编译原理

北方工业大学信息学院
North China University of Technology Computer Science
束劼
shujie@ncut.edu.cn
瀚学楼1122，88801615

课程概述

课程安排

	内容		
2019.02.27(Week1)	第一章引论	2019.03.26(Week5)	第四章语法分析4
2019.02.28(Week1)	第二章高级程序语言及其语法描述	2019.03.27(Week5)	第四章语法分析6
2019.03.05(Week2)	第三章 词法分析2	2019.04.02(Week6)	第四章语法分析8
2019.03.06(Week2)	第三章 词法分析4	2019.04.03(Week6)	第五章语法分析2
2019.03.12(Week3)	第三章 词法分析6	2019.04.09(Week7)	第五章语法分析4
2019.03.13(Week3)	第三章 词法分析8	2019.04.10(Week7)	第五章语法分析6
2019.03.19(Week4)	第三章 词法分析10	2019.04.16(Week8)	第五章语法分析8
2019.03.20(Week4)	第四章 语法分析2	2019.04.17(Week8)	第六章属性文法和语法制导翻译2

3

	内容		
2019.04.23(Week9)	第六章属性文法和语法制导翻译4	2019.05.21(Week13)	实验一 词法分析
2019.04.24(Week9)	第七章 语义分析和中间代码产生2	2019.05.22(Week13)	实验一 词法分析
2019.04.30(Week10)	第七章 语义分析和中间代码产生3-5	2019.05.28(Week14)	实验一 词法分析
2019.05.01(Week10)（放假）	第七章 语义分析和中间代码产生6	2019.05.29(Week14)	实验一 词法分析
2019.05.07(Week11)	第七章 语义分析和中间代码产生6-8	2019.06.04(Week15)	实验二 语法分析
2019.05.08(Week11)	第七章 语义分析和中间代码产生10	2019.06.05(Week15)	实验二 语法分析
2019.05.14(Week12)	第八章 符号表1 第九章 代码优化与代码生成 1	2019.06.11(Week16)	实验二 语法分析
2019.05.15(Week12)	第九章 代码优化与代码生成2-3	2019.06.12(Week16)	实验二 语法分析

4

实验安排

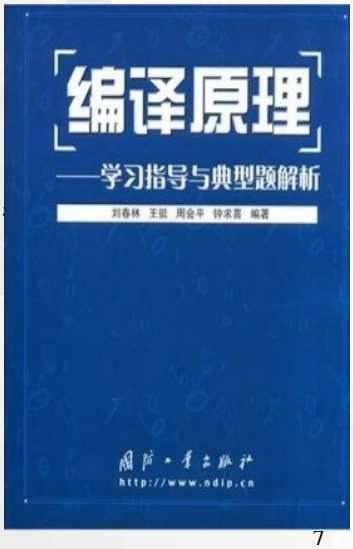
- 实验一：词法分析：13周和14周
- 实验二：语法分析：15周和16周
- 实验地点：瀚学楼910，911
- 指导教师：杨健，13488702754

考核与教材

- 考核方法：
  ① 平时作业10%
  ② 实验20%
  ③ 期末70%
- 教材：
  陈火旺等《程序设计语言编译原理》，国防工业出版社
- 参考书：
  ① 吕映芝。《编译原理》，清华大学出版社。
  ② 杜淑敏。《编译程序设计原理》，北京大学出版社

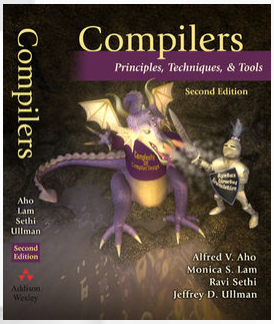
考核与教材

- 刘春林等编著，《编译原理：学习指导与典型题解析》，国防工业出版社
- 以陈火旺院士等编写的《程序设计语言编译原理（第三版）》教材的结构和内容为主线编写而成，旨在帮助学生正确理解书中的概念和原理，把握重点和难点，掌握解题技巧。每一章均包括学习要点、典型题解析和习题与解答3部分。
- 业出版社



