山东大学
SHANDONG UNIVERSITY

编译原理

第一章 引论

授课教师：余仲星
手机：15866821709（微信同号）
邮箱：zhongxing.yu@sdu.edu.cn

个人基本情况

- 教育经历
  - 本科---北航 (2006-2010)
  - 博士---北航 (2010-2016)

- 工作经历
  - 博士后-法国国立计算机及自动化研究院 (INRIA，2016-2018)
  - 博士后-瑞典皇家理工学院（KTH，2018-2020）
  - 教授-山东大学 (2020-至今)

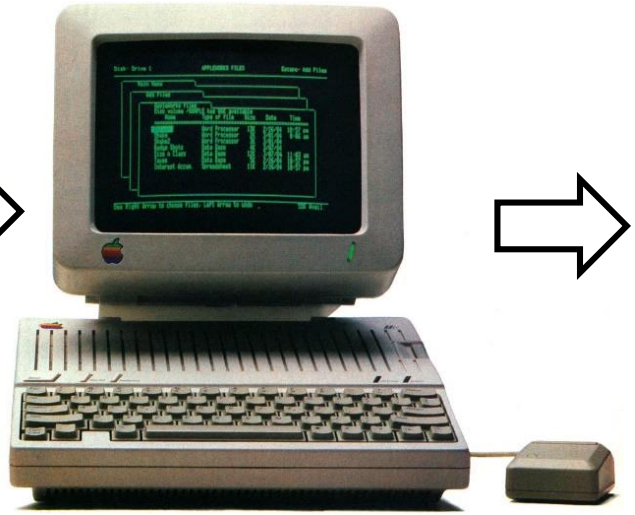
个人基本情况

- 研究领域
  - 编程语言 (Programming Languages)
  - 软件工程 (Software Engineering)
  - 形式化方法 (Formal Methods)

- 研究方向
  - 程序合成 (Program Synthesis)
  - 程序自动修复 (Automatic Program Repair)
  - 程序分析 (Program Analysis)

程序合成

I want a program which can sort a list of numbers



```
int[N] sort(int N, int[N] input){
    int[N] output=input;
    int[N] done = 0;
    int k=0;
    for(int i=0; i<N; ++i){
        for(int j=i+1; j<N; ++j){
            if( output[j]< output[i]){
                int tmp = output[j];
                output[j] = output[i];
                output[i] = tmp;
            }
        }
    }
    return output;
}
```

需求

自动搜索

基于逻辑推理

基于数据学习

程序代码

- 已成功应用于程序最优化、在线教育、脚本编写等
- Google、Facebook、Microsoft等均成立大型程序合成研究组

编译原理

- 编译原理是计算机专业最难的科目之一，是一门理论性和实践性都很强的学科，本课程将理论和实践环节作为考核重点。
- 考核方式：考试卷面50% + 平时成绩50%
- 平时：作业 + 实验 + 考勤
- 教材和参考书

- Alfred V.Aho, Monica S. Lam, Ravi Sethi, Jeffey D.Uiiman. Compilers: Principles, Techniques, and Tools, 人民邮电出版社第二版, ISBN: 978-7-11517265-5.
- 陈火旺等. 编译原理, 国防工业出版社第3版, ISBN: 978-7-118-02207-0.
- 蒋宗礼等. 形式语言与自动机理论, 清华大学出版社第3版, ISBN: 978-7-302-31802-6.

第一章 引论

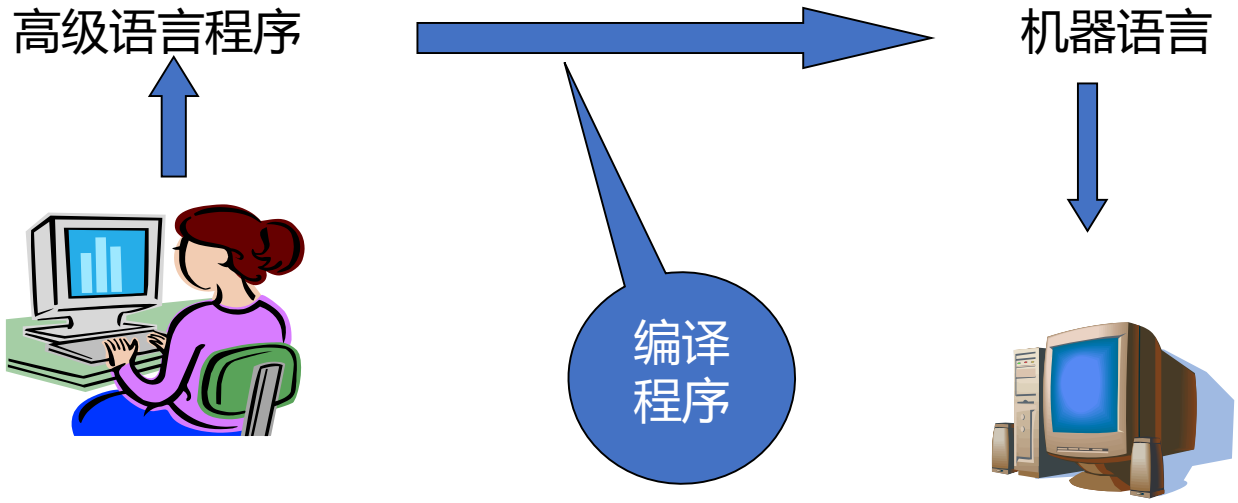
- 1.1 什么是编译程序
- 1.2 编译过程概述
- 1.3 编译程序的结构
  - 1.3.1 编译程序框架
  - 1.3.2 表格和表格管理
  - 1.3.3 出错处理
  - 1.3.4 遍
  - 1.3.5 编译前端和后端
- 1.4 编译程序与程序设计环境
- 1.5 编译程序的生成

