编 译 原 理

第一章 引论

中国海洋大学 计算机系 王欣捷

wangxinjie@ouc.edu.cn

任课教师介绍

- 编译原理课程小组：
  - 葛琳、王欣捷
- 王欣捷：
  - 本科毕业于武汉大学计算机学院
  - 博士毕业于浙江大学计算机学院
  - 曾任网易资深图形架构师3年
  - 热爱编程，曾为NOI山东省省队成员
  - 希望和同学们一起进步！
- 联系方式：
  - 邮箱：wangxinjie@ouc.edu.cn
  - 办公室：西海岸校区信南C405

课程信息

- 上课时间：周五 3-4节、5-6节
- 上课地点：西海岸学习综合体南415/416
- 7-16周5-6节为实验，信南B227/B231

- 助教：杜艺伟
  - duyiwei@stu.ouc.edu.cn
  - 微信号可从微信群里添加

课程信息

编译原理课程安排表

作业 本次课后要布置作业
报告 每两次实验写成一个报告

周五 3-4节 南415/416
5-6节 南415/416
5-6节 信南B227/B231（7-16周）

2023年12月26日校长办公会审议通过

中国海洋大学2024年春季学期校历

月份	二月		三月				四月				五月					六月			
周次		一	二	三	四	五	六	七	八	九	十	十一	十二	十三	十四	十五	十六	十七	十八
星期二																		18双周五	
星期五	23	1	8	15	22	29	5	12	19	26双周五	3	10	17	24	31	7	14	21	28
星期日							7双周五												
3-4节 10:30-12:15		1 引论	2 词法分析	2 习题课	作业2 3 语法分析	3 语法分析	作业3 3 语法分析	3 语法分析		作业4 3 语法分析		3 习题课	4 语法制导的翻译	4 语法制导的翻译	作业5 5 类型检查	5 类型检查	6 存储管理	6 存储管理	7 中间代码生成
5-6节 13:30-15:20		2 词法分析	作业1 2 词法分析	3 语法分析	3 语法分析	3 语法分析	3 语法分析	实验一		报告1 实验二		实验三	报告2 实验四	实验五	报告3 实验六	实验七	报告4 实验八	作业6 中间代码生成	8 复习课

BlackBoard（海大数字教学平台）

- 网址：https://wlkc.ouc.edu.cn/
- 手机APP：Blackboard

- **作业、课件**等信息会发布在课堂派平台上

请同学们通过BlackBoard提交作业和实验报告

头歌实验平台

- 网址：https://www.educoder.net/classrooms/FY8A4UCN
- 微信公众号：头歌
- **实验内容、签到**等信息会发布在头歌课堂上
- 实验分组：5人一组，每位同学都要在平台上完成实验，但实验报告可以共同撰写成1份

请同学们通过
头歌平台做实验

授课大纲

- 编译器概述（第一章）
- 词法分析（第二章）
- 上下文无关文法和语法分析（第三章）
- 语法制导翻译和类型检查（第四、五章）
- 运行时存储空间的组织和管理（第六章）
- 中间代码生成（第七章）
- 2次习题课，1次复习课

课程要求

- 本课程共64学时，其中讲授48学时，实验16学时。
- 学期成绩评分体系：
  - 课下作业、课堂讨论及平常表现：20%
  - 实验课出勤及实验表现、实验报告成绩：20%
  - 期末考试成绩：60%

- 由于概念较多，因此要求大家：
  - 上课带好教材和草稿本（或平板），有随堂练习
  - 认真听，理解并牢记各种概念
  - 课前预习、课后复习并加深理解，多看书和课件
  - 独立认真完成作业
  - 课程实验：自己动手

课程安排

本学期课程安排紧凑，周五一整天要进行四节课。
知识点密集，请同学们一定要预习！

作业 本次课后要布置作业
报告 每两次实验写成一个报告

周五 3-4节 南415/416
5-6节 南415/416
5-6节 信南B227/B231（7-16周）

2023年12月26日校长办公会审议通过

中国海洋大学2024年春季学期校历

月份	二月		三月				四月				五月					六月			
周次		一	二	三	四	五	六	七	八	九	十	十一	十二	十三	十四	十五	十六	十七	十八
星期二																		18双周五	
星期五	23	1	8	15	22	29	5	12	19	26双周五	3	10	17	24	31	7	14	21	28
星期日							7双周五												
3-4节 10:30-12:15		1 引论	2 词法分析	2 习题课	作业2 3 语法分析	3 语法分析	作业3 3 语法分析	3 语法分析		作业4 3 语法分析		3 习题课	4 语法制导的翻译	4 语法制导的翻译	作业5 5 类型检查	5 类型检查	6 存储管理	6 存储管理	7 中间代码生成
5-6节 13:30-15:20		2 词法分析	作业1 2 词法分析	3 语法分析	3 语法分析	3 语法分析	3 语法分析	实验一		报告1 实验二		实验三	报告2 实验四	实验五	报告3 实验六	实验七	报告4 实验八	作业6 中间代码生成	8 复习课

教学目标

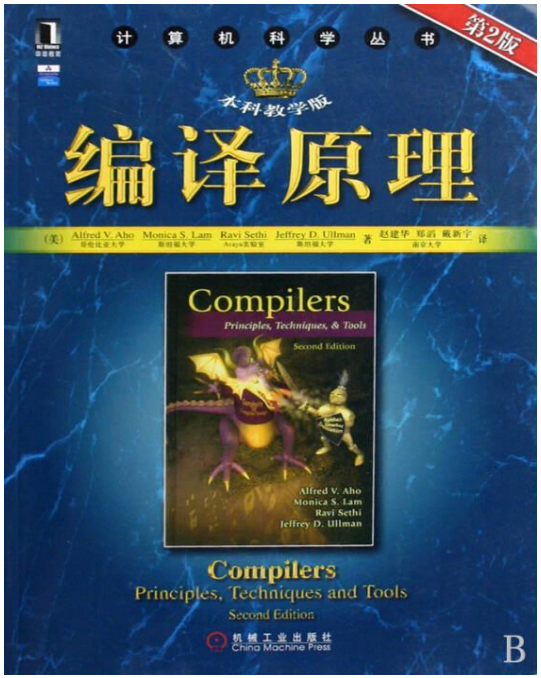
- 掌握编译器构造的一般原理和基本实现方法。
- 不偏向于某种源语言或目标机器，而是强调对编译原理和技术的宏观理解。

前驱课程

- 学习这门课程之前请确定你有一定计算机专业背景，并且已经学过下列课程：
  - 至少一门编程语言（比如C/C++/Java……）。
  - 数据结构、离散数学等，并有一定的编程经验。
- 拥有下列知识将对你学习这门课程很有帮助：
  - 计算机系统原理、体系结构
  - 算法和软件工程
  - 更多编程语言知识（例如Pascal、Fortran……）

教材和参考书

- 教材：《编译原理》，陈意云、张昱，第3版，高等教育出版社，2014
- 《编译原理》本科教学版， Alfred Aho 等著，赵建华等译，机械工业出版社，2010.3



学习编译原理的意义

- 编程工作中经常会遇到语言的设计，虽然是一些简单的语言（如输入输出），本课程有助于提高语言的设计水平。
- 能够确切地知道你所编写的程序是如何从高级语言程序变成可执行程序的，从而对编程有更深入的理解；进而可以提高纠错能力和编写高质量代码的能力。这种能力对大型软件的构造和维护尤为重要。
- 编译原理集中体现了计算机科学的很多核心思想：算法、数据结构，软件工程等，是其他领域的重要研究基础。编译器设计中的一些原理和技术足以应用到你今后职业生涯中的很多领域。

学习编译原理的意义

- 2019 年 5 月 15 日，美国商务部工业与安全局发表正式声明，华为及其在 26 个国家的 68家关系企业被列入出口管制的“**实体名单**”。英特尔、高通和博通三大芯片公司做出了**暂停为华为供货**的决定。
- 2019年8月31日，华为推出了首个完全**自主研发的方舟编译器**，可以支持多种编程语言、多种芯片平台的联合编译、运行，包含编译器、工具链、运行时环境等关键部件。
- 2020年3月17日，Java14发布，**龙芯中科JVM团队**在其中发挥了重要作用，为其错误修复工作做出了大量贡献。
- 2023年6月，搭载中国自主研发处理器的**神威·太湖之光**在全球超级计算机500强排名位列第七。
- 尽管取得了诸多成绩，但是核心技术缺乏，创新能力不足，导致国内的编译器、操作系统等系统软件的发展与国外差距**依然很大**。

学习编译原理的意义

- 2019 年 5 月 15 日 发表正式声明，华为及其在 26 个国家 管制的“**实体名单**”。英特尔、高通和 **为华为供货**的决定。
- 2019年8月31日， **发的方舟编译器**，可以支持多种编程 译、运行，包含编译器、工具链、运
- 2020年3月17日， **团队**在其中发挥了重要作用，为其错
- 2023年6月，搭载 **太湖之光**在全球超级计算机500强排名
- 尽管取得了诸多 新能力不足，导致国内的编译器、操 外差距**依然很大**。

什么是编译

答：1是预处理，2是编译，3是链接。

- 同学们写完代码，总是要用gcc等工具来“生成”可执行文件。gcc处理代码有好几个过程。
- 猜猜下列哪个是编译过程？

1

```
// This is my first program!
#include <stdio.h>
#define PI 3.14
int main() {
    float x = PI;/*assign*/
    printf("%f\n", x);
    return 0;
}
```

```
# 1 ".../stdio.h" 1 3 4
extern "C" {
typedef signed char __int8_t;
......}
int main() {
 float x = 3.14;
 printf("%f\n", x);
 return 0;
}
```

2

```
# 1 ".../stdio.h" 1 3 4
extern "C" {
typedef signed char __int8_t;
......}
int main() {
 float x = 3.14;
 printf("%f\n", x);
 return 0;
}
```

```
.LC1:
        .ascii "%f\12\0"
        ......
main:
        pushq %rbp
        .seh_pushreg    %rbp
        movq  %rsp, %rbp
        .seh_setframe   %rbp
        subq  $48, %rsp
        ......
```

3

```
A4 01 00 00 00 00 00 00
40 00 50 40 55 48 89 E5
00 F3 0F 10 05 04 00 00
EF C0 F3 0F 5A 45 FC 66
48 [illegible] 6E C2 66 0F 28 C8
[illegible] E8 00 00 00 00 90
```

2和3中间还有一个过程叫做汇编，有时编译和汇编合并在编译过程中。

什么是编译

编译：将高级语言翻译成汇编语言或机器语言的过程
源语言 目标语言

类似数学定义或自然语言的简洁形式

- 接近人类表达习惯
- 不依赖于特定机器
- 编写效率高

高级语言
(High Level Language)

引入助记符

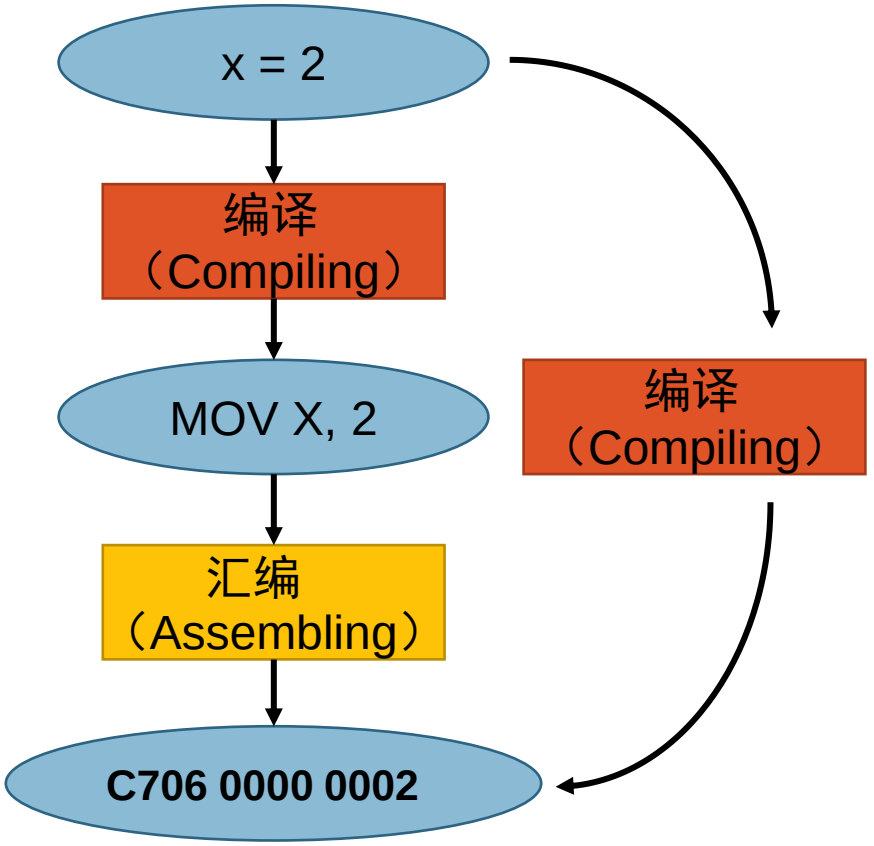
- 依赖于特定机器，非计算机专业人员使用受限制
- 编写效率依然很低

汇编语言
(Assembly Language)