英语教材

- Alfred V. Aho et al.
  Compilers: principles, techniques, & tools
  (编译原理技术与工具)
- Andrew W.Appel et al.
  Modern Compiler Implementation in C
  (现代编译原理—C语言描述)
- Steven S. Muchnick
  Advanced Compiler Design and Implementation
  (高级编译器设计与实现)



- 这三本书在编译原理领域分别被称为“龙书”、“虎书”和“鲸书”。

英语教材

- “龙书”最早出版于1986年，第二版中删除了语法制导翻译中的递归计算方法等过时技术，增加了面向对象编译、类型检查等新技术。
- “虎书”包括C语言版本、Java语言版本和ML语言版，在“龙书”知识点基础上，增加了数据流分析、循环优化和内存管理等内容。
  - ML语言是通用的函数式编程语言，由爱丁堡大学的Robin Milner及他人在二十世纪七十年代末开发的。
- “鲸书”重点介绍对编译器后端优化的处理，因此更适合作为研究生的教材或参考书。

第一章 引论

第一章 引论

- 本章目录
  - 1.1 什么是编译程序
  - 1.2 编译过程概述
  - 1.3 编译程序结构
  - 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

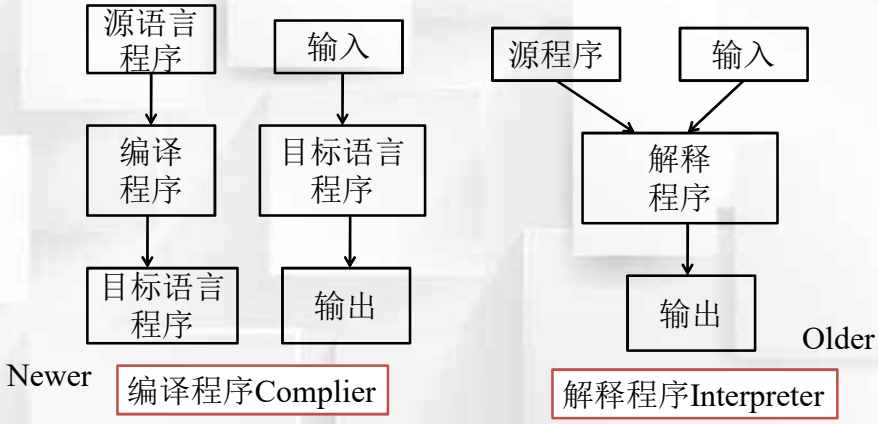
- 大纲要求
  1. 掌握：编译程序的结构，编译程序的生成。
  2. 理解：表格与表格管理。
  3. 了解：编译过程及编译程序总框图。

1.1 什么是编译程序

编译程序和解释程序

1.1 什么是编译程序

- 编译程序Compiler vs 解释程序Interpreter



1.1 什么是编译程序

- 编译程序Compiler vs 解释程序Interpreter

编译程序，能够把某一种语言程序（源语言程序）转换成另一种语言程序（目标语言程序），转换前后的程序在逻辑上是等价的。

解释程序，是另一个常用的语言处理程序，直接在源语言程序上处理输入数值。按照源程序的逻辑流程进行工作。

区别：

① 由编译程序生成的目标程序在处理输入时，通常比解释程序直接对程序与输入处理的速度要快。

② 解释程序在进行错误诊断时比编译程序效果更好，因为解释程序是逐条对源程序进行处理。

1.1 什么是编译程序

- 编译程序Compiler vs 解释程序Interpreter

编译程序，批量编译，主要作用于低级"low level"语言
C, C++

解释程序，主要作用于高级语言
Python

解释程序解释执行源程序，但并不生成目标程序。

某些语言两种都需要，比如Java
Interpreter + "Just in Time (JIT)" compiler

编译程序的历史

1.1 什么是编译程序

- 1954 IBM 发明了一个机器704
  costs of software > costs of hardware

- 如何使得软件的cost降低呢？
  Speedcoding (过去叫解释程序Interpreter ）
  by 1953 John Backus
  程序运行变快了，但是写代码却慢了10-20倍