第一章 引论

- 1.1 什么是编译程序
- 1.2 编译过程概述
- 1.3 编译程序的结构
  - 1.3.1 编译程序框架
  - 1.3.2 表格和表格管理
  - 1.3.3 出错处理
  - 1.3.4 遍
  - 1.3.5 编译前端和后端
- 1.4 编译程序与程序设计环境
- 1.5 编译程序的生成

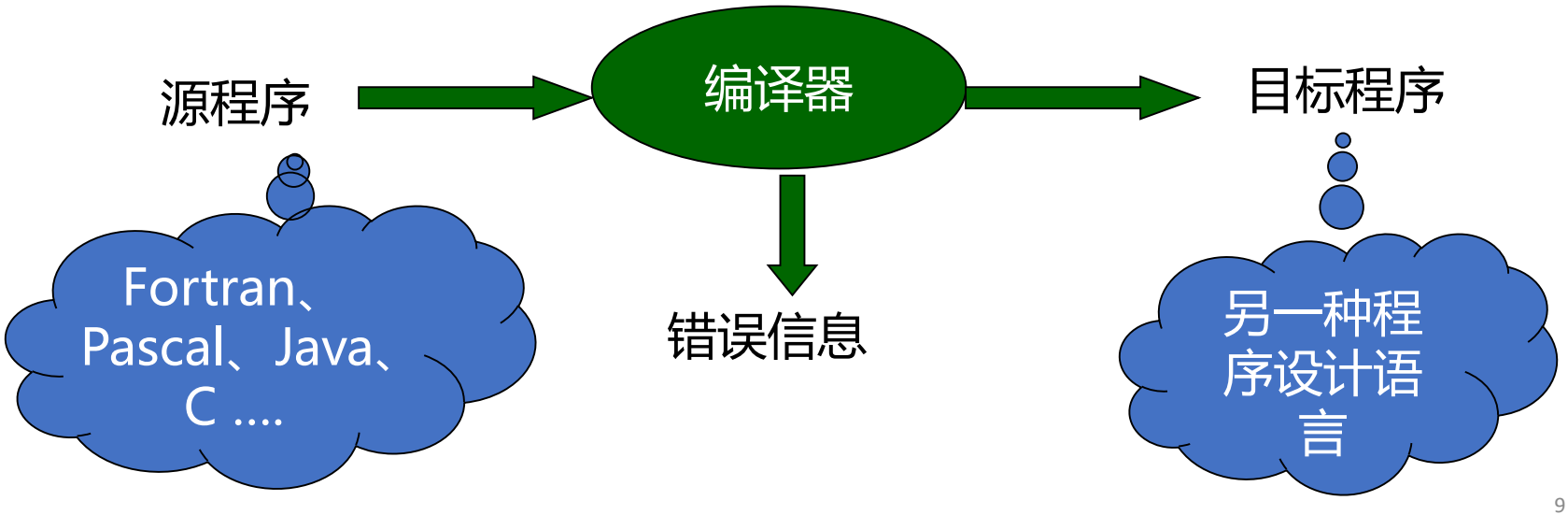
1.1 什么是编译程序

- 计算机包括硬件和软件两大部分；
  - 裸机从某个固定的地址开始载入“程序”，根据“程序逻辑”执行逻辑操作



1.1 什么是编译程序

- 翻译程序：把一种语言程序（源语言）转换成另一种语言程序（目标语言）
- 编译程序：源语言为高级语言，目标语言为低级语言的翻译程序。
- 解释程序：以源语言作为输入，但不产生目标程序，而是边翻译边执行源程序本身。



