编译原理

数据科学与计算机学院

陈炬桦

实验报告提交：1967074105@qq.com

isscjh@mail.sysu.edu.cn

QQ：597371232

课程简介

课程内容

- 介绍编译器构造的一般原理和基本实现方法；
- 介绍的理论知识：形式语言：(文法、正规表达式、自动机理论)，语法制导翻译方法，优化等。

编译原理

- 编译：高级语言程序=>机器语言
- 原理：一般方法
- 目标：输入=>输出
- 如：机器人的动作；人员工作安排。

形式语言

- 数学语言—精确语言
- 自然语言—不精确，二义性

- 一种数学模型

数学模型

- 有多种表示形式

描述图像色彩，有以下几种：

（1）RGB模式（R：红色，G：绿色，B：蓝色）

（2）HSB模式

H：色调，表明光谱波长，决定颜色，如红、绿等色

S：饱和度，颜色纯度从0%（灰色）→100%（纯色）

B：亮度，0%（黑色）→ 100%（白色）

（3）索引颜色模式：使用调色板（注：相当于一个由颜色索引值到RGB颜色值的映射表）存放并索引图像中的颜色。

圆（椭圆）的表示

- 笛卡尔坐标--直角坐标(x,y)
- 极坐标(r,θ)
- 其它坐标(交叉直线，交叉曲线)

课程简介

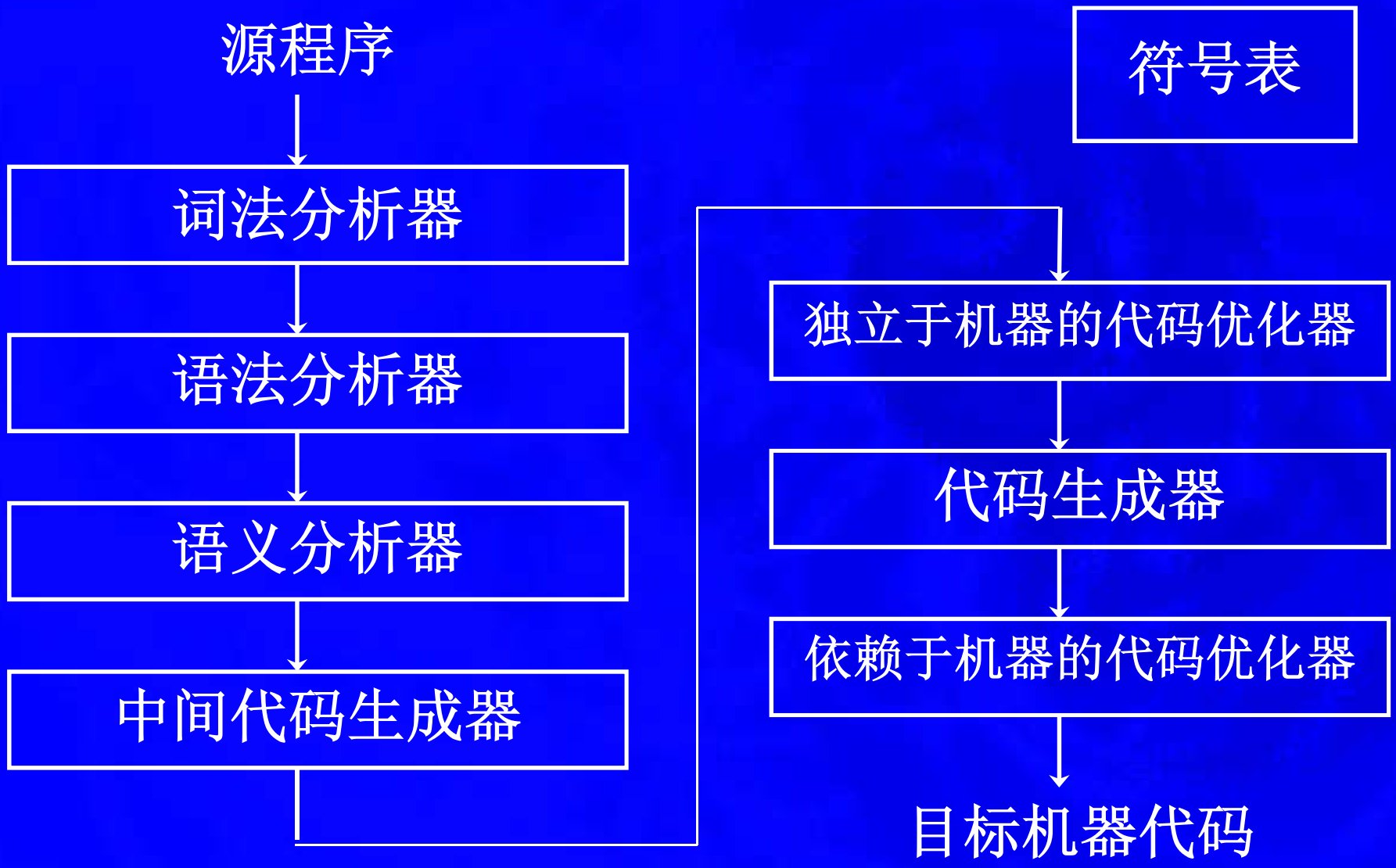
教材和参考书

- 陈意云、张昱，*编译原理*，高等教育出版社，2003，2008
- **A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools* , 2nd edition, Addison-Wesley, 2007**
- 何炎祥，编译原理
- **http://222.200.185.45/**
- **(soj.sysu.edu.cn)**
- **cconline.sysu.edu.cn**