19

1.1 什么是编译程序

- FORTRAN
  Formula Translation Project, 1954-1957
  第一个编译程序

1958， 50% 代码用FORTRAN写成，大大提高了代码运行速度。提高抽象程度，提高程序员的产出，让大家更容易使用电脑。

20

1.1 什么是编译程序

- FORTRAN One

  Formula Translation Project, 1954-1957
  第一个编译程序

  优点：编写转换方式的时间只需要一年即可；
  　　把理论+实际结合起来。
  缺点：无法判断代码的长度。

  现代的自动编译程序也使用了它的框架。

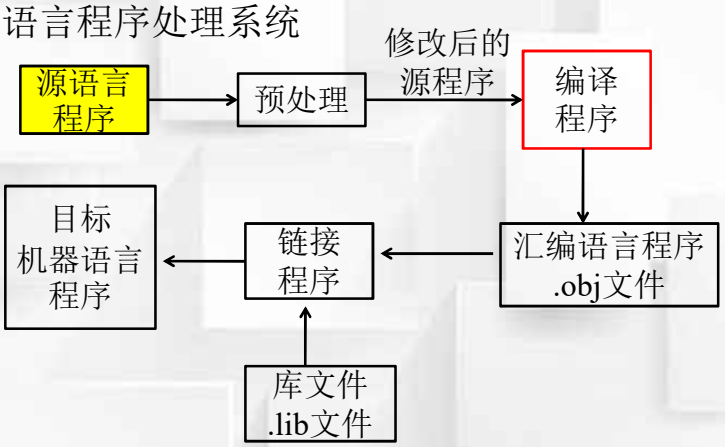
21

语言程序处理系统

22

1.1 什么是编译程序

- 语言程序处理系统



1.1 什么是编译程序

- 源语言程序：
  - **高级语言**：从具体机器中抽象出来，摆脱了依赖具体机器的计算机程序设计语言。高级语言程序需要翻译（编译）成最终能够直接执行的机器语言程序才能执行。
  - 例如：Fortran、Pascal、C 语言等

    特点：不依赖具体机器，移植性好、便于描述问题处理过程和算法、易使用、易维护等。

1.1 什么是编译程序

- 源语言程序：
  - **低级语言**：字位码、机器语言、汇编语言
    特点：与特定的机器有关，效率高、灵活，但使用复杂、繁琐、编写费时、易出错。
  - **脚本（描述）语言**：脚本语言又被称为扩建的语言，或者动态语言，是一种编程语言，用来控制软件应用程序，脚本通常以文本（如ASCII)保存，只在被调用时进行解释或编译。例如：JavaScript，VBScript等。

1.1 什么是编译程序

- 源语言程序：

为什么要对源语言程序使用编译程序？

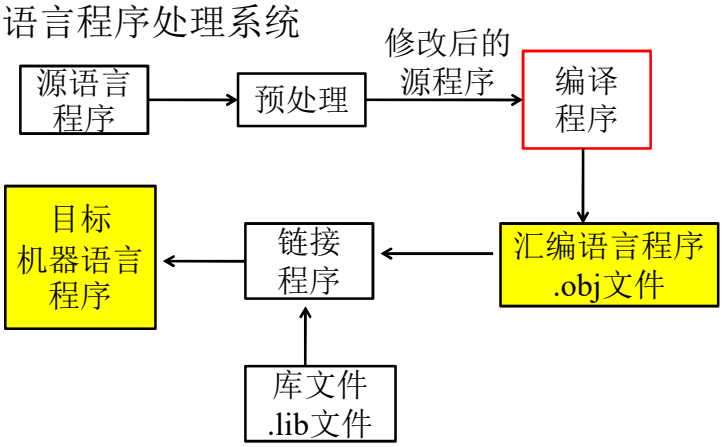
用高级语言编制的程序，计算机不能立即执行,必须通过一个“翻译程序”加工，转化为与其等价的机器语言程序，机器才能执行。这种翻译程序，称之为“编译程序”。