1.1 什么是编译程序

- 编译原理的研究对象
  - ✓ 形式语言：指程序设计语言，是用精确的数学或机器可处理的公式定义的语言。
  - × 自然语言：就是人类讲的语言，比如汉语、英语和法语。
- 语言的分类
  - ➢ 机器语言：是机器能直接识别的程序语言或指令代码，每一操作码在计算机内部都有相应的电路来完成它。
  - ➢ 汇编语言：将计算机指令用易于记忆的符号表示。
  - ➢ 高级语言：由表达各种不同意义的“关键词”和“表达式”，按一定的语义规则组成的程序。
  - ➢ MSIL和Java Bytecode：属于在虚拟机上运行的中间语言，作用类似于汇编语言，但结构要比汇编语言高级的多，需要虚拟机二次编译或者解释才能执行。

1.1 什么是编译程序

- 对机器的名称约定
  - 宿主机：运行编译程序的计算机。
  - 目标机：运行编译程序所生成的目标代码的计算机。
- 编译程序的进一步分类
  - 诊断编译程序(Diagnostic Compiler)：专门用于帮助程序开发和调试的编译程序。
  - 优化编译程序(Optimizing Compiler)：着重于提高目标代码效率的编译程序。
  - 交叉编译程序(Cross Compiler)：编译程序产生不同于其宿主机的机器代码。
  - 可变目标编译程序(Retargetable Compiler)：不需要重新编译程序中与机器无关的部分，就可以改变目标机。

第一章 引论

- 1.1 什么是编译程序
- 1.2 编译过程概述
- 1.3 编译程序的结构
  - 1.3.1 编译程序框架
  - 1.3.2 表格和表格管理
  - 1.3.3 出错处理
  - 1.3.4 遍
  - 1.3.5 编译前端和后端
- 1.4 编译程序与程序设计环境
- 1.5 编译程序的生成

1.2 编译过程概述

- 自然语言的翻译
  - ① 识别出句子中的一个个单词
  - ② 分析句子的语法结构
  - ③ 根据句子的含义进行初步翻译
  - ④ 对译文进行修饰
  - ⑤ 写出最后的译文

- 编译
  - ① 词法分析
  - ② 语法分析
  - ③ 语义分析和代码自动生成
  - ④ 代码优化
  - ⑤ 目标代码生成

1.2 编译过程概述

- （1）词法分析：输入源程序，对构成源程序的字符串进行扫描和分解，识别出一个个的单词（又称单词符号或简称符号）
  - 基本字，如begin, end, if ,while, for等
  - 标识符
  - 常数
  - 算符和界符，如标点符号、左右括号等
- 如Pascal语句：FOR I := 1 TO 100 DO
  - 基本字 FOR
  - 标识符 I
  - 赋值符号 :=
  - 整型常数 1
  - 基本字 TO
  - 整型常数 100
  - 基本字 DO