可以被计算机直接理解

- 与人类表达习惯相去甚远
- 难记忆
- 难编写、难阅读
- 易写错

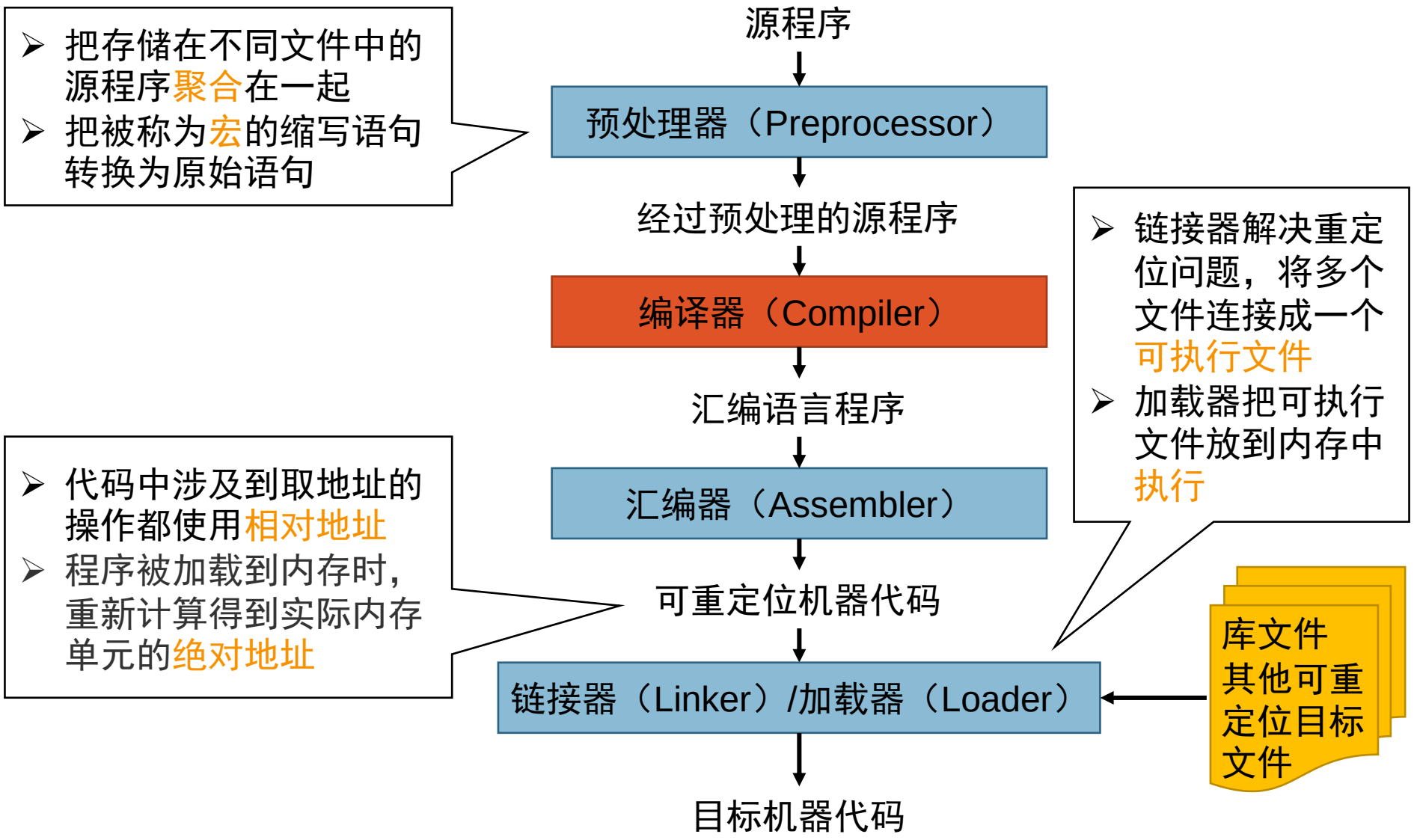
机器语言
(Machine Language)



本节任务

- 编译的概念
- 高级语言、汇编语言、机器语言的概念
- 源语言、目标语言的概念
- 编译器的几个基本阶段，及各阶段的功能
- 编译器阶段的分组：前端、后端；遍
- 编译器、解释器的区别
- 编译器技术的应用*

编译器在语言处理系统中的位置



编译器的结构

```
#include<stdio.h>
int main()
{
    int a,b,ge,shi,bai,m,n,i,number;
    printf("请输入(输入完两个数按一次回车键): \n");
    scanf("%d %d",&a,&b);
    while(a!=0,b!=0)
        scanf("%d %d",&a,&b);
    if(a>=100,b<=999)
        if(a>b)
            m=b,n=a;
        else
            m=a,n=b;
        for(i=m;i<=n;i++)
        {
            bai=i/100;
            shi=(i%100)/10;
            ge=i%10;
            if(i==bai*bai*bai+shi*shi*
            printf("%5d",i);
            number++;
            printf("\n");
        }
        if(number==0)
            printf("no\n");
}
```

机器是怎么翻译的？

```
59      call sc
60      mov  al, [si - 1]
61      call sc
62      mov  al, [si]
63      call sc
64      mov  al, ' '
65      call sc
66      ret
67  storechr endp
68  ;清屏，无入口参数
69  clearcrt proc near
70      mov  ax, 0600h
71      mov  bh, 07h
72      mov  cx, 00h
73      mov  dx, 184fh
74      int  10h
75      ret
76  clearcrt endp
77  ;回车换行，入口参数，行数 no
78  nextlien proc near, no:word
79      mov  cx, no
        .if cx ==0
          mov  cx, 1
        .endif
        .repeat
          mov  ah, 02h
          mov  dl, cr
86        int  21h
87        mov  ah, 02h
88        mov  dl, lf
89        int  21h
90      .untilcxz
91      ret
92  nextlien endp
93          end
94
```

