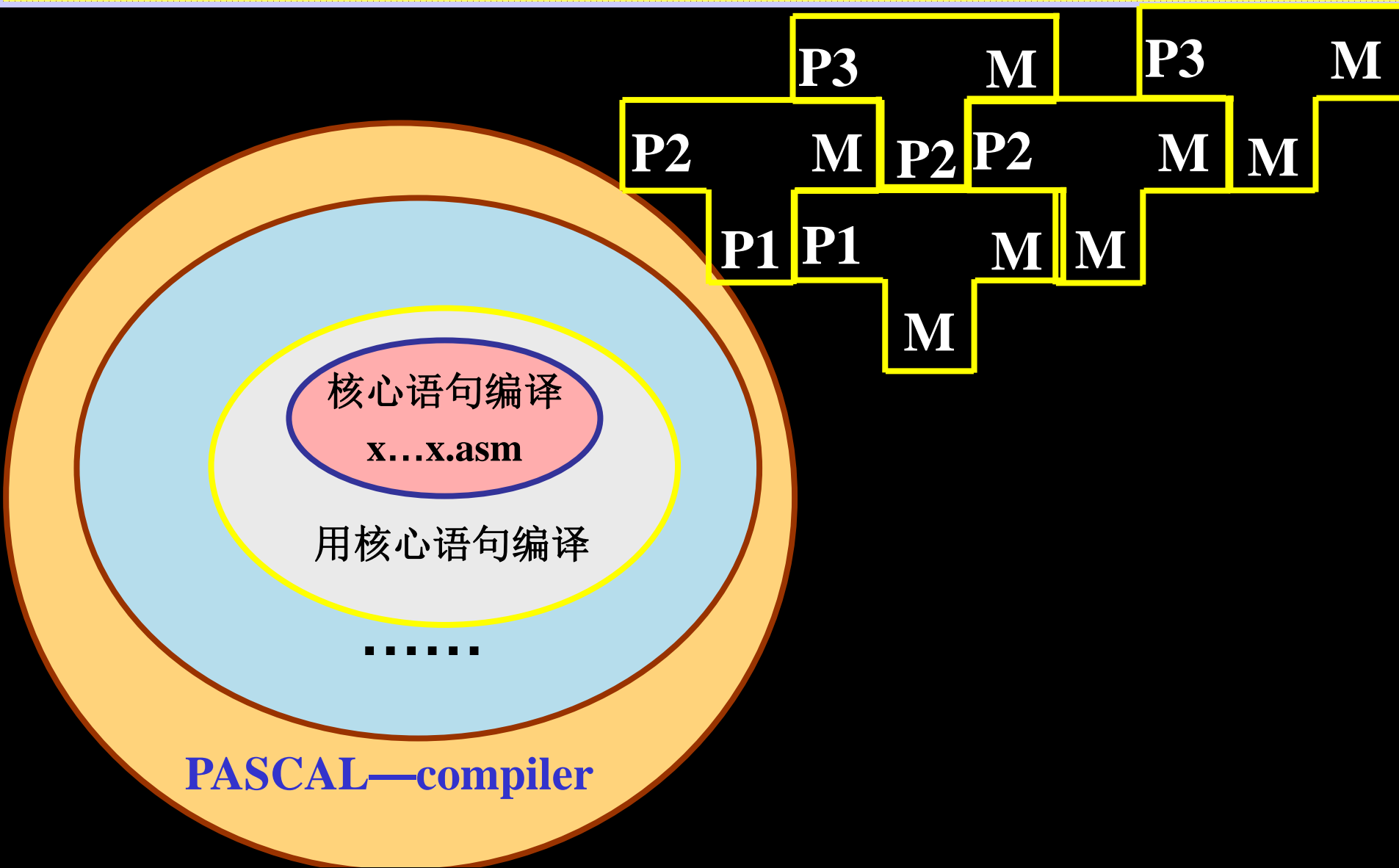


■ 编译程序构造要素



■ 编译程序的生成

- 1) 使用编程语言;
- 2) 移植方式;
- 3) 自编译 (用源语言作为宿主语言) ;