第一章　引　论

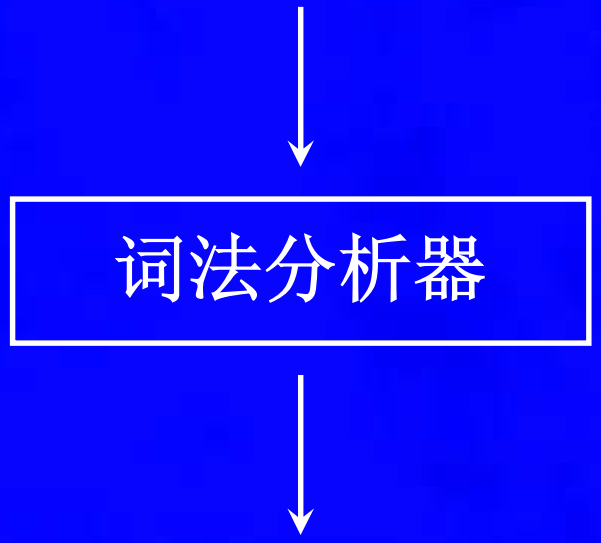
- 翻译器(translator)、编译器(compiler)、解释器(interpreter)
- 编译器从逻辑上可以分成若干阶段
- 每个阶段把源程序从一种表示变换成另一种表示
- 本章通过描述编译器的各个阶段来介绍编译这个课题

1.1 编译器概述


源程序
词法分析器
语法分析器
语义分析器
中间代码生成器
符号表
独立于机器的代码优化器
代码生成器
依赖于机器的代码优化器
目标机器代码

1.1 编译器概述

position = initial + rate * 60



⟨id, 1⟩ ⟨=⟩ ⟨id, 2⟩ ⟨+⟩ ⟨id, 3⟩ ⟨*⟩ ⟨60⟩

符 号 表

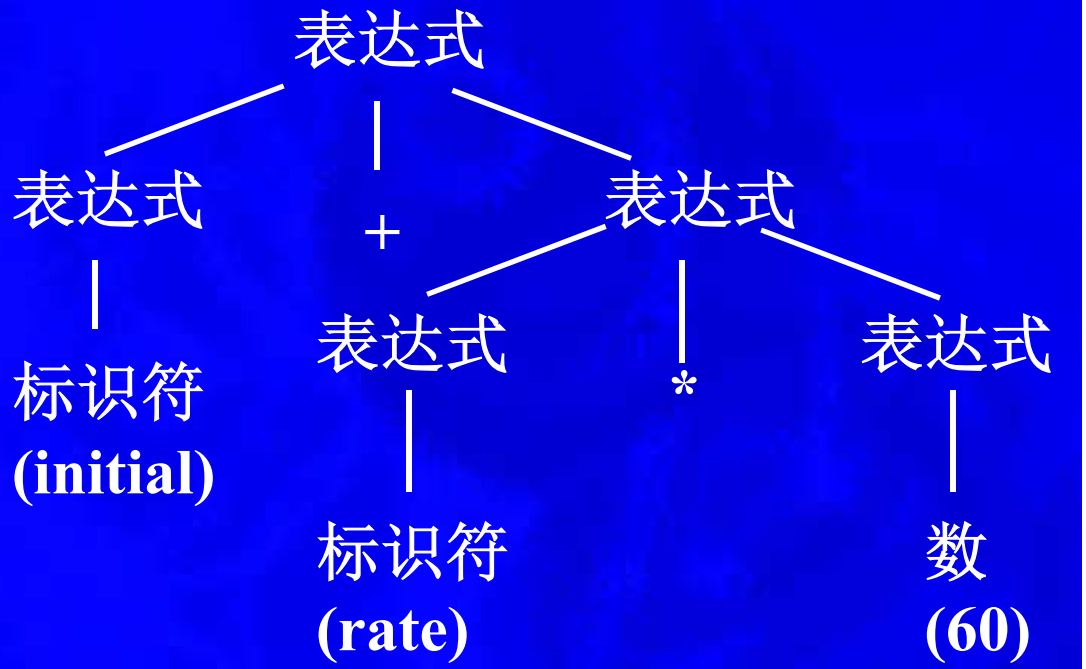
1	position	· · ·
2	initial	· · ·
3	rate	· · ·

1.1 编译器概述

表达式的语法特征

- 任何一个标识符都是表达式
- 任何一个数都是表达式
- 如果e_1和e_2都是表达式，那么
  - $e_1 + e_2$
  - $e_1 * e_2$
  - (e_1)