编译器

高级语言程序

汇编语言程序/机器语言程序

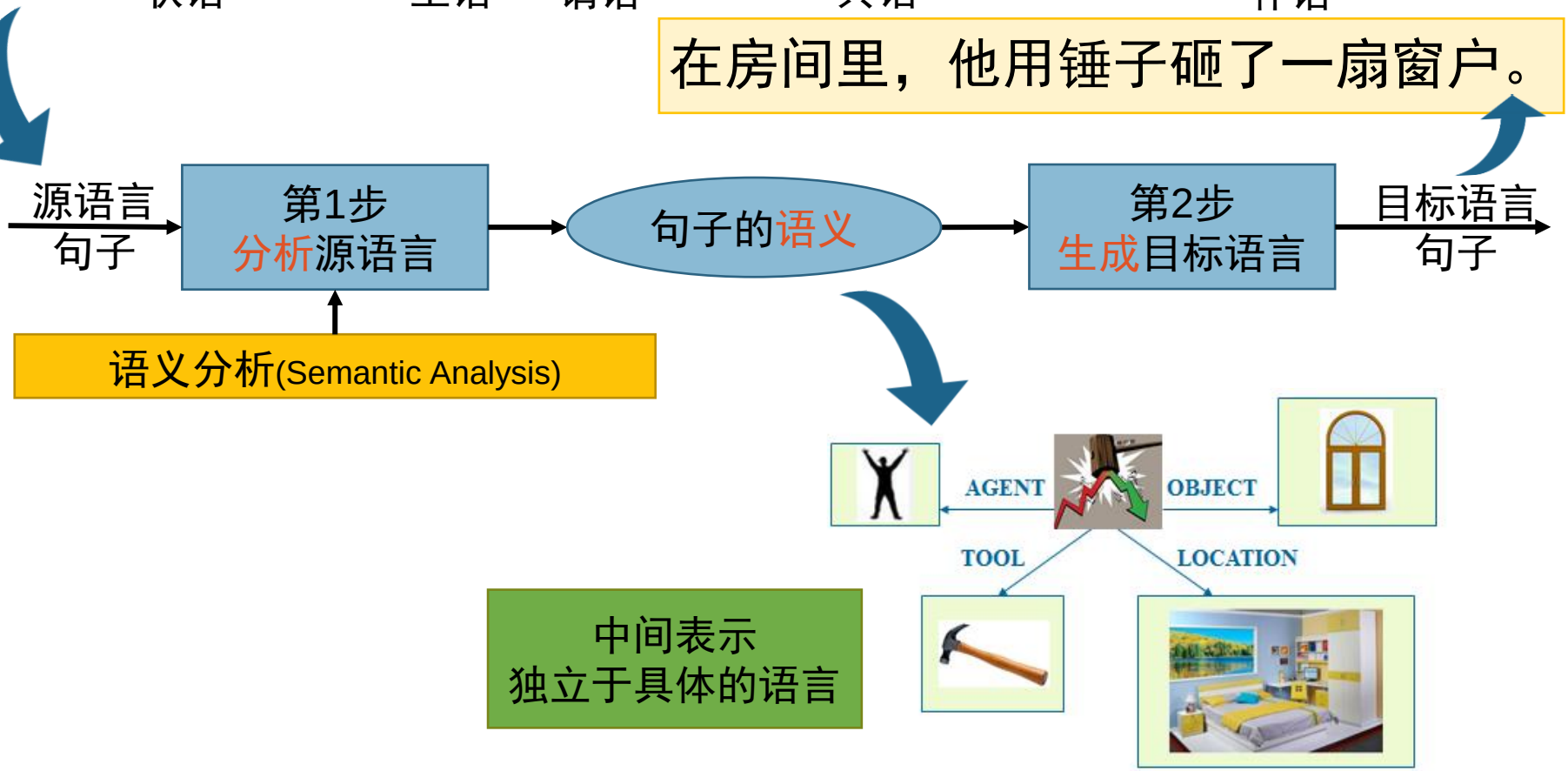
编译器的结构

- 你是如何把英文翻译成中文的？

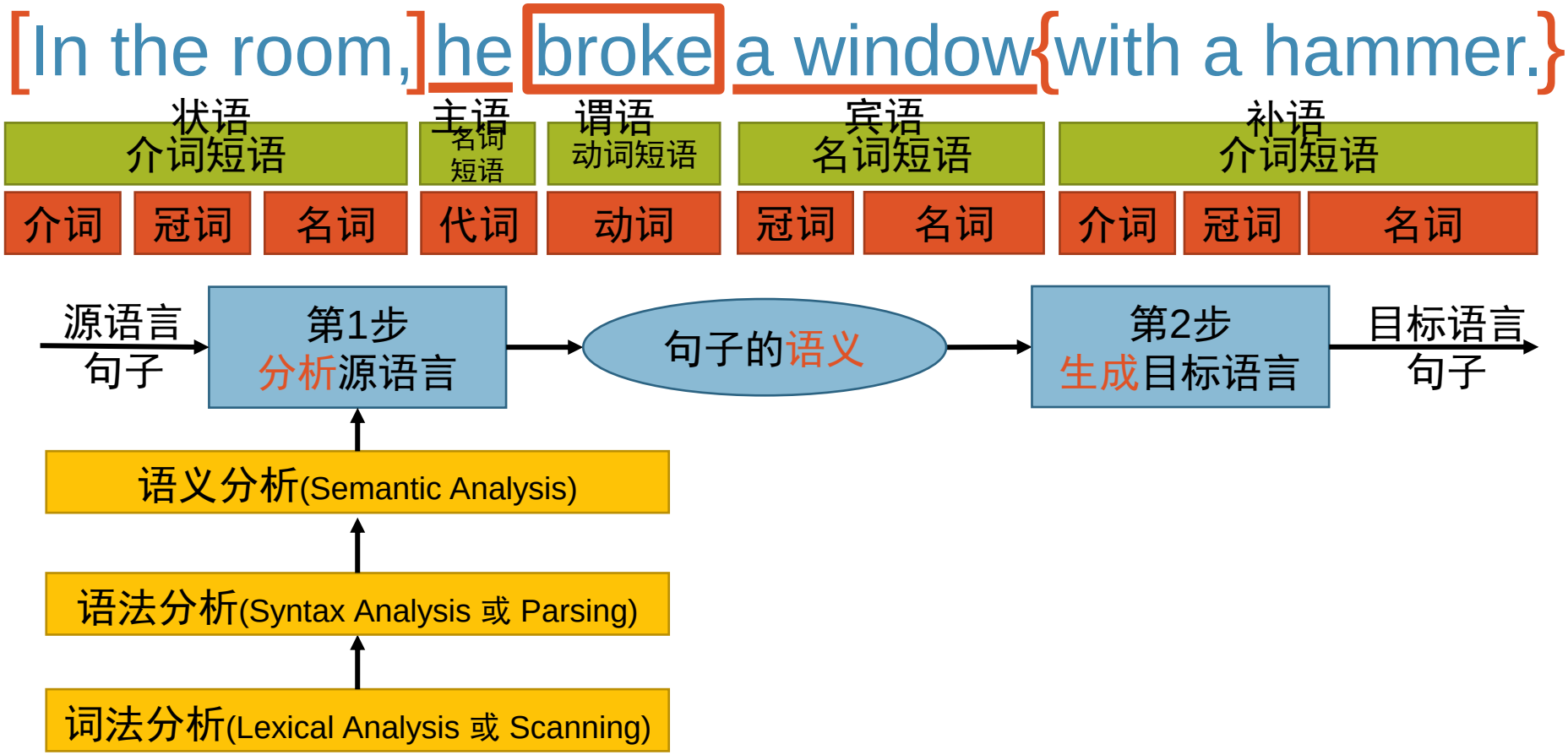
[In the room,] he broke a window {with a hammer.}

状语 主语 谓语 宾语 补语

在房间里，他用锤子砸了一扇窗户。



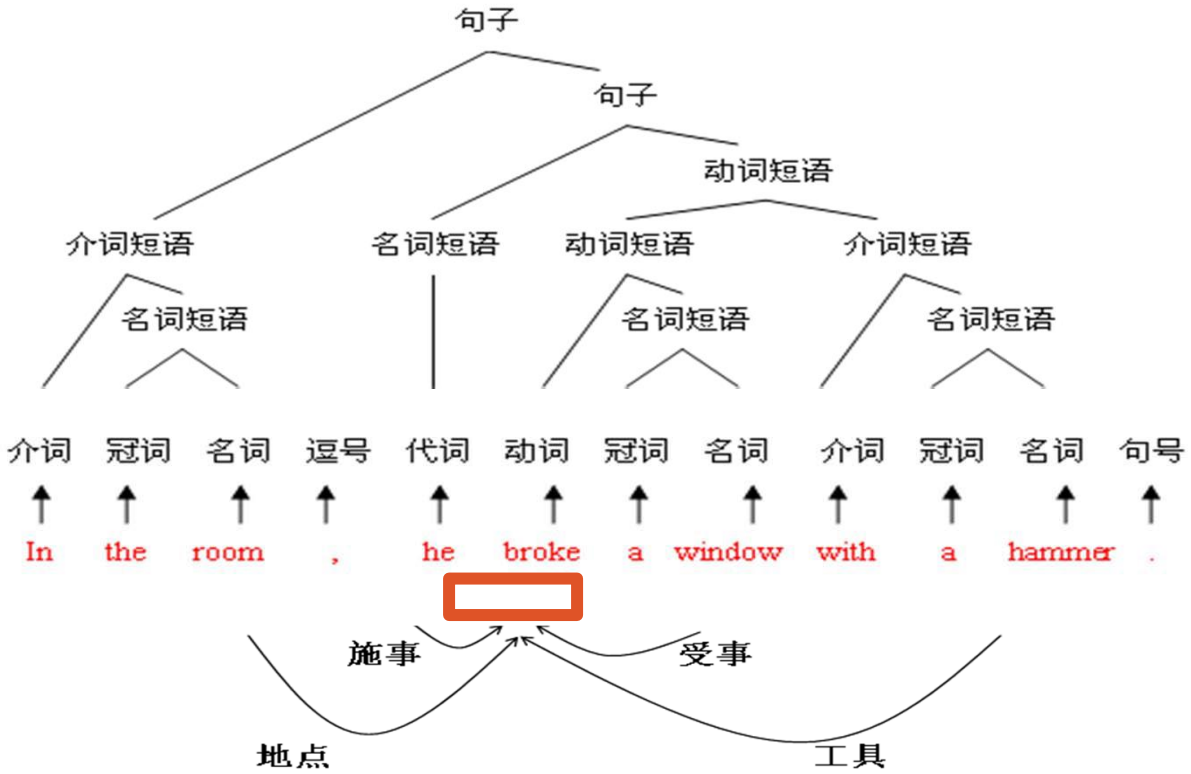
编译器的结构


[In the room,] he broke a window {with a hammer.}
状语
介词短语
介词
冠词
名词
主语
名词短语
代词
谓语
动词短语
动词
宾语
名词短语
冠词
名词
补语
介词短语
介词
冠词
名词
源语言
句子
第1步
分析源语言
句子的语义
第2步
生成目标语言
目标语言
句子
语义分析(Semantic Analysis)
语法分析(Syntax Analysis 或 Parsing)
词法分析(Lexical Analysis 或 Scanning)

编译器的结构

In the room, he broke a window with a hammer.

- 词法分析
- 语法分析
- 语义分析



编译器的结构

- 你用编程语言写的程序是怎样被计算机编译的？

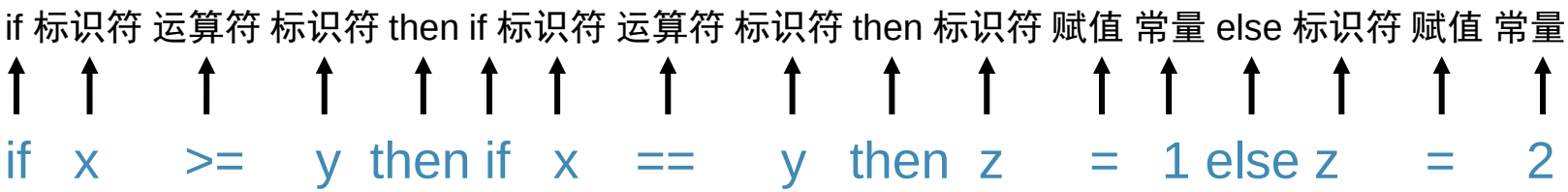
  if x>=y then if x==y then z = 1 else z = 2