翻译程序扫描所输入的源程序，并将其转换为目标程序。

低级语言需要翻译吗？

1.1 什么是编译程序

- 语言程序处理系统



1.1 什么是编译程序

- 汇编语言程序与目标机器语言程序：
- 源程序是用高级语言或汇编语言编写的，而目标程序则是用目标语言表示的。

高级语言程序 → 编译程序 → 低级语言程序

汇编语言程序 → 汇编程序 → 机器语言程序

汇编程序与编译程序都是翻译程序，主要区别是加工对象的不同。

1.1 什么是编译程序

- 汇编语言程序与目标机器语言程序：
  - **汇编语言**：用助记符代替机器语言二进制编码的一种语言，这就是汇编语言。但是计算机并不能直接识别这种符号化语言，用汇编语言编写的程序必须翻译成机器语言之后才能执行，这种“翻译”是通过专门的软件——汇编程序实现的。
  - **计算机的机器语言**：二进制形式的指令集合称为该计算机的机器语言，它是计算机惟一能够直接识别并接受的语言。

1.1 什么是编译程序

- 链接程序两大功能：
  - **链接功能**：大型程序通常是分片进行编译程序处理，因此编译后的目标机器语言程序需要链接在一起。除了链接目标机器语言程序外，还需要跟其他的文件链接，比如库文件和其他目标程序文件等。
  - **载入功能**：把所有可执行目标文件载入内存。

1.1 什么是编译程序

- 编译程序分类：
  - ① 诊断编译程序
    帮助程序调试或测试
  - ② 优化编译程序
    提高代码效率
  - ③ 交叉编译程序
    在一个平台上生成另一个平台上的可执行代码
  - ④ 可变目标编译程序
    如果除与目标机相关的部分之外，具有通用性…

1.2 编译过程概述

1.2 编译过程概述

- 编译过程可大体上分为五个阶段：
  ① 词法分析 Lexical Analysis
  ② 语法分析 Parsing
  ③ 语义分析与中间代码产生 Semantic Analysis
  ④ 优化 Optimization
  ⑤ 目标代码生成 Code Generation

由于代码大都是英语的方式表述，因此，前3个阶段（句法与类型）与人们理解英语的方式相近

33

1.2 编译过程概述

- 认识单词
  由字母组成的最小单元

  This is a book.

- 词法分析的重要性

  Ist hi sab ook

很难看明白这是什么，跟咱们常用的识字方式不同，需要一个分隔词汇的方式

34

1.2 编译过程概述

- 词法分析 Lexical Analysis

词法分析：对输入的源程序，扫描分解，识别出单词(words)与标识符(token)，如begin, if, while, (,)等。

A = B ＋C * 60

（标识符名称，属性值）

‘A’ (id, 1), ‘=‘(标识符), ‘B’(id, 2), ‘+’(标识符), ‘C’(id, 3), ‘*’(标识符), ’60’(标识符),

id = identifier, 1、2、3是对应标识符在符号表中的位置，符号表存有标识符的信息，例如，名字和类型。

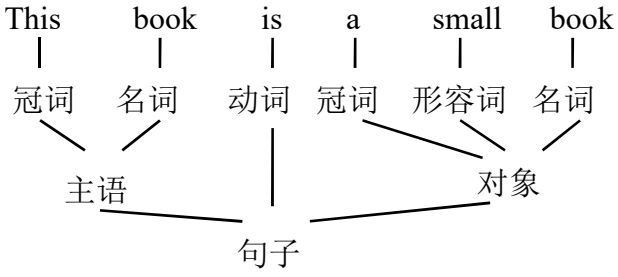
35

1.2 编译过程概述

- 语法分析 Parsing

由词组句，树形图

语法分析：在词法分析的基础上，根据语言的语法规则，把单词符号串分截成各类语法单位。通过语法分析，确定整个输入串是否构成语法上正确的程序。

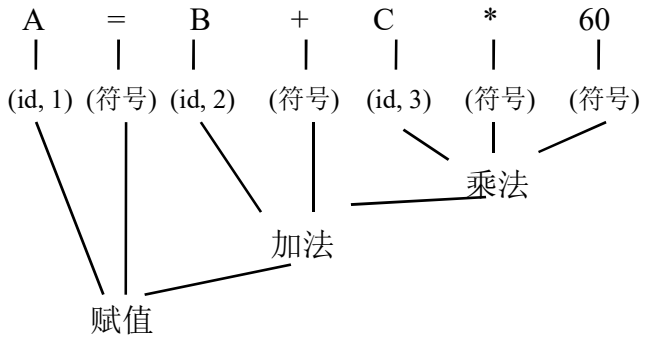


36

1.2 编译过程概述

- 语法分析 Parsing

  词法分析也是一样，比如



37

1.2 编译过程概述

- 语义分析与中间代码产生 Semantic Analysis

  语义分析：识别一句话所表达的实际意义。即弄清楚“干什么了”，“谁干的”，“这个行为的原因和结果是什么”以及“这个行为发生的时间、地点及其所用的工具或方法”等。

  编译程序只会对简单的冲突进行识别与处理。

38

1.2 编译过程概述

- 语义分析与中间代码产生 Semantic Analysis

  Li said Jim loves his girlfriend very much.