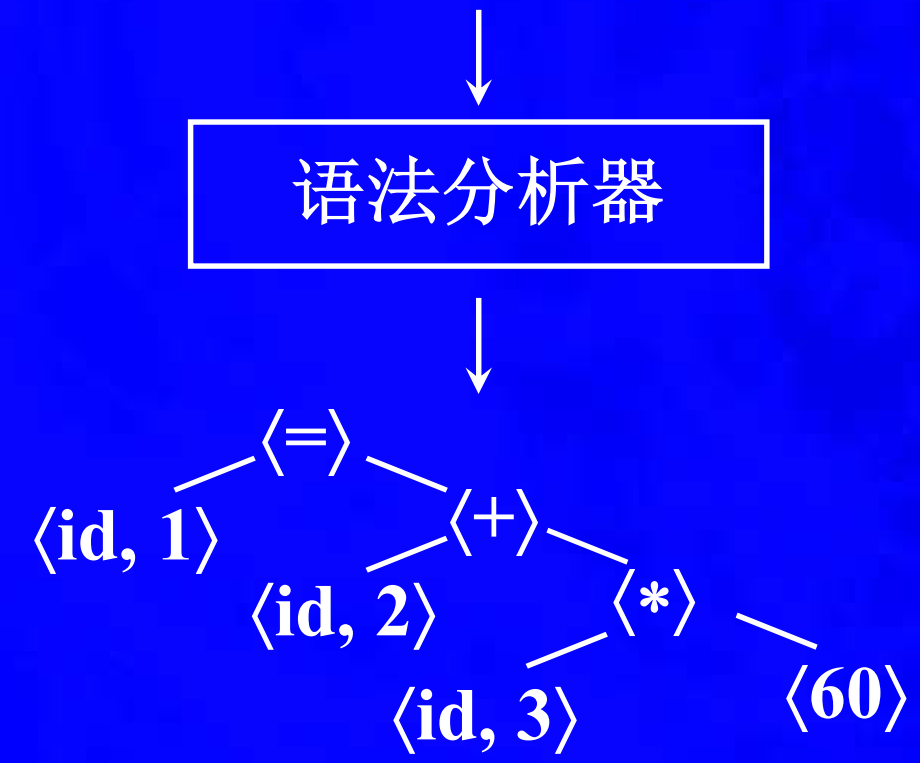
也都是表达式



initial + rate * 60的分析树

1.1 编译器概述

⟨id, 1⟩ ⟨=⟩ ⟨id, 2⟩ ⟨+⟩ ⟨id, 3⟩ ⟨*⟩ ⟨60⟩

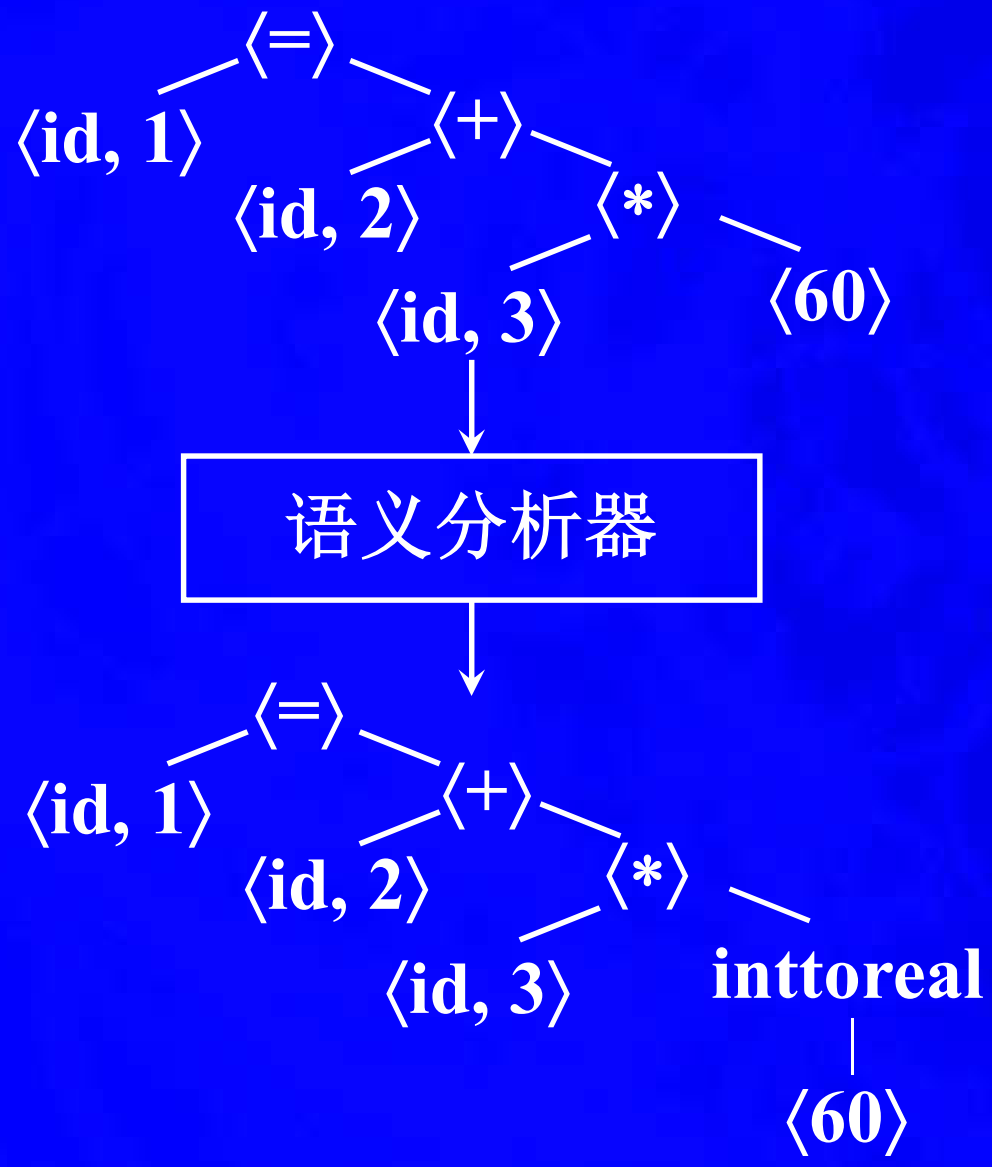


← 语法树

符 号 表

1	position	· · ·
2	initial	· · ·
3	rate	· · ·

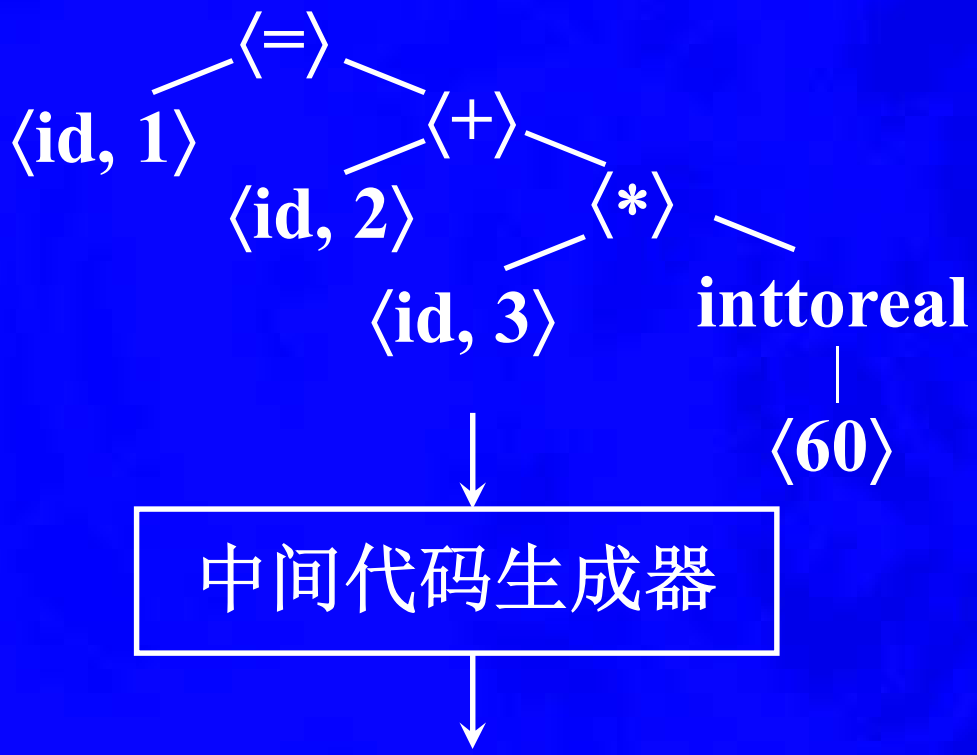
1.1 编译器概述



符 号 表

1	position	···
2	initial	···
3	rate	···

1.1 编译器概述



符 号 表

1	position	···
2	initial	···
3	rate	···

t1 = inttoreal(60) 整数转实数（浮点数）

t2 = id3 * t1

t3 = id2 + t2

id1 = t3

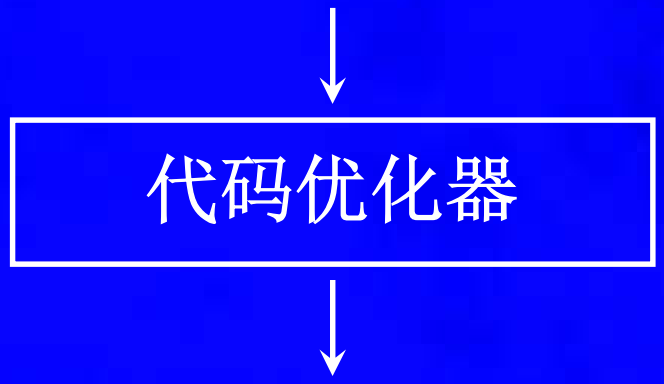
1.1 编译器概述

t1 = inttoreal(60)