编译器的结构

- 你用编程语言写的程序是怎样被计算机编译的？

  if x>=y then if x==y then z = 1 else z = 2

  - 词法分析

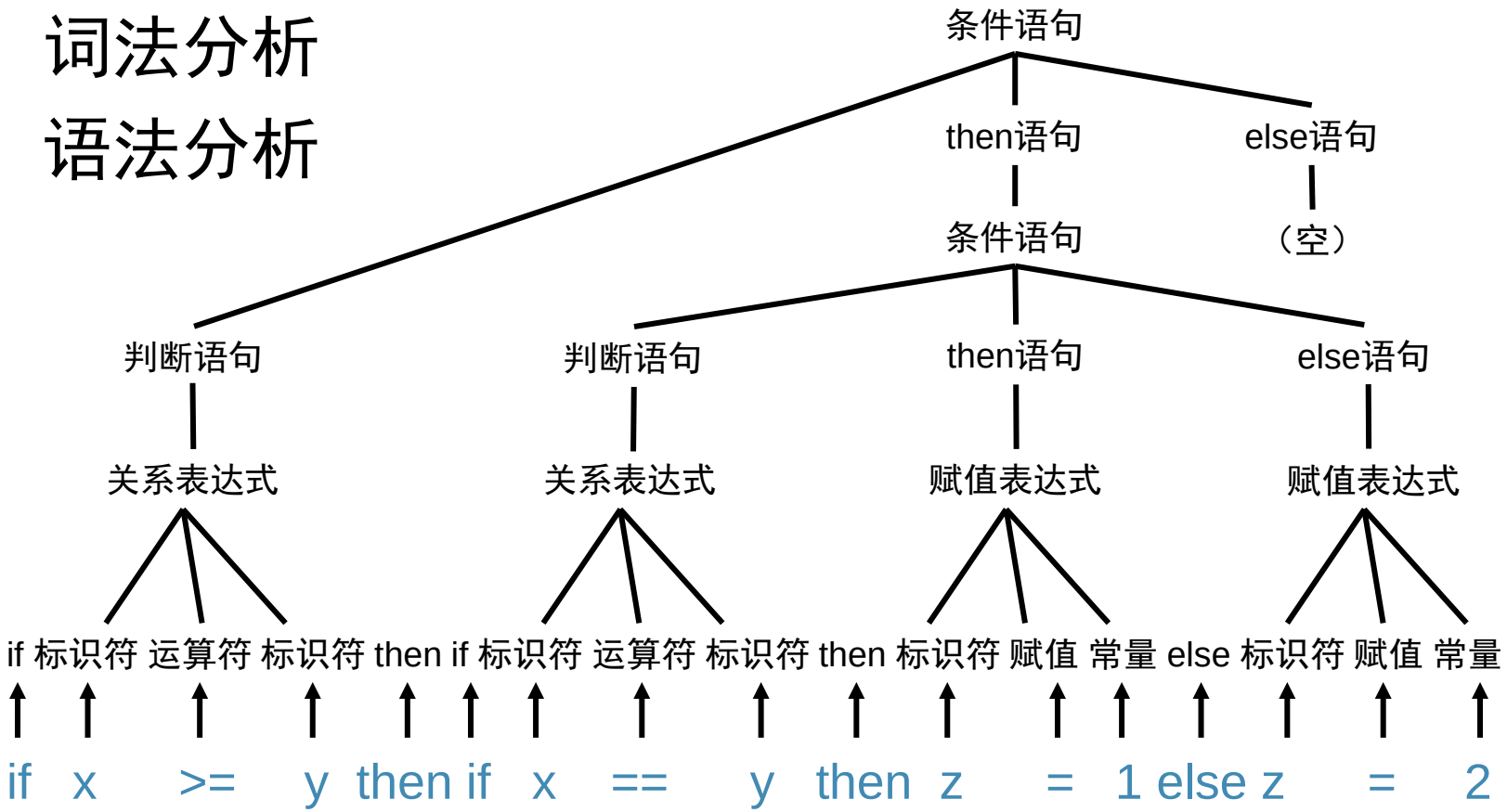


编译器的结构

- 你用编程语言写的程序是怎样被计算机编译的？

if x>=y then if x==y then z = 1 else z = 2

- 词法分析
- 语法分析

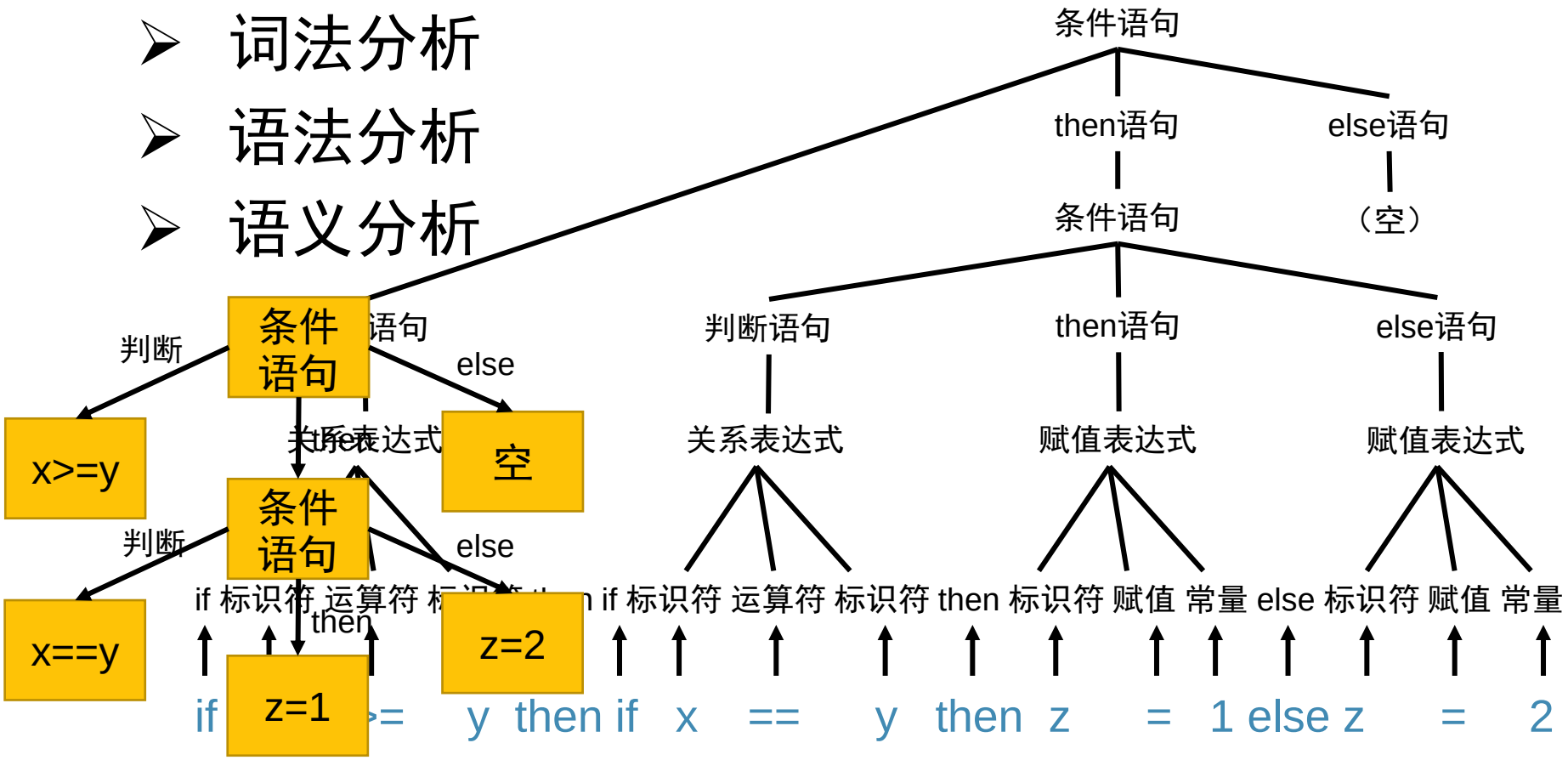


编译器的结构

- 你用编程语言写的程序是怎样被计算机编译的？

if x>=y then if x==y then z = 1 else z = 2

- 词法分析
- 语法分析
- 语义分析



编译器的结构

自然语言处理

- 语义分析对计算机来说是一项较为困难的任务

  Jack said Jerry left his assignment at home.

- his指谁？此处出现了二义性，通常需要结合上下文（实际语境）来判断

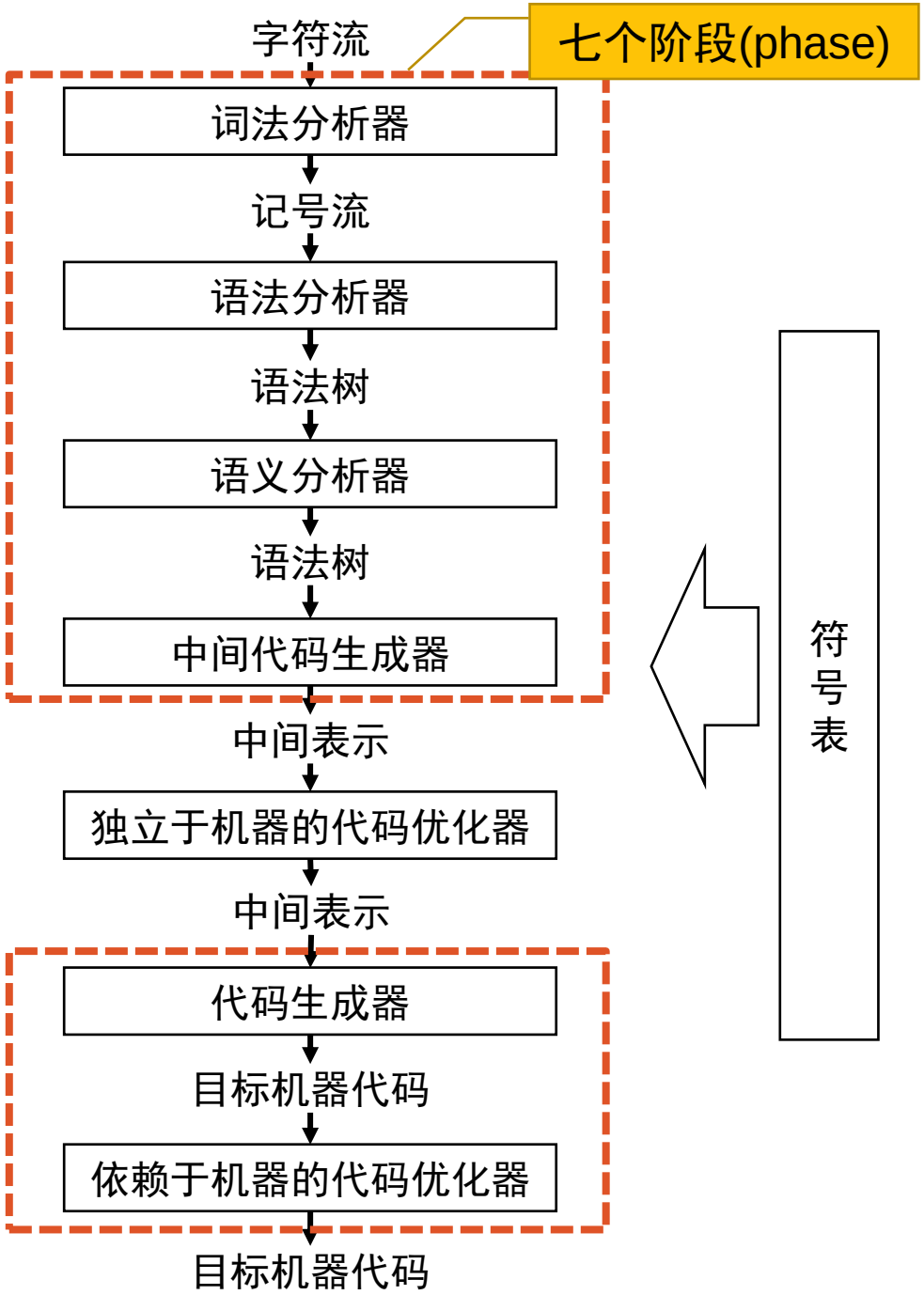
  if x>=y then if x==y then z = 1 else z = 2

- else到底是跟着哪一个then？
- 设计程序语言时，要合理制定规则，避免二义性

编译器的结构

分析部分
也叫前端(front end)
与源语言相关

综合部分
也叫后端(back end)
与目标语言相关



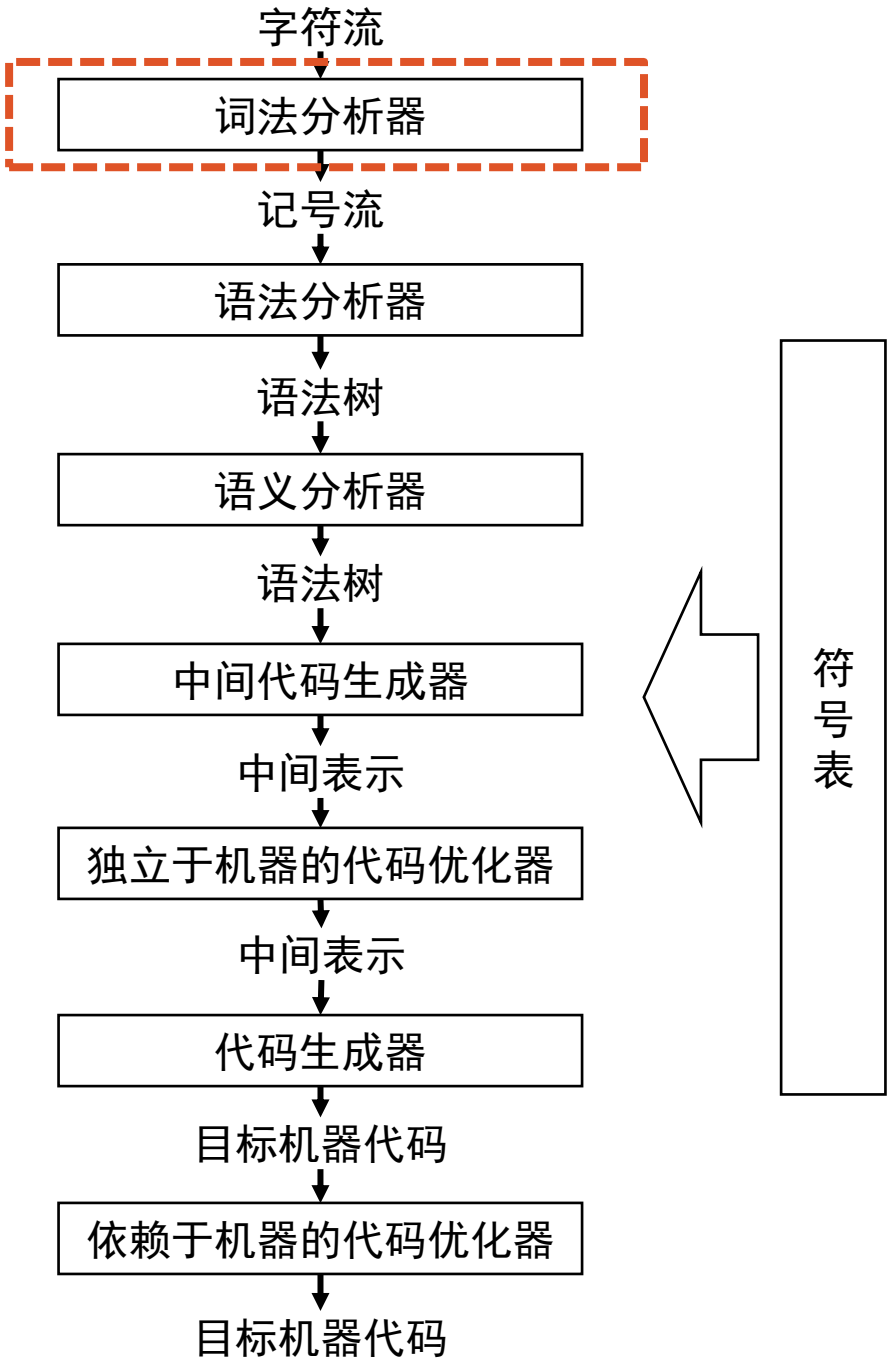
课堂练习

- 1、计算机中的编译就是将（高级语言）翻译成（汇编语言）或机器语言的过程。

- 2、下列哪个不是分析部分（前端）的阶段？ CF
  - A. 语义分析　　B. 词法分析
  - C. 依赖于机器的代码优化　　D. 语法分析
  - E. 中间代码生成器　　F. 预处理

词法分析

- 以下列程序为例：
- position = initial + rate * 60
- 假设变量的类型都是浮点数



词法分析

- 词法分析的主要任务
  - 从左向右逐行扫描源程序的字符，识别出各个单词，确定单词的类型。
  - 将识别出的单词转换成统一的机内表示——词法记号(token)
  - token: <记号名, 属性值>

	单词类型	类别	记号名
1	关键字	program、if、else、then、…	一词一名
2	标识符	变量名、数组名、记录名、过程名、…	多词一名
3	常量	整型、浮点型、字符型、布尔型、…	一型一名
4	运算符	算术（+ - * / ++ --） 关系（> < == != >= <=） 逻辑（& | ~）	一词一名 或 一型一名
5	界限符	; ( ) = { } …	一词一名

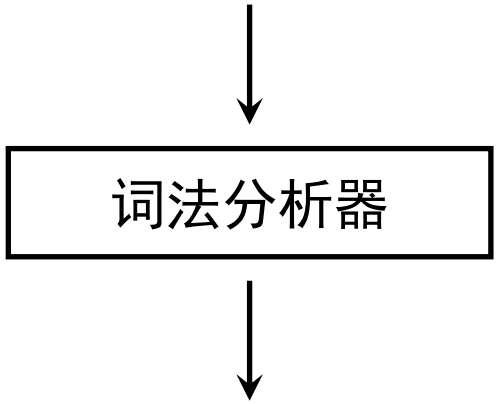
例：词法分析后得到的token序列

- 输入：**position = initial + rate * 60**
- 输出：

1	**position**	**< id,**	**1 >**
2	**=**	**< assign,**	**- >**
3	**initial**	**< id,**	**2 >**
4	**+**	**< +,**	**- >**
5	**rate**	**< id,**	**3 >**
6	*****	**< *,**	**- >**
7	**60**	**< int,**	**60 >**