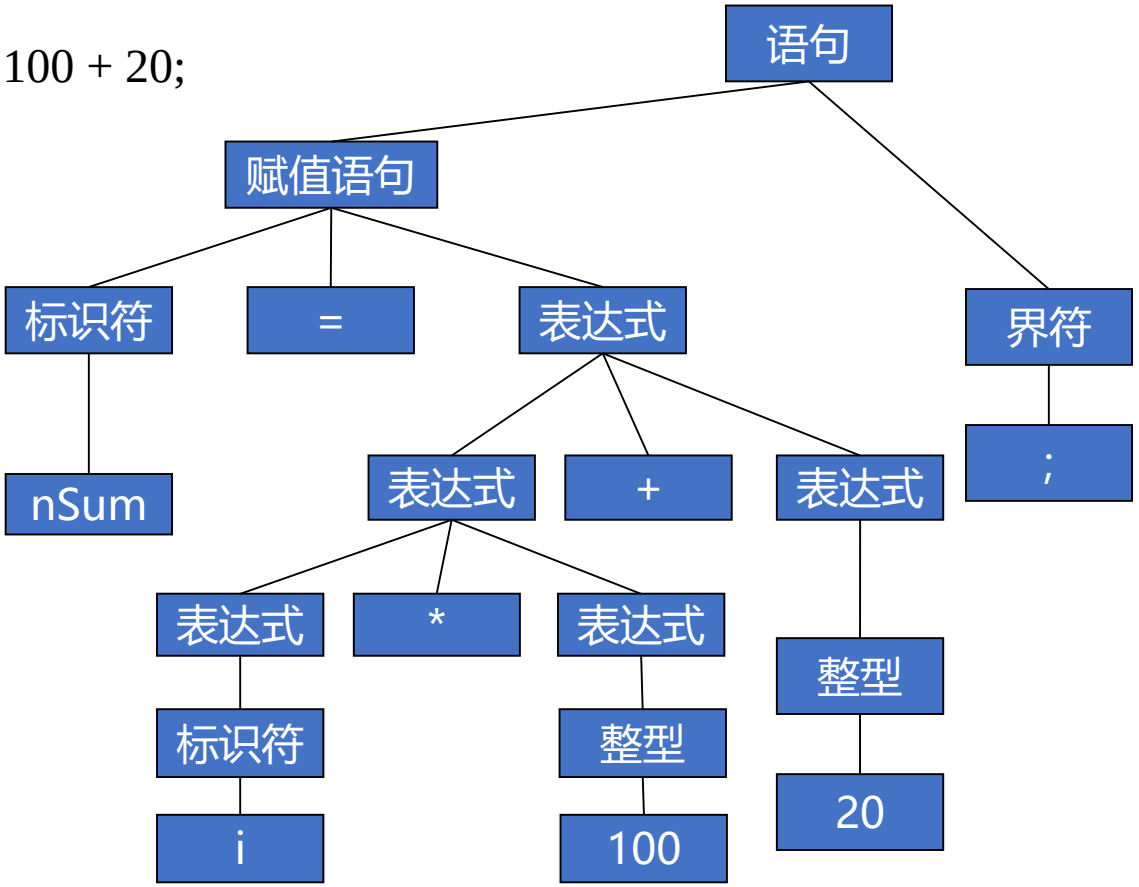
1.2 编译过程概述

- 如C++语句： for (int i = 0; i < 100; i++)
  - 基本字 for
  - 左括号 (
  - 基本字 int
  - 标识符 i
  - 赋值符号 =
  - 界符 ;
  - 标识符 i
  - 关系运算符 <
  - 界符 ;
  - 标识符 i
  - 自增算符 ++
  - 右括号)

1.2 编译过程概述

- （2）语法分析：在词法分析基础上，根据语言的语法规则，把单词符号串分解成各类语法单位，如短语、子句、语句、程序段、程序等，确定整个输入串是否构成语法上正确的“程序”。

  - 例：nSum = i * 100 + 20;



1.2 编译过程概述

- （3）语义分析与中间代码生成：对语法分析所识别出的各类语法范畴，分析其含义，并进行初步翻译，产生中间代码。
  - 首先对每种语法范畴进行静态语义检查，如变量是否定义、类型是否正确等
  - 如果语义正确，则进行中间代码的翻译。
- 中间代码：是一种含义明确、便于处理的记号系统，通常独立于具体的硬件但与指令形式有某种程度的接近，或者能够比较容易的转换为机器指令
  - 三元式
  - 间接三元式
  - 四元式：（算符，左操作数，右操作数，结果）
  - 逆波兰式
  - 树形表示

1.2 编译过程概述

- 中间代码：z := (x + 0.418) * y / w

① (+, x, 0.418, T_1)

② (*, T_1, y, T_2)

③ (/, T_2, w, z)

1.2 编译过程概述

- （4）优化：对前段产生的中间代码进行加工变换，以期在最后阶段能产生出更为高效（省时间和空间）的目标代码。
  - 公共子表达式的提取
  - 循环优化
  - 删除无用代码
  - 有时，并行化处理

1.2 编译过程概述

- 优化示例：

```
for k := 1 to 100 do
begin
        m := i + 10 * k
        n := j + 10 * k
end
```

- 注：
  - 实际代码不需要循环，只需要把10*k替换成10*100即可，但本课程介绍的优化理论尚无法做到这一步。

- 中间代码：
  - ① (:=, 1, —, k)
  - ② (j<, 100, k, ⑨)
  - ③ (*, 10, k, T1)
  - ④ (+, i, T1, m)
  - ⑤ (*, 10, k, T2)
  - ⑥ (+, j, T2, n)
  - ⑦ (+, k, 1, k)
  - ⑧ (j, —, —, ②)
  - ⑨ …

- 优化后：
  - ① (:=, i, —, m)
  - ② (:=, j, —, n)
  - ③ (:=, 1, —, k)
  - ④ (j<, 100, k, ⑨)
  - ⑤ (+, m, 10, m)
  - ⑥ (+, n, 10, n)
  - ⑦ (+, k, 1, k)
  - ⑧ (j, —, —, ④)
  - ⑨ …