  “his”在这里指代谁？Li/Jim ？

  更糟糕的是：

  Li said Li loves his girlfriend very much.

  几个”Li”？

1.2 编译过程概述

- 语义分析与中间代码产生 Semantic Analysis

  对程序语言来说，其定义是十分严格，这些严格的定义会避免出现刚才语义不清的情况。我们来看一个例子：

```
{
        int Li = 3;
    {
        int Jim = 4;
        cout << Jim;
    }
}
```

这个C++程序输出是什么？

输出：4，用了局部定义的变量

1.2 编译过程概述

- 语义分析与中间代码产生 Semantic Analysis

  编译程序会进行大量的错误检验，比如类型检验(type checking)：

  Jim loves her friend very much.

  Jim 和 her 类型上不匹配，他们是不同的人

1.2 编译过程概述

- 语义分析与中间代码产生 Semantic Analysis

  中间代码：对语法分析所识别的各类语法范畴，识别其含义，并进行初步翻译，产生中间代码。

  A = B + C * 60

  T1,T2是临时引入的工作变量

符号	算符	左操作数	右操作数	结果
(1)	*	C	60	T1
(2)	+	B	T1	T2

1.2 编译过程概述

- 优化 Optimization

  优化：对前一步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的目标代码。

  高效：运行更快；
  占用内存更少；
  节省资源；

  这一步更接近于文字的编辑工作。

1.2 编译过程概述

- 优化 Optimization

X = Y * 0　　优化 ➡　　X = 0

1.2 编译过程概述

- 目标代码生成 Code Generation

  目标代码生成：把中间代码变换成特定机器上的低级语言代码。这是一段很复杂的工作，涉及到硬件功能的调用以及寄存器的调度等等。

  通常生成汇编语言。这跟咱们日常的翻译不同。

  中间代码通常会按降序排列，高序列为资源，低序列为汇编语言

1.3 编译程序结构

1.3 编译程序结构

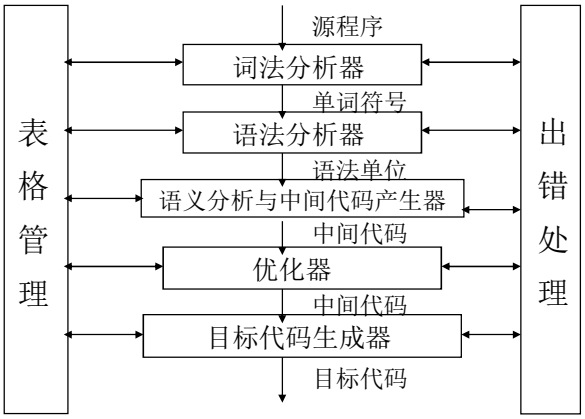


图1.1 编译程序总框

1.3 编译程序结构

- 表格与表格管理
  - 表格：登记源程序的各类信息和编译各阶段的进展状况。最重要的表格是**符号表**。
  - 符号表：登记每个名字及名字的各种属性。比如常量名、变量名、过程名等等。如果是变量名，属性包括类型、占用内存、地址等等。
  - 扫描并识别出一个名字，就填入符号表，其属性需要在后续阶段填入。比如类型是在语义分析时填入，地址则是在目标代码生成时确定。