词法分析后得到的token序列

position = initial + rate * 60 ← 字符流



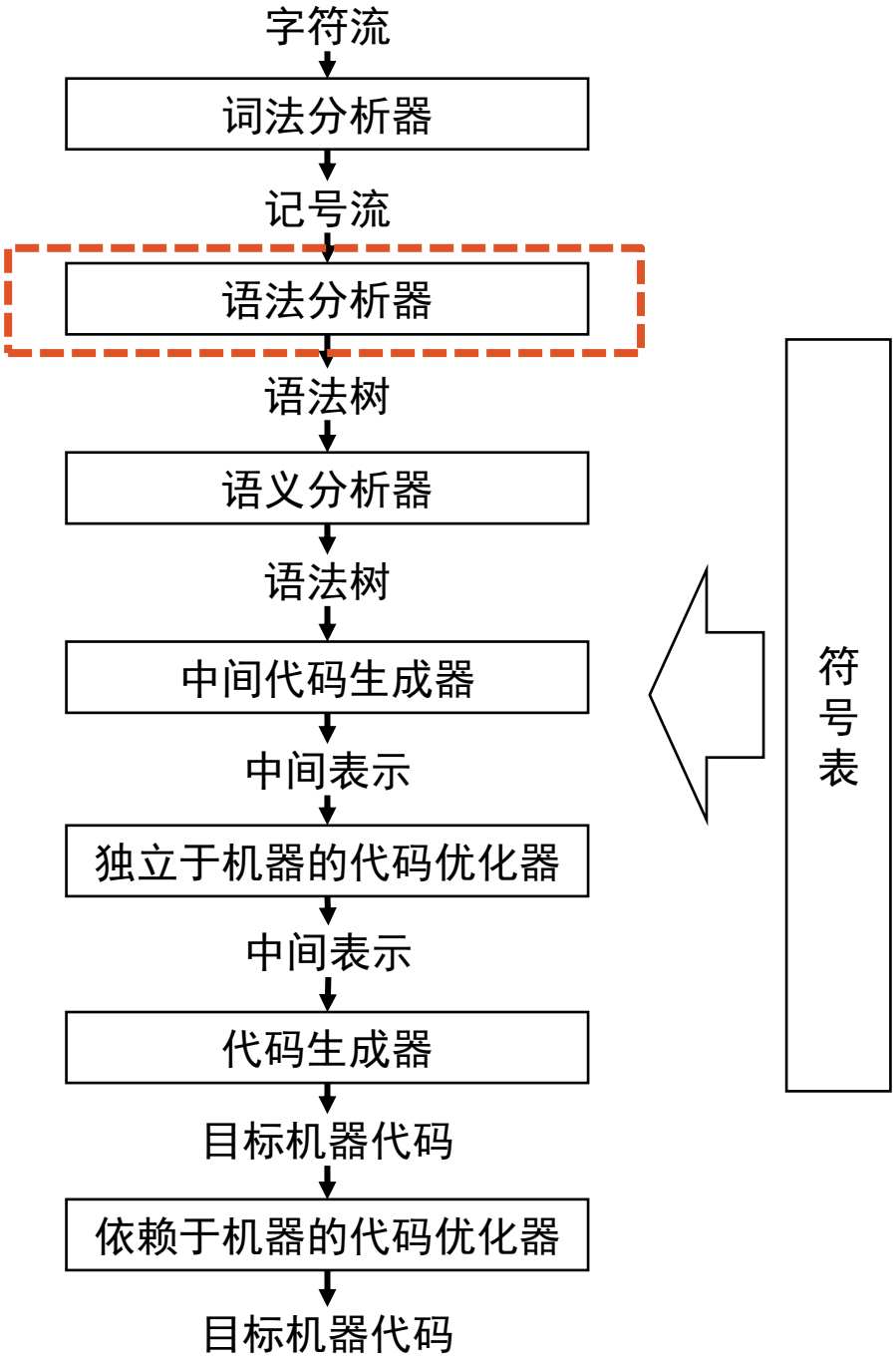
⟨id, 1⟩ ⟨=⟩ ⟨id, 2⟩ ⟨+⟩ ⟨id, 3⟩ ⟨*⟩ ⟨60⟩ ← 记号流

符 号 表

1	**position**	. . .
2	**initial**	. . .
3	**rate**	. . .

语法分析

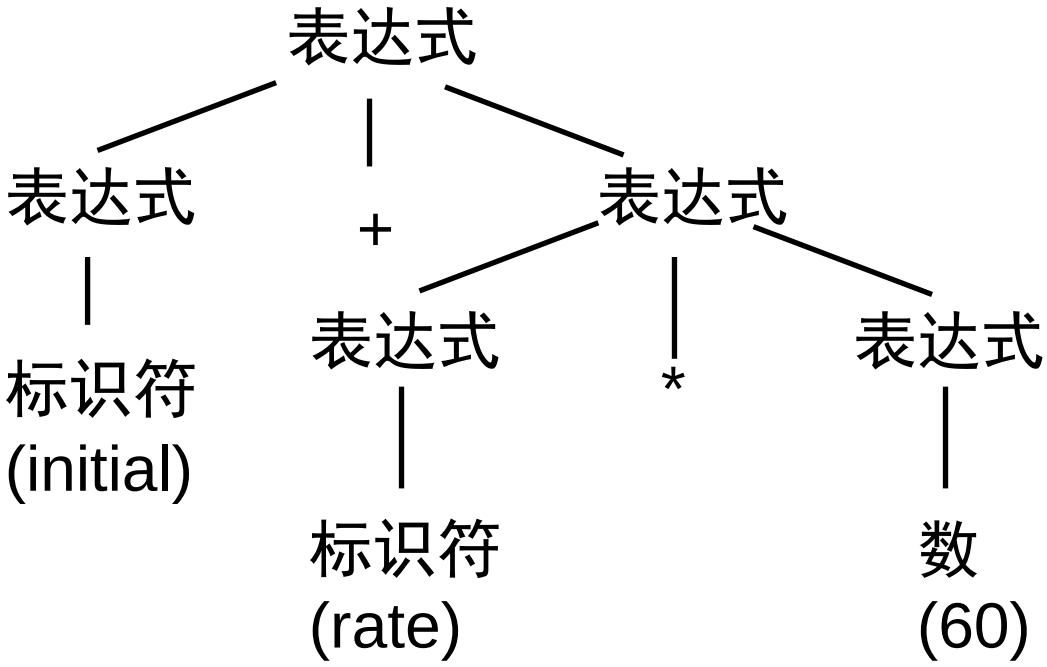
- 语法分析器(parser)从词法分析器输出的token序列中识别出各类短语，并构造语法树(syntax tree)
- 语法树描述了句子的语法结构，内部节点表示运算，子节点表示运算的运算对象



例：赋值语句的分析树和语法树

- position = initial + rate * 60
- ⟨id, 1⟩ ⟨=⟩ ⟨id, 2⟩ ⟨+⟩ ⟨id, 3⟩ ⟨*⟩ ⟨60⟩

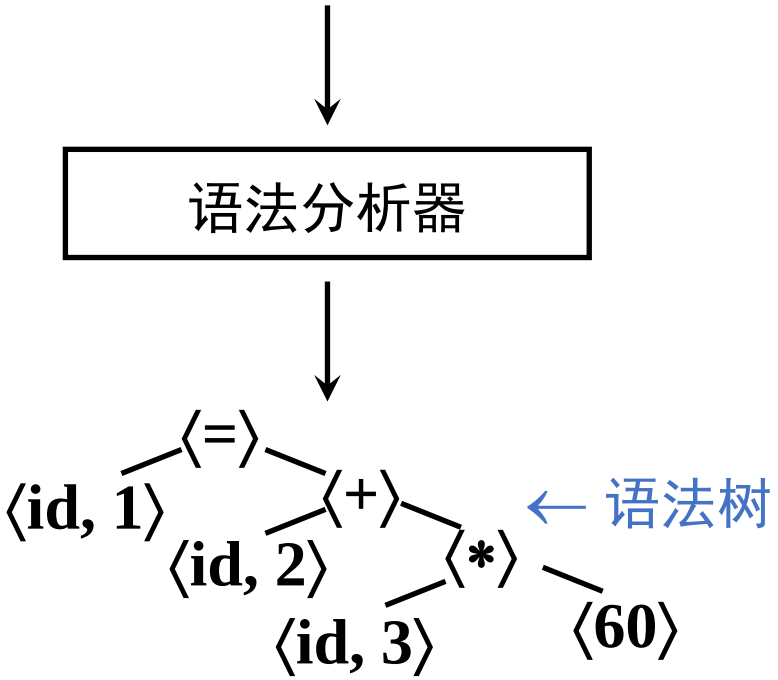
- 任何一个标识符都是表达式
- 任何一个数都是表达式
- 如果e1和e2都是表达式，那么
  e1 + e2
  e1 * e2
  (e1)
  也都是表达式



initial + rate * 60的分析树

例：赋值语句的分析树和语法树

⟨id, 1⟩ ⟨=⟩ ⟨id, 2⟩ ⟨+⟩ ⟨id, 3⟩ ⟨*⟩ ⟨60⟩ ← 记号流



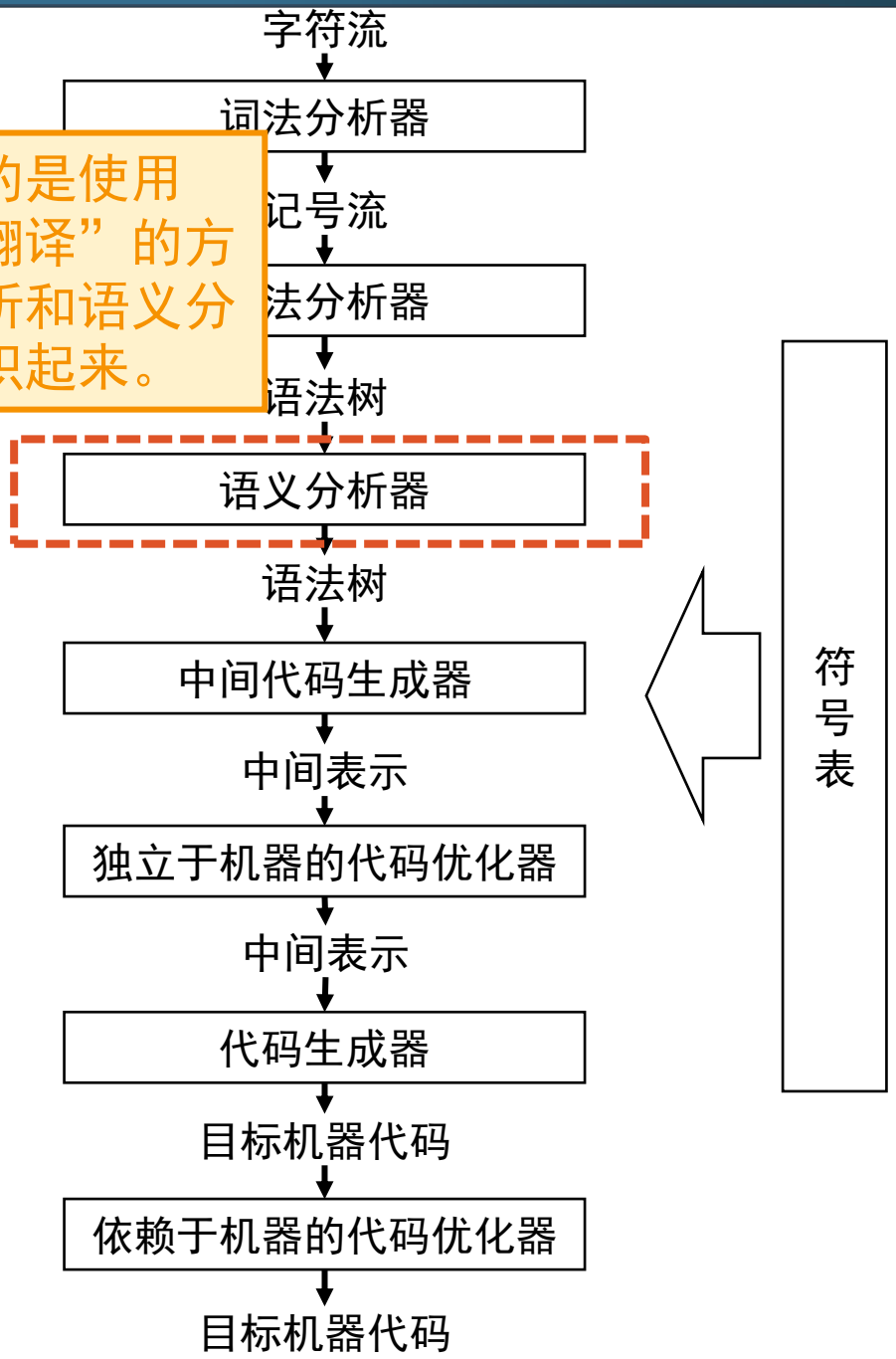
符 号 表

1	position	. . .
2	initial	. . .
3	rate	. . .