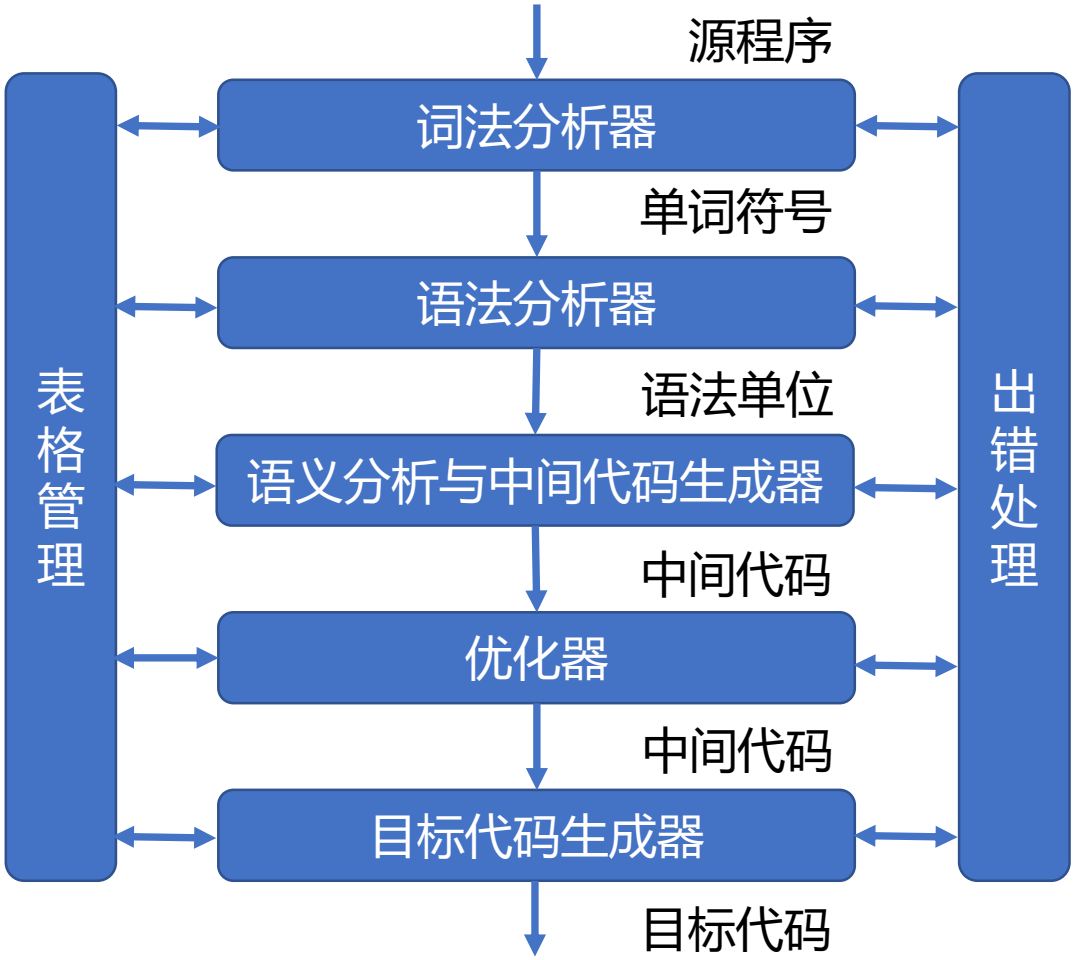
1.2 编译过程概述

- （5）目标代码生成：把中间代码变换为特定机器上的低级语言代码，如果产生出足以发挥硬件效率的目标代码是一件非常不容易的事情。
  - 硬件系统功能部件的运用
  - 机器指令的选择
  - 各种数据类型变量的存储空间分配
  - 寄存器的调度
- 目标代码的形式
  - 汇编指令代码：需要汇编才能执行
  - 绝对指令代码：可以立即执行
  - 可重定位指令代码：运行前必须借助于一个连接装配程序，把各个目标模块（包括系统提供的库模块）连接在一起，确定程序变量（或常数）在主存中的位置，装入内存中指定的起始地址，使之成为一个可以运行的绝对指令代码程序