t2 = id3 * t1

t3 = id2 + t2

id1 = t3



t1 = id3 * 60.0

id1 = id2 + t1

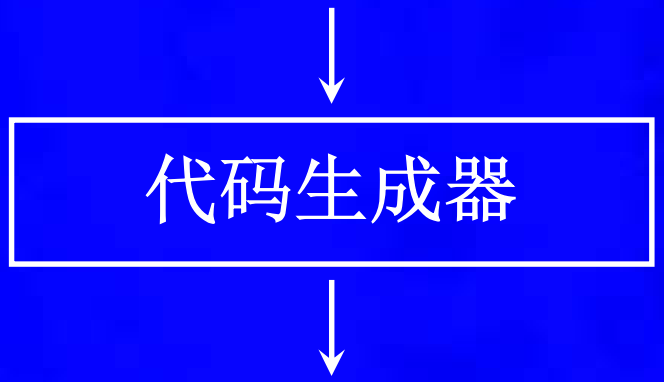
符 号 表

1	position	···
2	initial	···
3	rate	···

1.1 编译器概述

t1 = id3 * 60.0

id1 = id2 + t1



MOVF id3, R2

MULF #60.0, R2

MOVF id2, R1

ADDF R2, R1

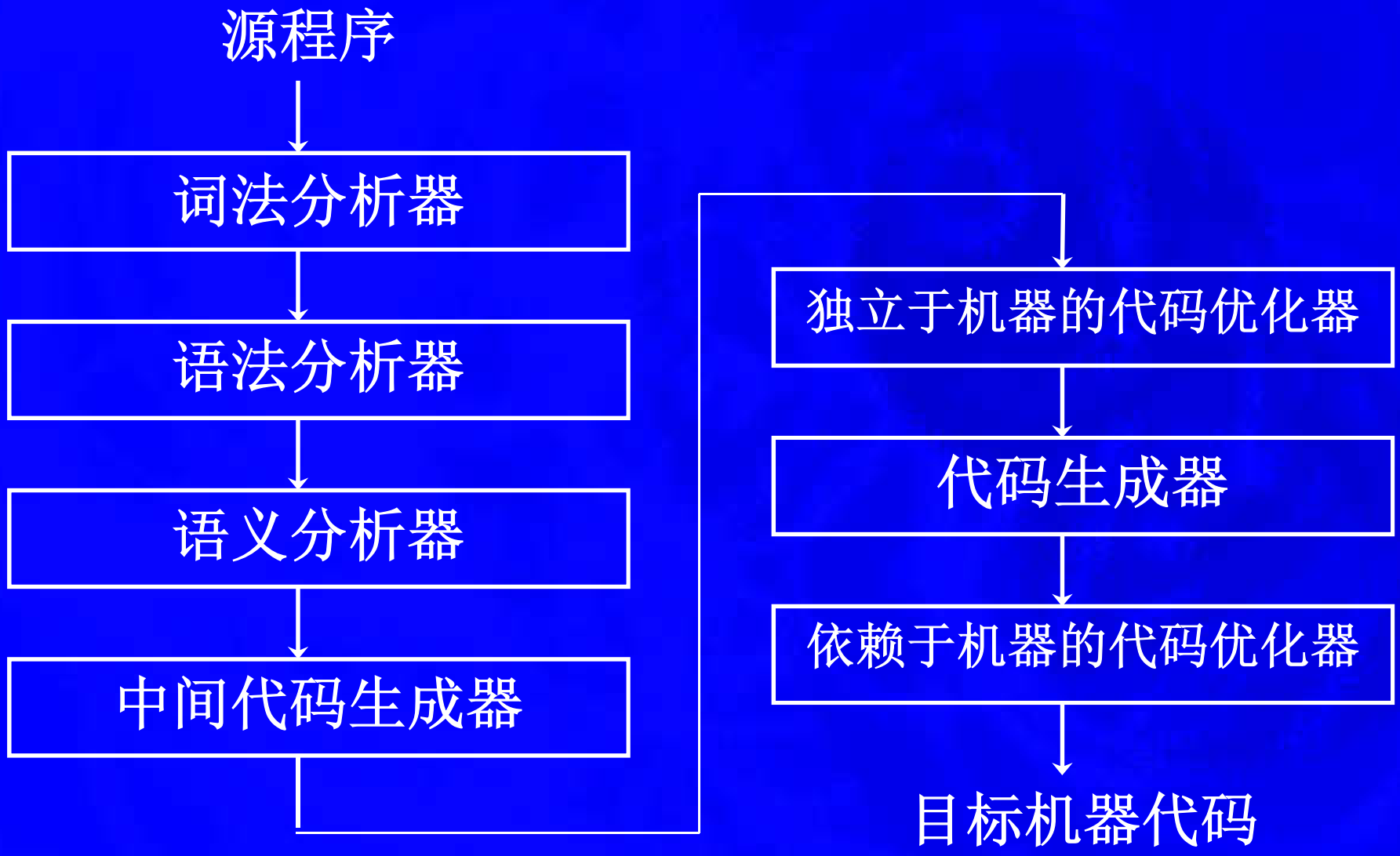