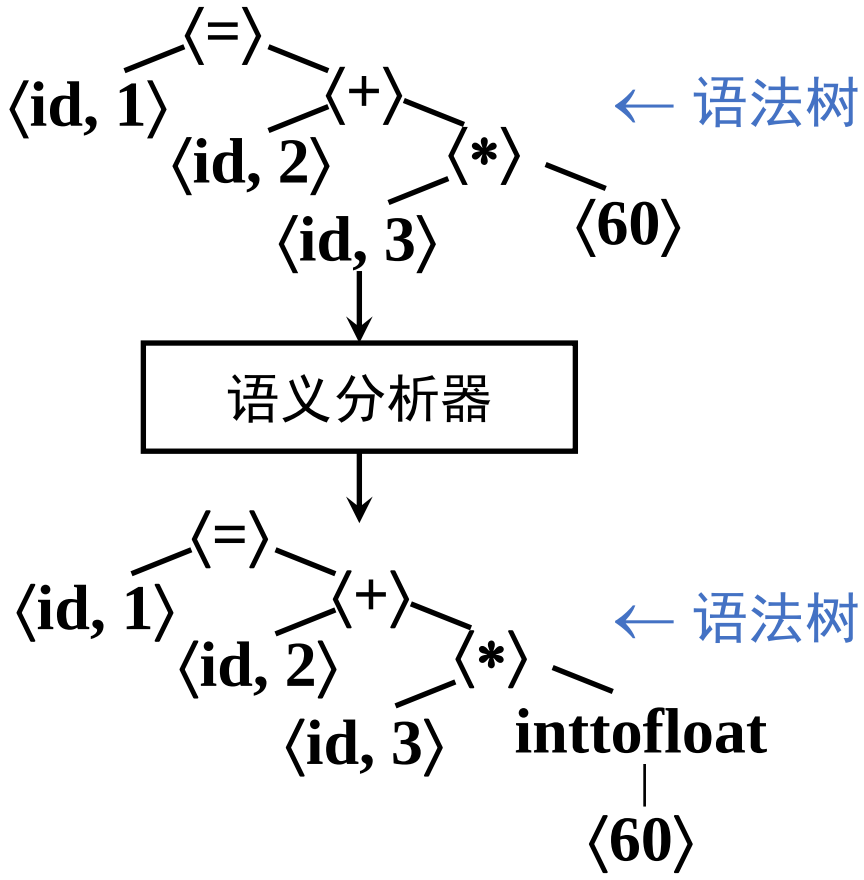
语义分析

- 收集标识符的属性信息
- 类型检查
  - 数组下标不是整数
  - 操作符与操作数之间的类型不匹配
  - 运算分量类型不匹配
  - 过程调用的参数类型或数目不匹配
  - 函数返回类型有误
  - ……
- 语义一致性检查
  - 变量或过程未经声明就使用
  - 变量或过程名重复声明
  - 对非数组变量使用数组访问操作符
  - 对非过程名使用过程调用操作符
  - ……
- 隐式类型转换（coercion）

目前较流行的是使用“语法制导翻译”的方法将语法分析和语义分析有机地组织起来。



语义分析

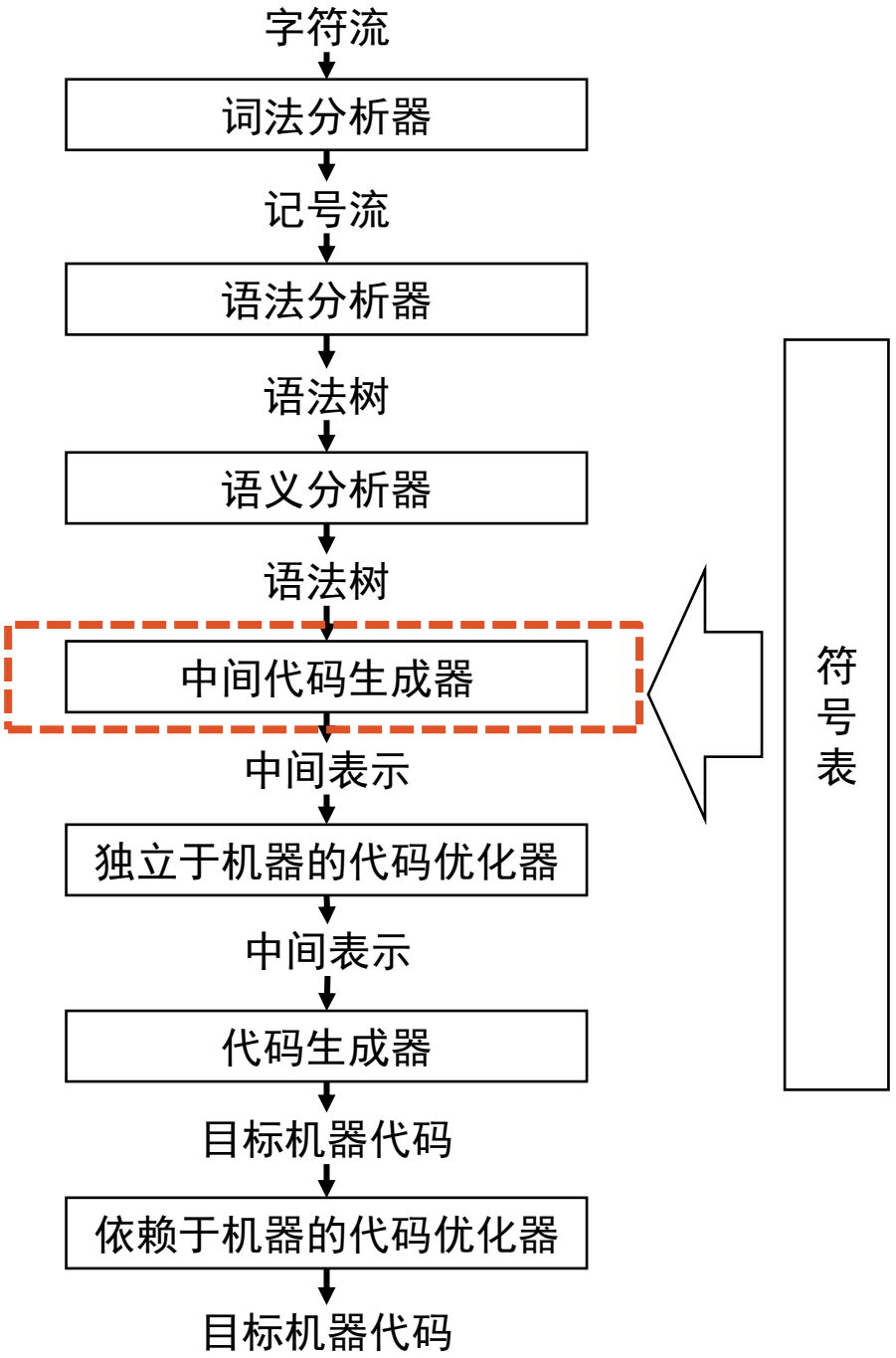


符 号 表

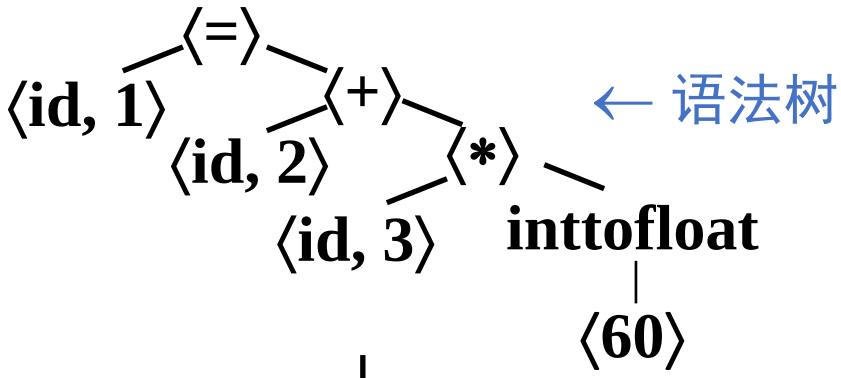
1	position	. . .
2	initial	. . .
3	rate	. . .

中间代码生成

- 中间代码有多种表示形式：
  - 后缀表示
  - 图形表示
  - 三地址代码：每个指令的右部最多有三个操作数
- 中间代码必须具备的特点：
  - 易于产生
  - 易于翻译成目标程序



中间代码生成



← 语法树

中间代码生成器

```
t1 = inttofloat(60)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3
```

← 三地址中间代码

符 号 表

1	position	. . .
2	initial	. . .
3	rate	. . .

特点：
指令确定了运算次序
需产生临时变量保存结果
不一定有三个运算对象

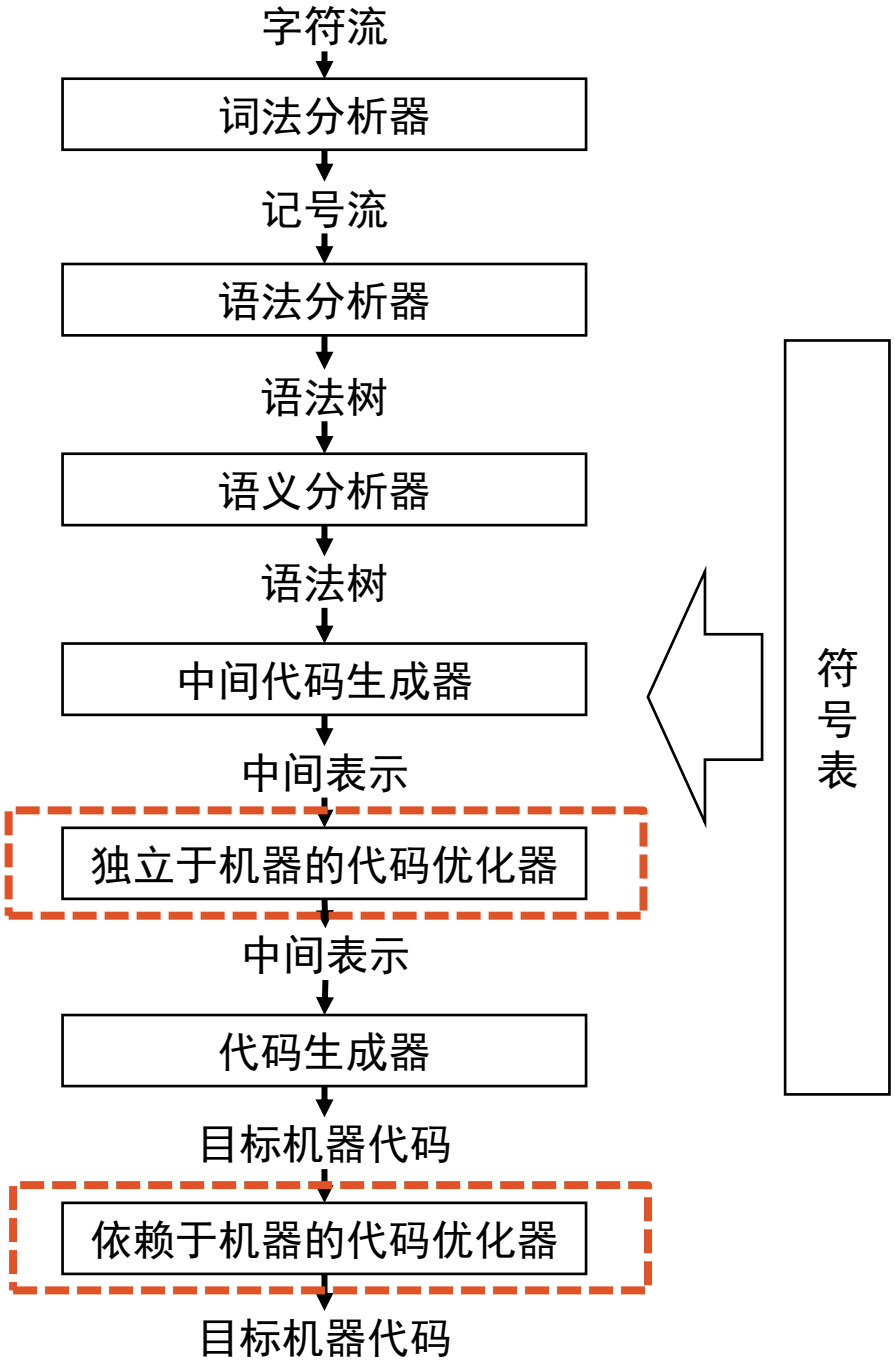
代码优化*

该章节不作为授课内容，可自行学习

- 为改进代码所进行的等价程序变换，产生更好的目标代码
- “更好”指的是：运行更快、或代码更短、或能耗更低，或多者兼顾
- 例：

```
t1 = inttofloat(60)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3
```

```
t1 = id3 * 60.0
id1 = id2 + t1
```