第一章 引论

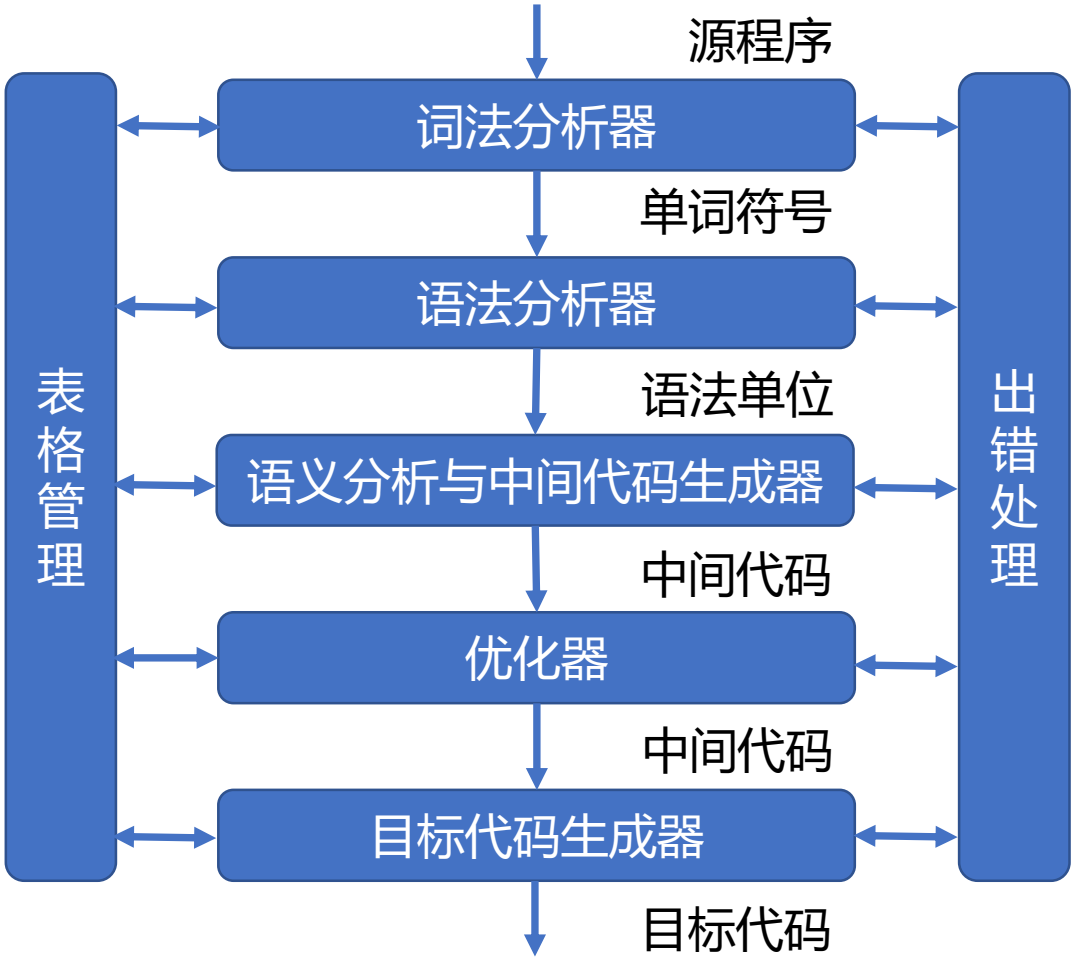
- 1.1 什么是编译程序
- 1.2 编译过程概述
- 1.3 编译程序的结构
  - 1.3.1 编译程序框架
  - 1.3.2 表格和表格管理
  - 1.3.3 出错处理
  - 1.3.4 遍
  - 1.3.5 编译前端和后端
- 1.4 编译程序与程序设计环境
- 1.5 编译程序的生成

1.3.1 编译程序框架



- 词法分析器，又称扫描器，输入源程序，进行词法分析，输出单词符号。
- 语法分析器，又称分析器，对单词符号串进行语法分析，识别出各类语法单位，最终判断输入串是否构成语法上正确的“程序”。

1.3.1 编译程序框架



- 语义分析与中间代码生成器，按照语义规则对语法分析器归约（或推导）出的语法单位进行语义分析，并把它们翻译成一定形式的中间代码。
- 优化器，对中间代码进行优化处理。
- 目标代码生成器，把中间代码翻译成目标代码。

1.3.2 表格与表格管理

- 表格：用于登记源程序的各类信息和编译各阶段的进展状况。
  - 符号表：登记源程序中的每个名字及属性，如变量名、常量名、过程名等。
  - 变量名的信息包括：类型、占用内存大小、地址等等。
  - 通常，编译程序在处理到名字的定义性出现时，把名字的各种属性填入符号表；处理到名字的使用性出现时，对名字的属性进行查证。
- 名字填入表格的各阶段
  - 当扫描器识别出一个名字（标识符）后，把该名字填入符号表，但此时无法完全确定名字的属性。
  - 名字的类型要在语义分析时才能确定。
  - 名字的地址要在目标代码生成时才能确定。

1.3.3 出错处理

- 出错处理的作用：
  - 最大限度的发现各种错误。
  - 准确指出错误的性质和发生错误的地点。
  - 将错误所造成的影响限制在尽可能小的范围，以便使得源程序的其余部分可以继续被编译下去，以进一步发现其它可能的错误。
  - 如果可能，自动校正错误。

1.3.4 遍

- 遍：对源程序或中间结果从头到尾扫描一次，并做有关加工处理，生成新的中间结果或目标程序。
  - 开始于从外存上获得前一遍的中间结果；
  - 结束于完成相关工作并记录于外存。
- 可以几个不同阶段合为一遍，也可以一个阶段分成若干遍。
  - 如词法分析、语法分析和为一遍，甚至与语义分析和中间代码生成一起和为一遍。
  - 又如优化阶段，按优化目标分为多遍。
  - 遍数多则逻辑清晰，但效率差。

1.3.5 编译前端与后端

- 前端：由与源语言有关但与目标机器无关的部分组成。
  - 词法分析
  - 语法分析
  - 语义分析与中间代码生成
  - 与目标机无关的代码优化
- 后端：与目标机器有关的部分。
  - 与目标机有关的代码优化
  - 目标代码生成

第一章 引论

- 1.1 什么是编译程序
- 1.2 编译过程概述
- 1.3 编译程序的结构
  - 1.3.1 编译程序框架
  - 1.3.2 表格和表格管理
  - 1.3.3 出错处理
  - 1.3.4 遍
  - 1.3.5 编译前端和后端
- 1.4 编译程序与程序设计环境
- 1.5 编译程序的生成