MOVF R1, id1

符 号 表

1	position	···
2	initial	···
3	rate	···

解释器和编译器的区别

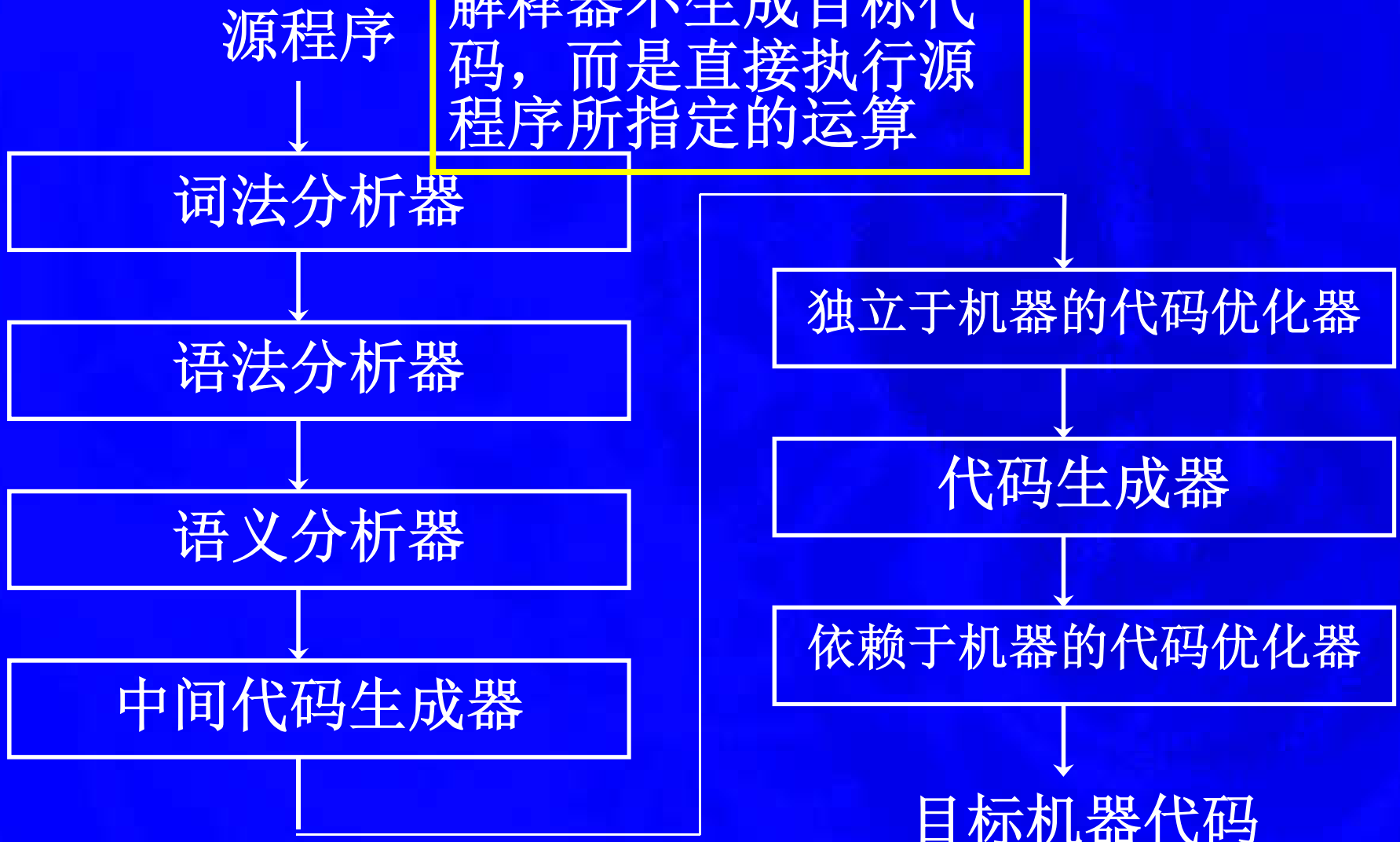
1.1 编译器概述



1.1 编译器概述

解释器和编译器的区别

解释器不生成目标代码，而是直接执行源程序所指定的运算

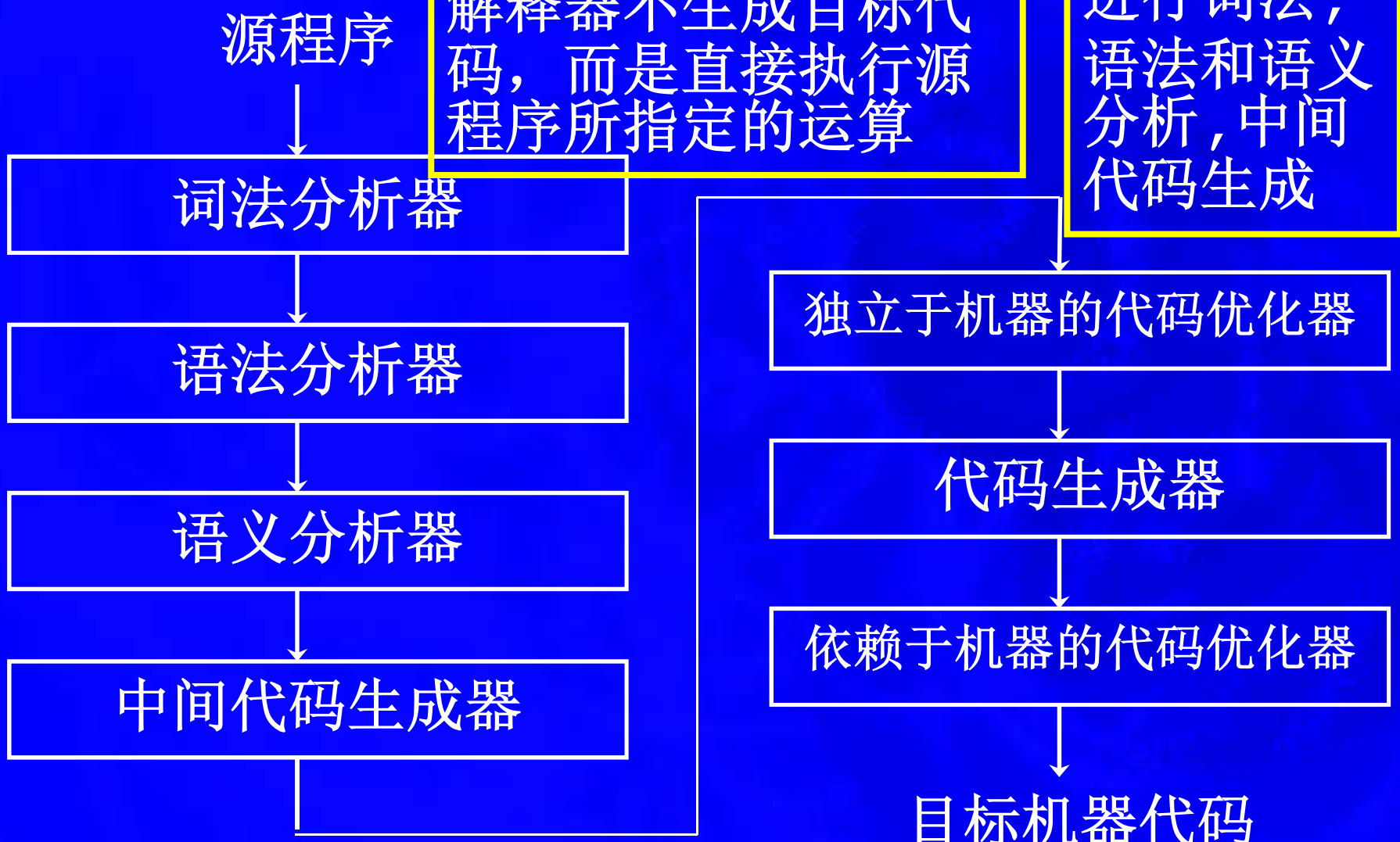


1.1 编译器概述

解释器和编译器的区别

解释器不生成目标代码，而是直接执行源程序所指定的运算