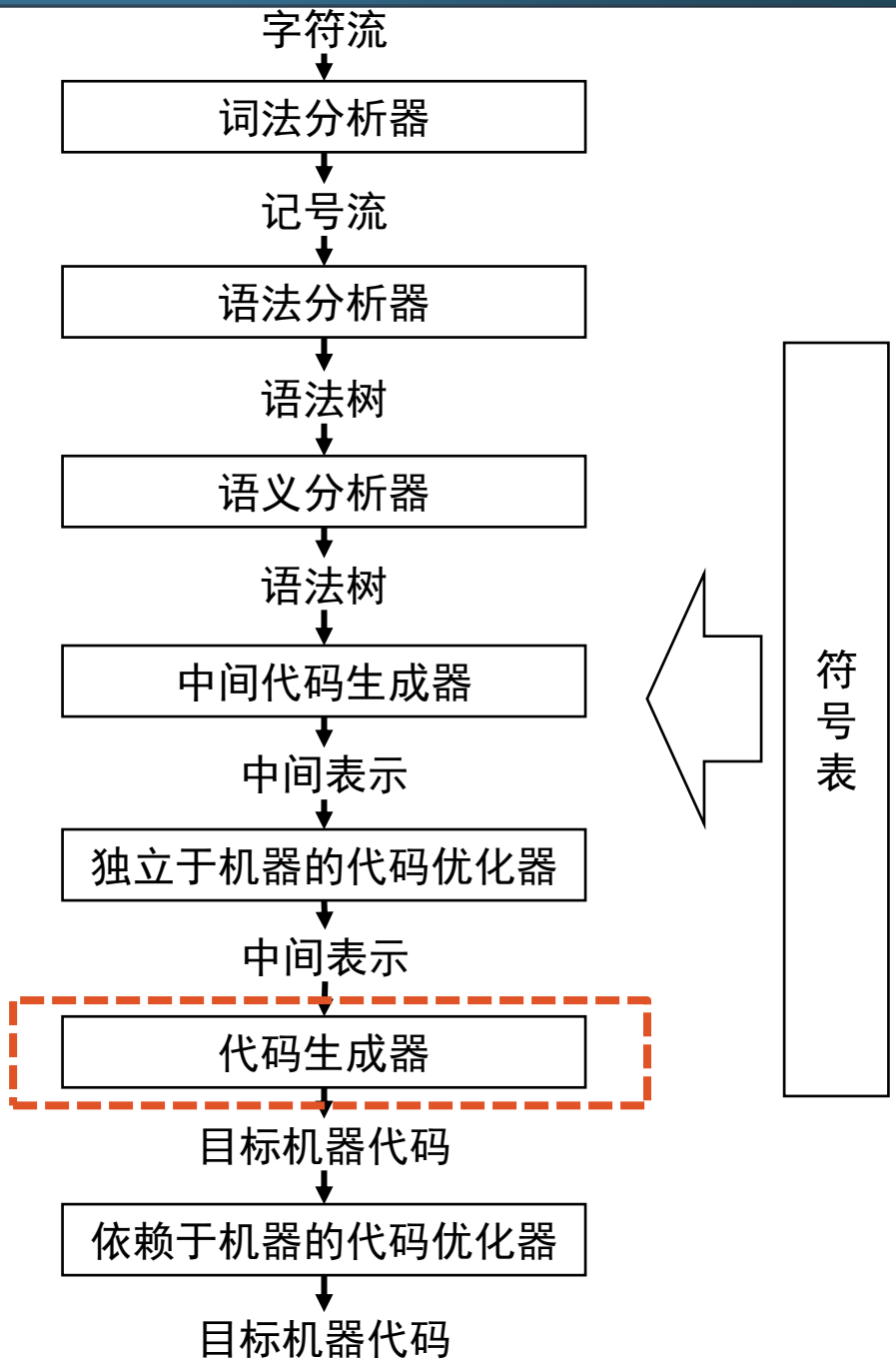
代码生成*

该章节不作为授课内容，可自行学习

- 由中间代码生成目标代码（汇编代码或机器代码）
- 代码生成阶段的工作：
  - 为变量选择寄存器或者内存单元
  - 把中间代码翻译成等价的机器指令序列
- 此阶段的关键问题是：寄存器分配

```
t1 = id3 * 60.0
id1 = id2 + t1
```

```
MOVF id3,  R2
MULF #60.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1
```



符号表管理

- 编译器的一项重要工作是记录并收集标识符及其各种属性，以提供标识符存储分配、类型、作用域等信息。
- 符号表为每个标识符保存一个记录的数据结构，记录的域是标识符的各个属性。
- 该数据结构应允许编译器迅速找到一个名字的记录并读取和存储数据。

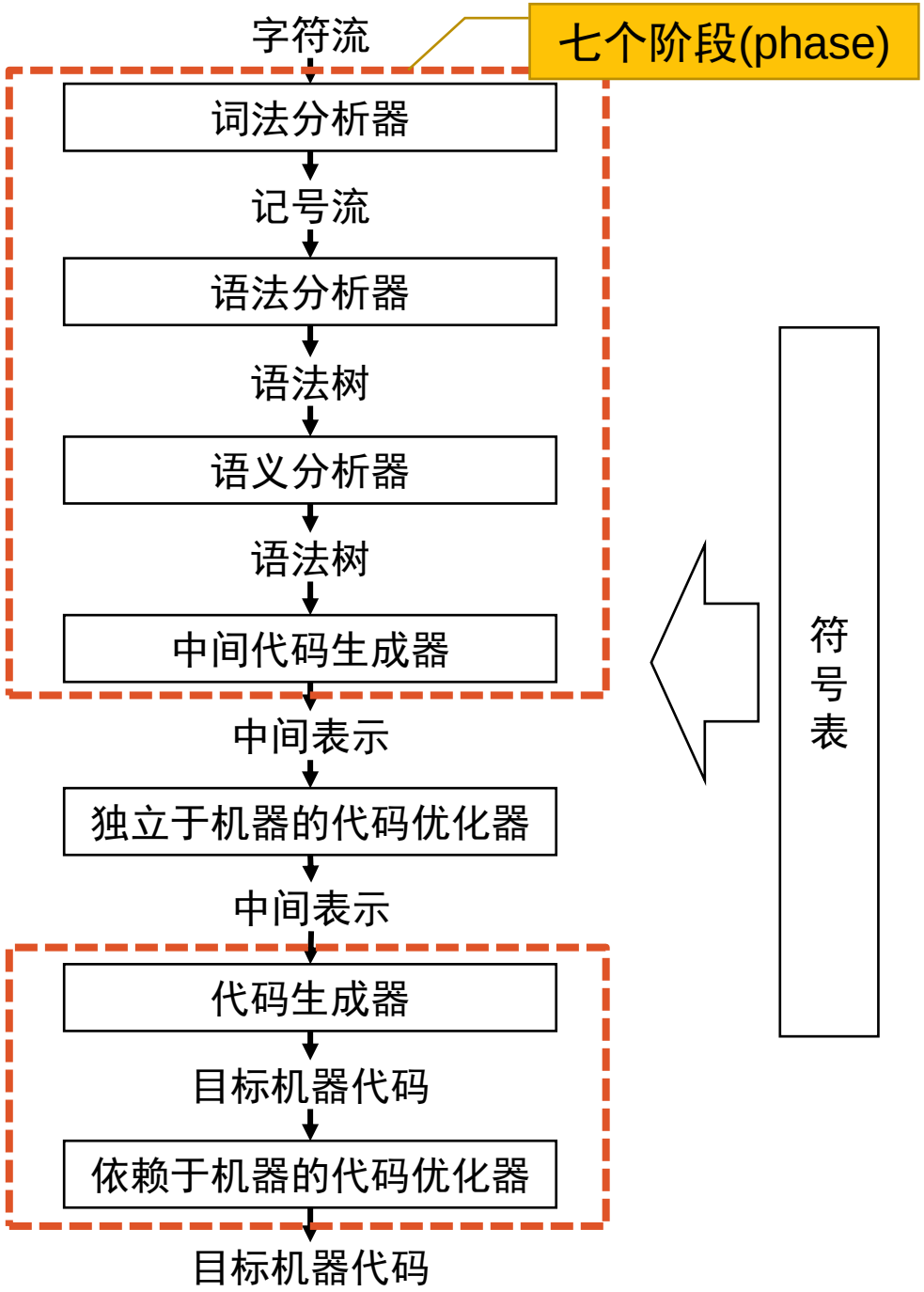
符 号 表

1	**position**	. . .
2	**initial**	. . .
3	**rate**	. . .

阶段的分组

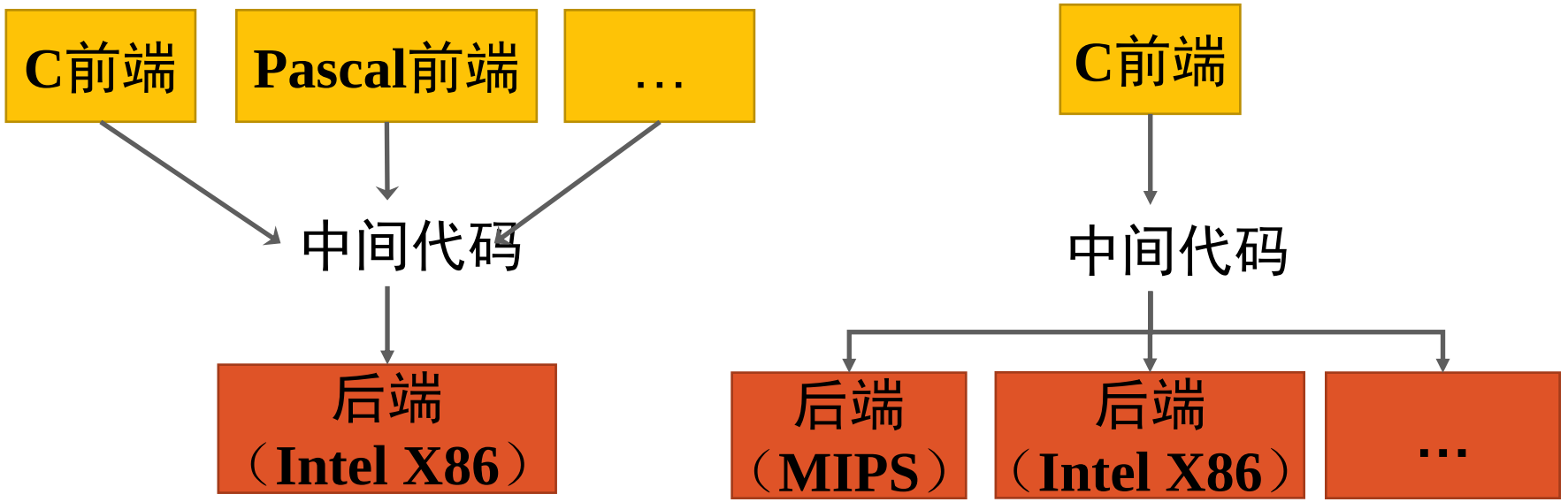
分析部分
也叫前端(front end)
与源语言相关

综合部分
也叫后端(back end)
与目标语言相关



阶段的分组

- 编译器划分前端后端有利于移植及简化编译器的设计



阶段的分组

- 在编译器实现时，几个阶段的活动可能被组合起来形成一“遍”（pass）
- 每一“遍”都会读一个输入文件（例如源程序或源程序的中间表示），并写一个输出文件。
- 例：前端的四个阶段：词法分析、语法分析、语义分析、中间代码生成可以组合形成一“遍”。
- 例：代码优化可以进行很多“遍”。