1.4 编译程序与程序设计环境

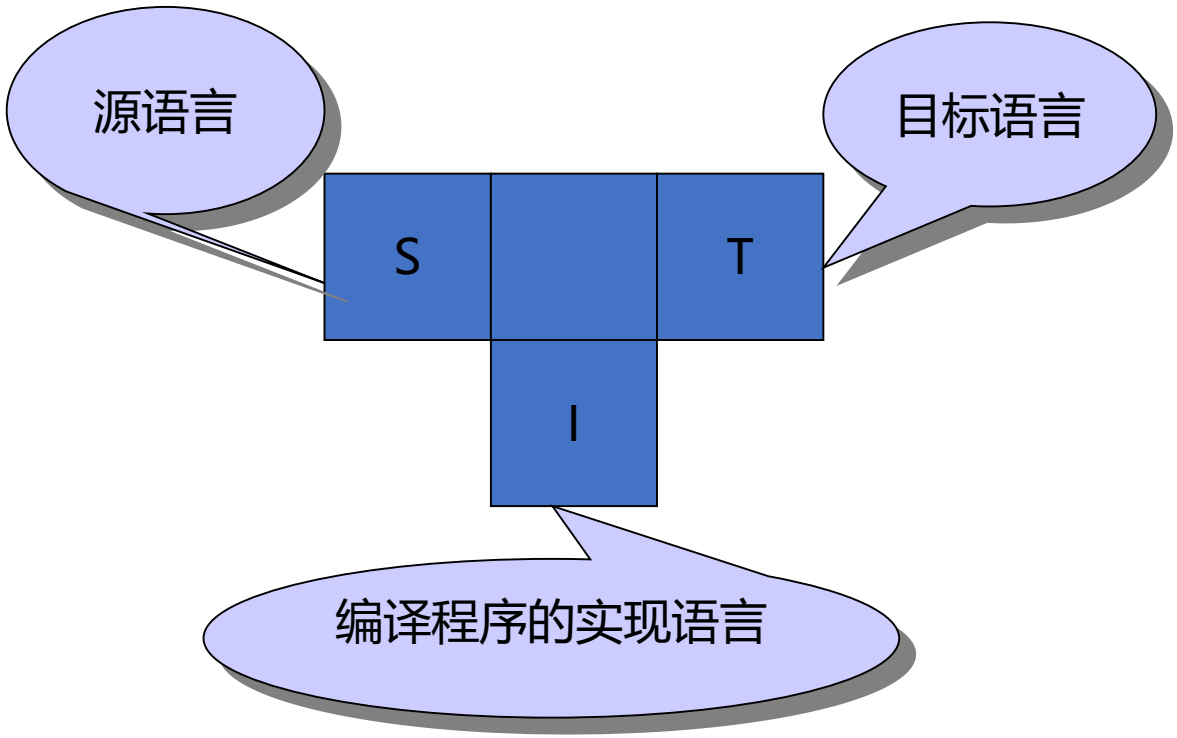
- 程序设计环境
  - 编辑程序
  - 编译程序
  - 连接程序
  - 调试工具

第一章 引论

- 1.1 什么是编译程序
- 1.2 编译过程概述
- 1.3 编译程序的结构
  - 1.3.1 编译程序框架
  - 1.3.2 表格和表格管理
  - 1.3.3 出错处理
  - 1.3.4 遍
  - 1.3.5 编译前端和后端
- 1.4 编译程序与程序设计环境
- 1.5 编译程序的生成