解释器也需要对源程序进行词法，语法和语义分析，中间代码生成

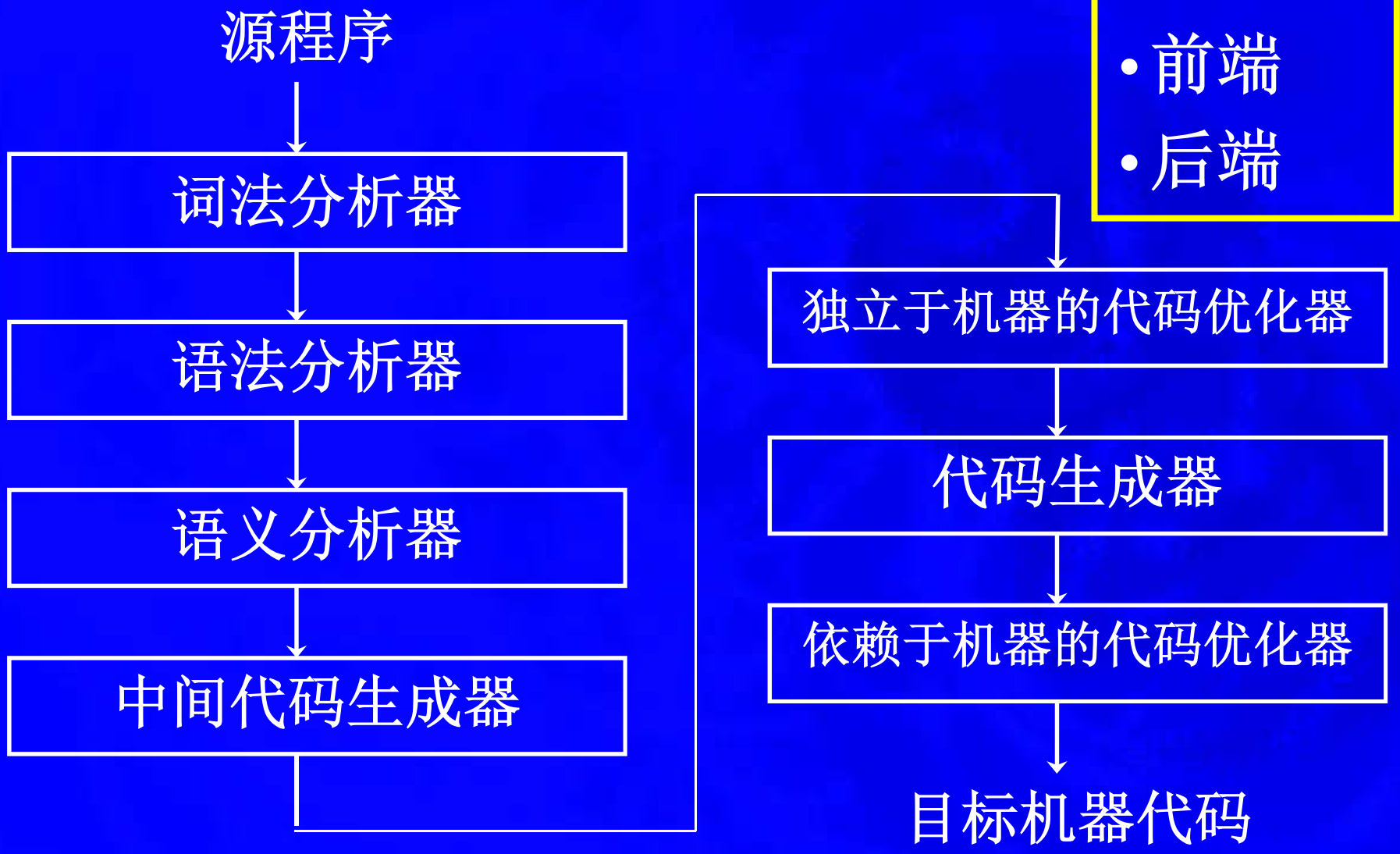


1.1 编译器概述

- **BASIC语言年代解释器**
  - 功能：它将高级语言的源程序翻译成一种中间语言程序，然后对中间语言程序进行解释执行
  - 在那个年代，编译和解释两个功能是合在一个程序中，该程序被称为解释器
- **Java语言年代解释器**
  - 解释器的上述两个功能分在两个程序中
  - 前一个编译器，它把源程序翻译成一种叫做字节码的中间语言程序
  - 后一个叫做解释器，它对字节码程序进行解释执行