编译器 vs 解释器

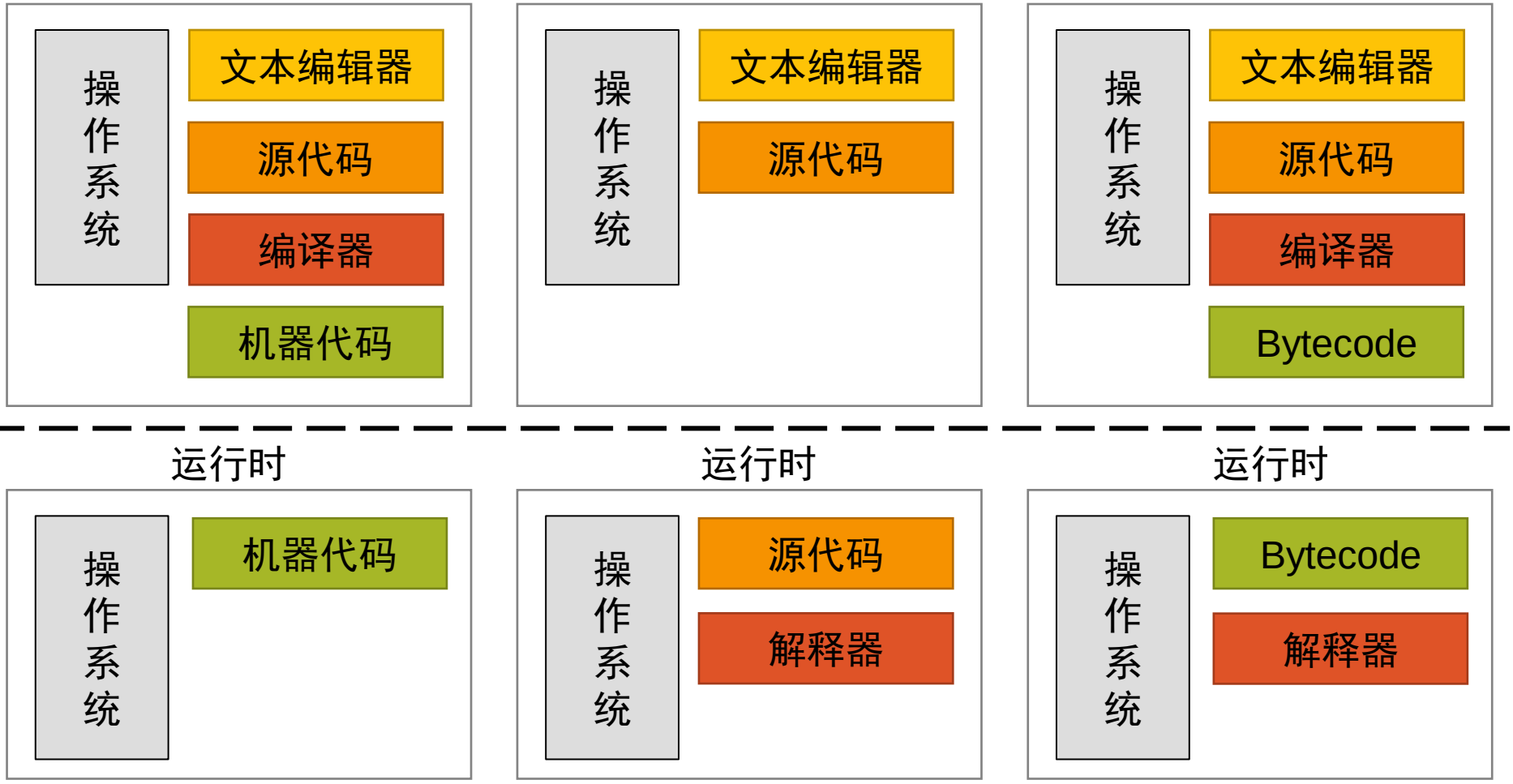
- 翻译器（translator）：
  - 能够将一种语言编写的程序翻译成语义等价的另外一种语言程序的软件系统。
- 编译器（compiler）：
  - 翻译器的一种，通常其目标语言比源语言低级（低级语言：编程语言越接近机器语言，我们就说这种语言越低级。）
- 解释器（interpreter）：
  - 翻译器的一种，但不会生成目标程序，而是直接执行源程序所指定的运算。

编译器 vs 解释器

C
C++
Objective-C
Fortran

BASIC
Python
JavaScript
Ruby

Java
C#



编译器技术的应用*

- 1. 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

编译器技术的应用*

- 1. 高级语言的实现
  - 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

编译器技术的应用*

- 2. 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

编译器技术的应用*

- 3. 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

编译器技术的应用*

- 4. 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

编译器技术的应用*

- 5. 提高软件开发效率的工具
  - 源于编译器中代码优化技术的程序分析一直在改进软件开发效率
  - 类型检查
    - 类型检查是一种捕捉程序中前后不一致的成熟而有效的技术
  - 边界检查
    - 数据流分析技术可用来定位缓冲区溢出
  - 内存管理
    - 自动的内存管理删除内存泄漏等内存管理错误

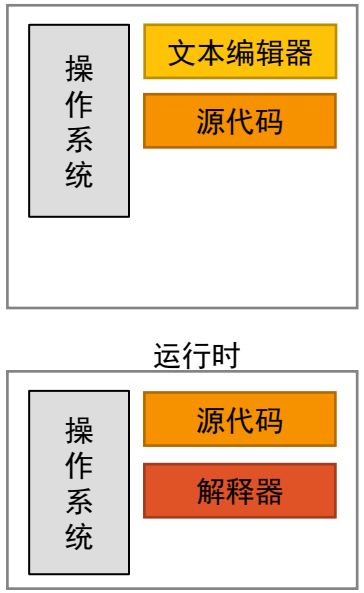
课堂练习

- 编译器的（ A ）阶段的功能是将源程序字符流变为记号流。
  - A.词法分析　　B.语法分析
  - C.语义分析　　D.中间代码生成
- 编译器的阶段顺序是（ FECGDAB ）
  - A.代码生成阶段　　B.依赖于机器的优化阶段
  - C.语义分析阶段　　D.独立于机器的代码优化阶段
  - E.语法分析阶段　　F.词法分析阶段
  - G. 中间代码生成阶段

课堂练习

- 判断：C编译器在中间代码生成阶段生成的是汇编代码。错误
- 下列哪幅示意图是解释器的工作原理？ A

A

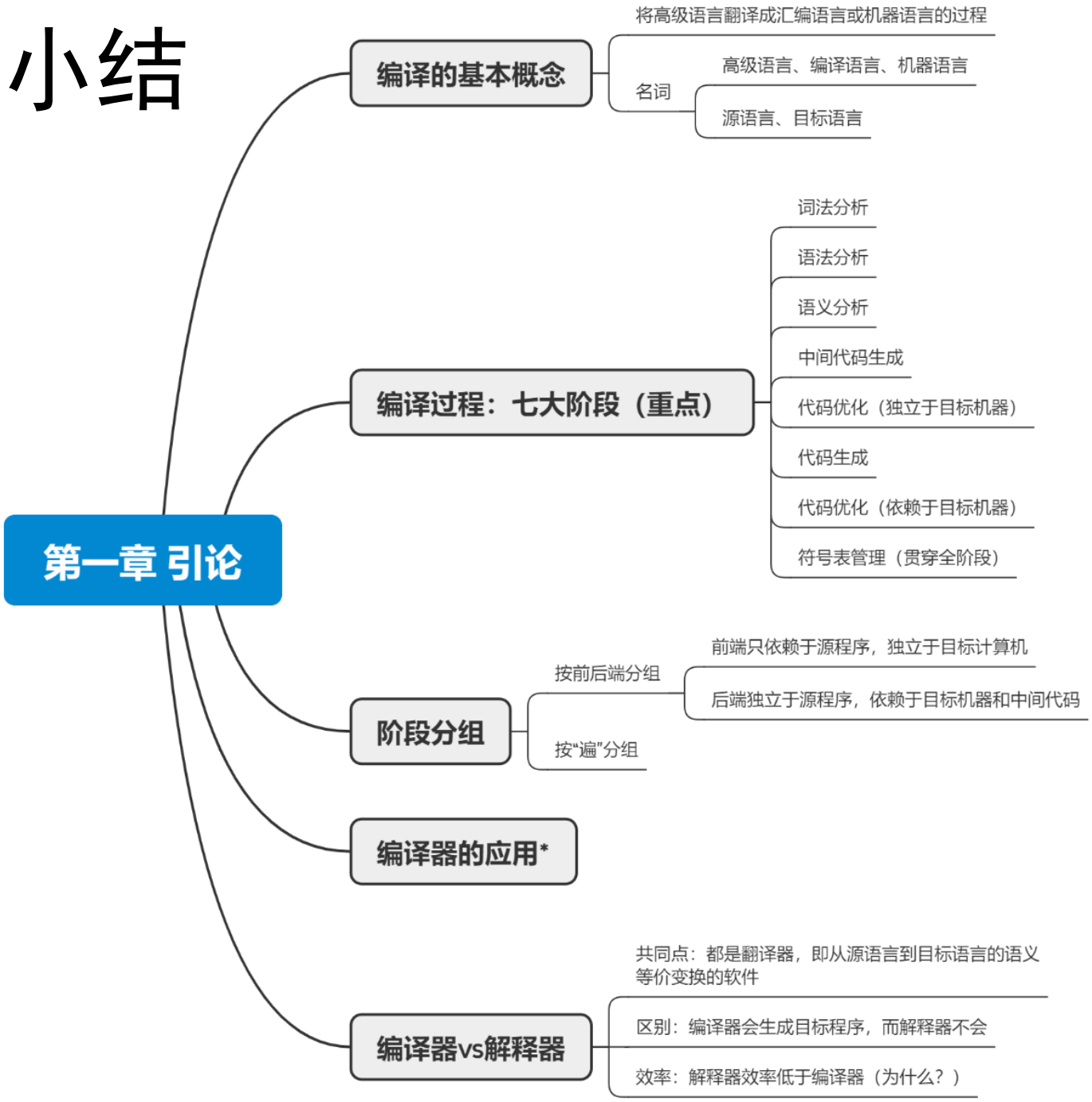


B

课堂练习

- 根据上图，为什么解释器的效率比编译器要低？
  - 编译器生成目标程序，运行时直接执行机器指令。
  - 解释器不会生成目标程序，而是将源代码直接在目标机器上运行。执行时，解释器读取一句源代码之后，先进行词法分析和语法分析，再将源代码转换为解释器能够执行的中间代码（字节码），最后，由解释器将中间代码解释为可执行的机器指令。
  - 由于解释器运行时步骤更加繁琐，所以解释型语言的执行效率会低一些。

本章小结


第一章 引论
编译的基本概念
将高级语言翻译成汇编语言或机器语言的过程
名词
高级语言、编译语言、机器语言
源语言、目标语言
编译过程：七大阶段（重点）
词法分析
语法分析
语义分析
中间代码生成
代码优化（独立于目标机器）
代码生成
代码优化（依赖于目标机器）
符号表管理（贯穿全阶段）
阶段分组
按前后端分组
前端只依赖于源程序，独立于目标计算机
后端独立于源程序，依赖于目标机器和中间代码
按"遍"分组
编译器的应用*
编译器vs解释器
共同点：都是翻译器，即从源语言到目标语言的语义等价变换的软件
区别：编译器会生成目标程序，而解释器不会
效率：解释器效率低于编译器（为什么？）

课后任务

- 请大家学习BB平台的使用方式；请大家自行在BB平台上组建小组（贯穿整个课程），请商量好后再加入，一旦加入无法退出，只能老师手工修改。
- 以小组为单位，任意选择一种主流编译器（如GCC），了解它的发展历史、特点等，与组内成员展开讨论，并在BB平台的小组内以此为主题**创建一篇新博客。截止时间为下周上课前。**