自编译（用源语言作为宿主语言）

■ 编译程序的生成

- 1) 使用编程语言;
- 2) 移植方式;
- 3) 自编译 (用源语言作为宿主语言);
- 4) 自动生成。

TWS (**T**ranslater **W**riting **S**ystem)

词法分析器自动生成器**LEX** **FLEX**

语法分析器自动生成器**YACC** **Bison**

语法制导翻译器

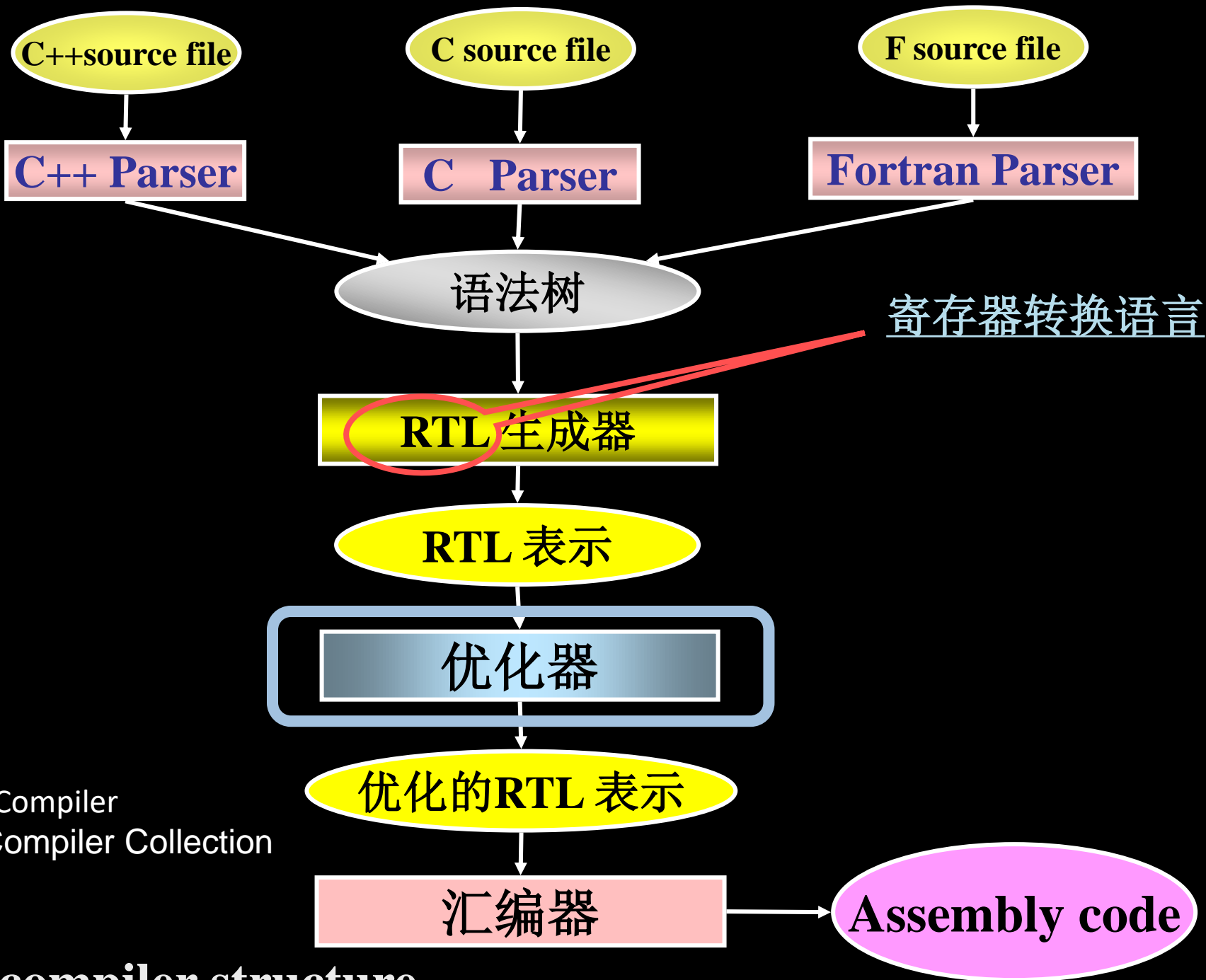
代码自动生成器

数据流分析装置



- 程序设计语言
- 编译引论
- GCC和LLVM介绍

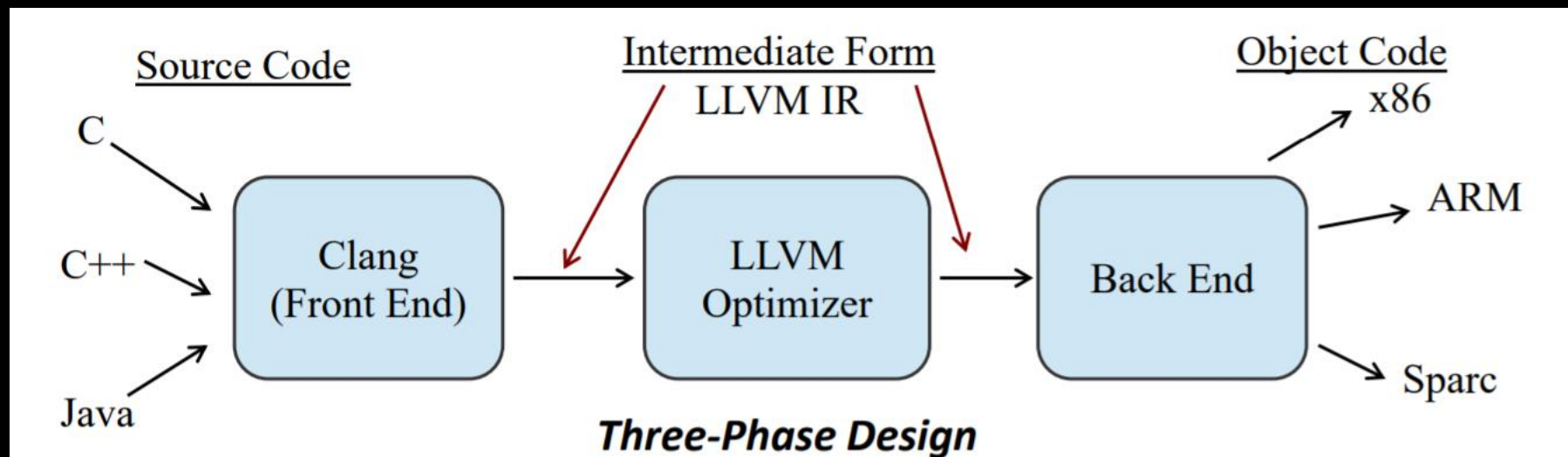




GNU C Compiler
GNU Compiler Collection

Gcc compiler structure

LLVM(Low Level Virtual Machine)





- 在Linux平台上安装和运行工业界常用的编译器GCC和LLVM，如果系统中没有安装，则需要首先安装编译器，安装完成后编写简单的测试程序，使用编译器编译，并观察中间输出结果。
- 具体要求见乐学平台上相关材料

第一章提要

- ◆ 编译程序的定义。
- ◆ 源程序的两种运行模式。编译程序的分类与表示。
- ◆ 术语：源语言、源程序、目标语言、目标程序、宿主语言、宿主机（目标机）、遍。
- ◆ 编译程序的组成结构,各部分间逻辑关系和主要功能。
- ◆ 编译程序的构造要素。
- ◆ 编译程序的组织及生成。

第一章

end