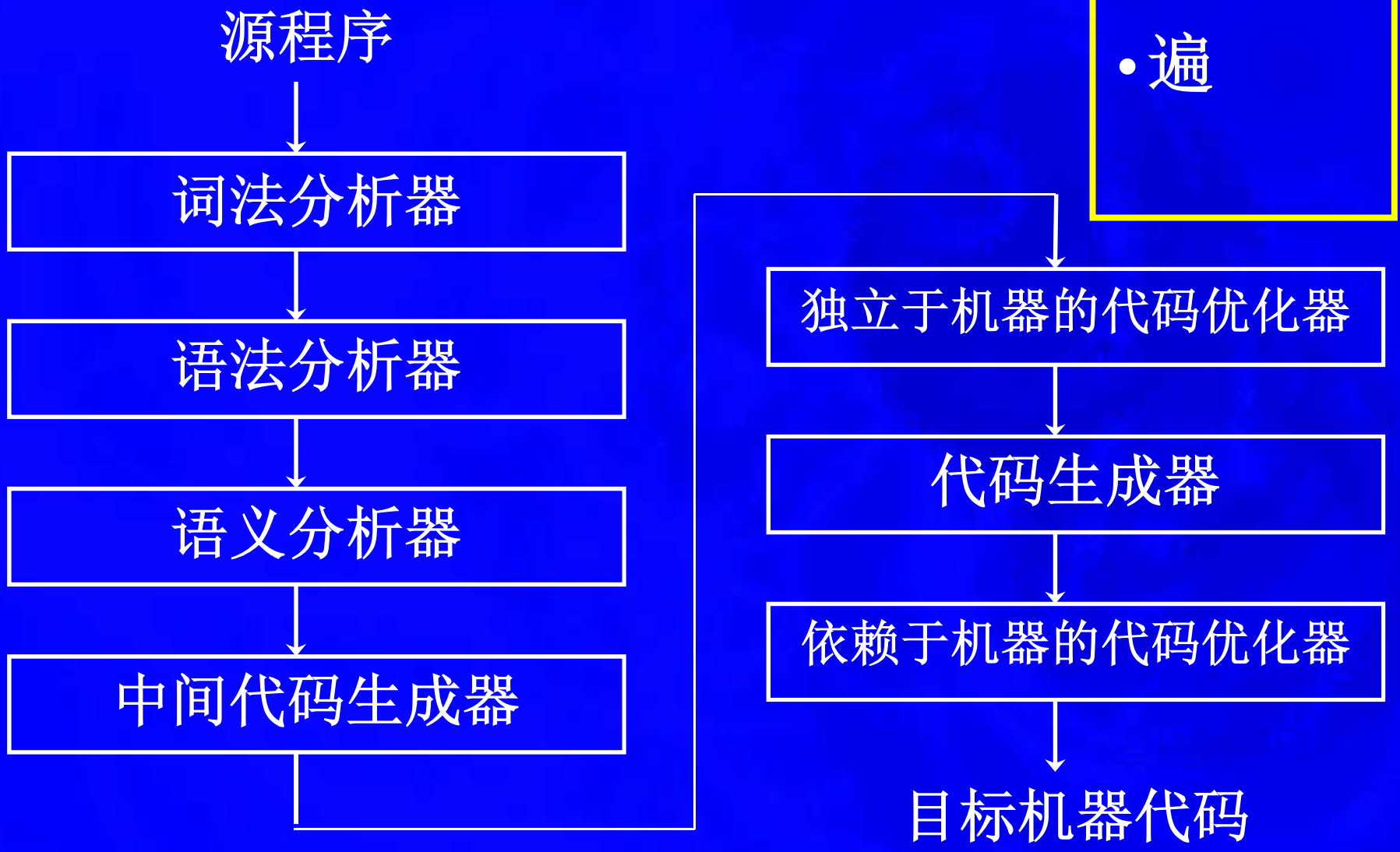
1.1 编译器概述

阶段分组
- 前端
- 后端



1.1 编译器概述

阶段分组
- 遍



1.2 编译器技术的应用

- 高级语言的实现
  - 高级编程语言易于编程，但程序运行较慢
  - 低级语言编程时可实施更有效的控制方式，得到更有效的代码，但难编写、易出错、难维护
  - 流行编程语言的大多数演变都是朝着提高抽象级别的方向
  - 每一轮编程语言新特征的出现都刺激编译器优化的新研究

1.2 编译器技术的应用

- 高级语言的实现

  每一轮编程语言新特征的出现都刺激编译器优化的新研究

  - 支持用户定义的聚合数据类型和高级控制流，如数组和记录、循环和过程调用：C、Fortran
  - 面向对象的主要概念是数据抽象和性质继承，使得程序更加模块化并易于维护：Smalltalk、C++、C#、Java
  - 类型安全的语言：Java没有指针，也不允许指针算术。它用无用单元收集机制来自动地释放那些不再使用的变量占据的内存
  - Java设计来支持代码移植和代码移动

1.2 编译器技术的应用

- 针对计算机体系结构的优化
  - 计算机体系结构的迅速演化引起对新的编译器技术一种不知足的需要
  - 并行化
    - 编译器重新整理指令，使得指令级并行更有效
    - 编译器从传统的串行程序自动生成并行代码，使之运行于多处理器上
  - 内存分层
    - 编译器优化历来集中在优化处理器的执行上，但是现在更强调要使内存分层更有效

1.2 编译器技术的应用

- 新计算机体系结构的设计
  - 现在计算机系统的性能不仅仅取决于它的原始速度，还取决于编译器是否能生成充分利用其特征的代码
  - 在现代计算机体系结构的研究中，在处理器的设计阶段就开发编译器，并将编译生成的代码在模拟器上运行，以评价拟采用体系结构的特征
  - 编译器技术影响计算机体系结构设计的一个著名例子是精简指令集计算机（RISC）的发明

1.2 编译器技术的应用

- 程序翻译
  - 二进制翻译
    - 编译器技术可用于把一种机器的二进制代码翻译成另一种机器的代码，以运行原先为别的指令集编译的代码
  - 数据库查询解释器
    - 数据库查询由一些谓词组成，这些谓词由包含关系运算的布尔表达式组成，可以被解释执行，也可以被编译成搜索数据库的命令