1.5 编译程序的生成

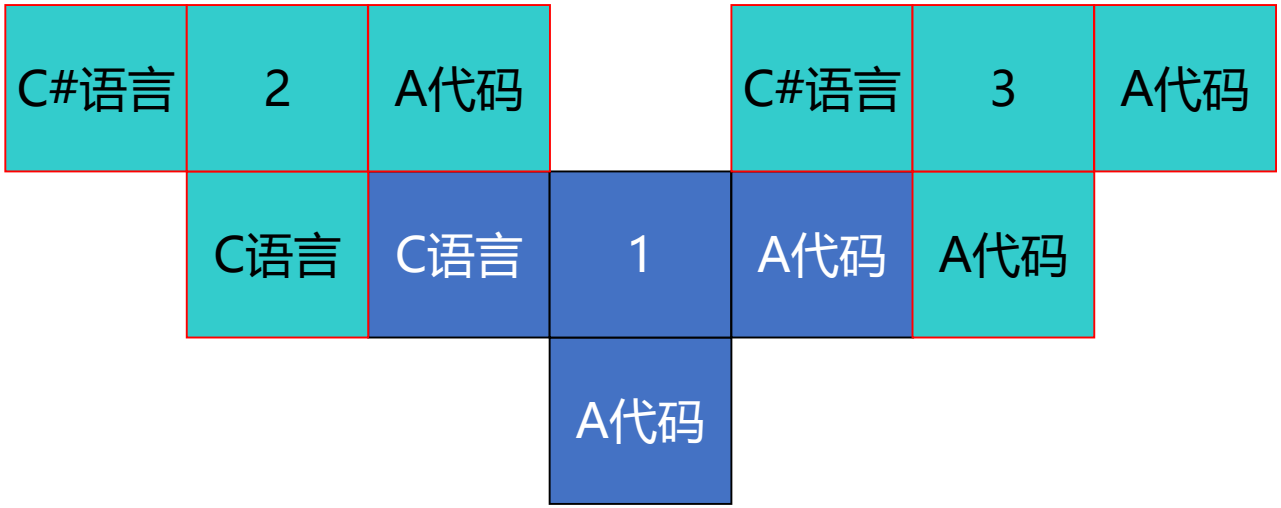
- T形图
  - 源语言S
  - 目标语言T
  - 编译程序实现语言I



1.5 编译程序的生成

- A机器上已有用A机器代码实现的高级语言C的编译程序

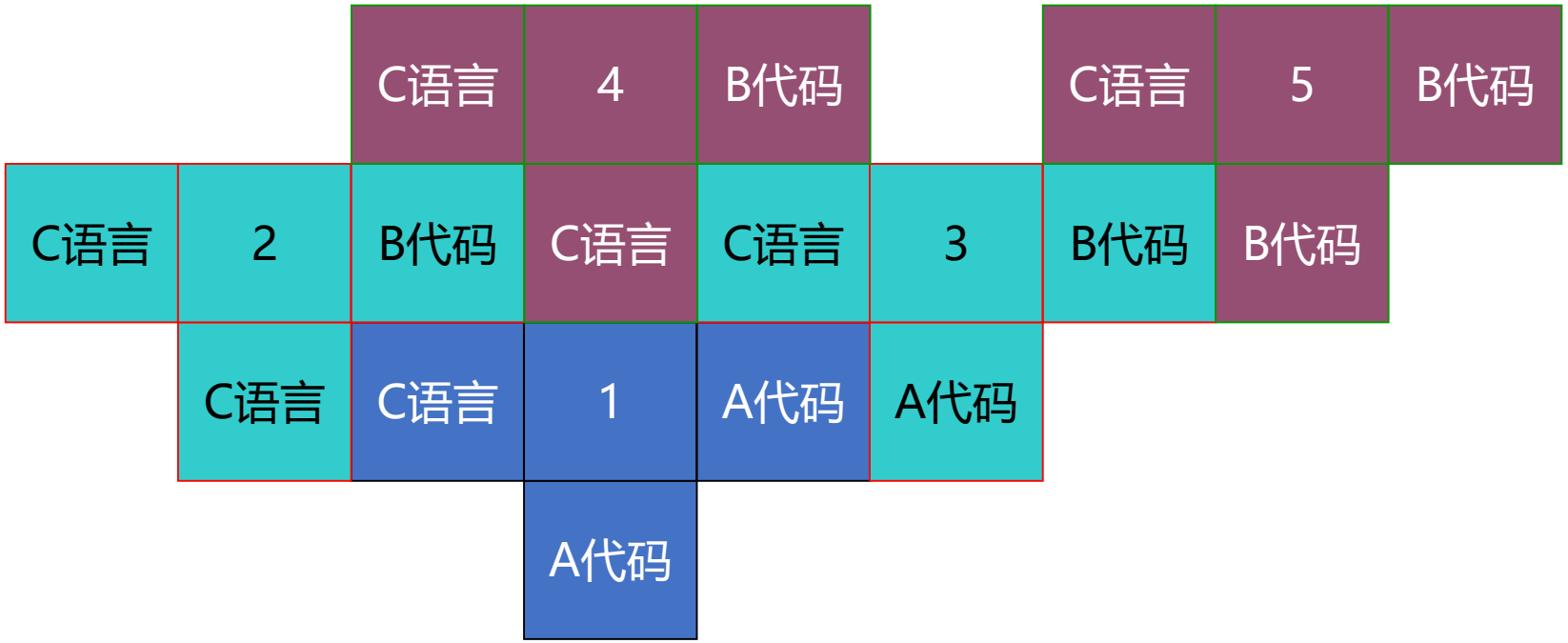
  → A机器上用A机器代码实现的高级语言C#的编译程序



1.5 编译程序的生成

- A机器上已有用A机器代码实现的高级语言C的编译程序

  → B机器上用B机器代码实现的高级语言C的编译程序



1.5 编译程序的生成

- 自编译方式产生编译程序
  - 先对语言的核心部分构造一个小的编译程序；
  - 以它为工具构造一个能够编译更多语言成分的较大编译程序；
  - 如此扩展，最后形成所期望的整个编译程序。

第一章作业

【作业1-1】画图表示编译过程的各个阶段，并简要说明各阶段的功能。