1.3 编译程序结构

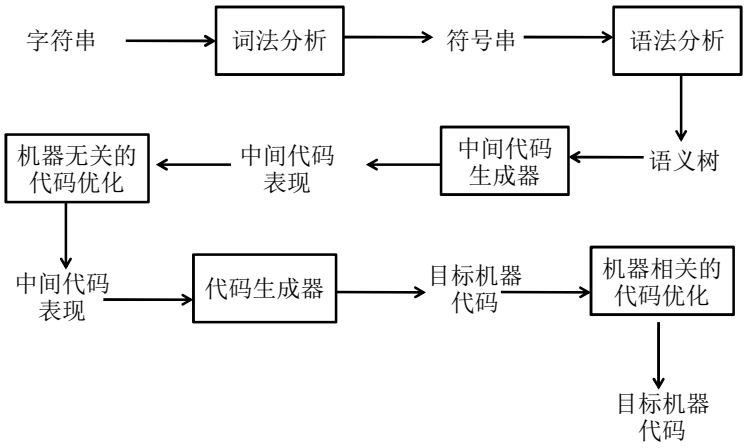
- 出错处理
  - 编译过程的每一阶段都可能错误。其中，绝大多数错误可以在编译的前三阶段检测出来。
  - **语法错误：**是指源程序中不符合语法规则的错误，它们可在词法分析或语法分析时检测出来。
  - **语义错误：**是指源程序中不符合语义规则的错误，这些错误一般在语义分析时检测出来，有的语义错误要在运行时才能检测出来。语义错误通常包括：说明错误、作用域错误、类型不一致等等。

1.3 编译程序结构

- 遍 (Pass/Phases)
  - 编译过程是很多遍(a sequence of phases)，每一遍都把对源程序的理解转换成一个新的理解。
  - 每一遍都是把源程序中的一段代码，通过5个阶段产生目标代码。
  - 实际中，编译程序结构中部分结构可以组合起来，而组合起来这部分结构的中间代码，并不是每一步都需要清晰的得到。

1.3 编译程序结构

- 遍 (Pass/Phases)



51

1.3 编译程序结构

- 前端和后端(front end & back end)
  - **前端**：与分析有关的是前端。词法分析、语义分析与中间代码生成。有的代码优化部分也可包含进前端。
  - **后端**：综合性的处理是后端。与目标相关的部分，目标机器有关代码生成、目标代码生成等。**后端仅依赖于中间语言，与源语言无依赖关系。**
  - **好处**：便于移植、便于编译程序的构造。在中间语言的支持下，实现目标机改变。 如Java的Bytecode可以实现跨平台运行。

52

1.4 编译程序与程序设计环境

1.4 编译程序与程序设计环境

- 程序设计环境
  - **其他需要的工具：**编辑程序、连接程序、测试工具等等。如Visual C++、C++Builder、Visual J++等
  - **程序设计环境：**编译程序与上述工具的结合。
  - **好处：**便于移植、便于编译程序的构造。在中间语言的支持下，实现目标机改变。 如Java的Bytecode可以实现跨平台运行。

1.5 编译程序的生成

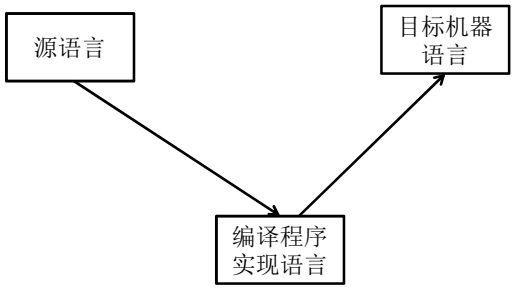
1.5 编译程序的生成

- 编译程序的作用与设计目的
  - 目标程序小，执行速度快。
  - 编译程序小，执行速度快。
  - 诊断能力强，可靠性强。
  - 可移植性强，可扩充性强。

- 编译程序目前大都用**机器语言或汇编语言**作工具，但越来越多的人在**用高级语言写编译程序**。**好处**是节省程序设计时间，构造的编译程序易于阅读、维护和移植。

1.5 编译程序的生成

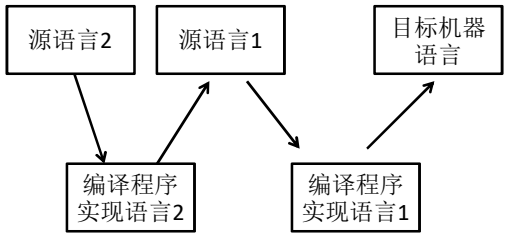
- 编译程序的移植
  - 某个高级语言的编译程序生成



57

1.5 编译程序的生成

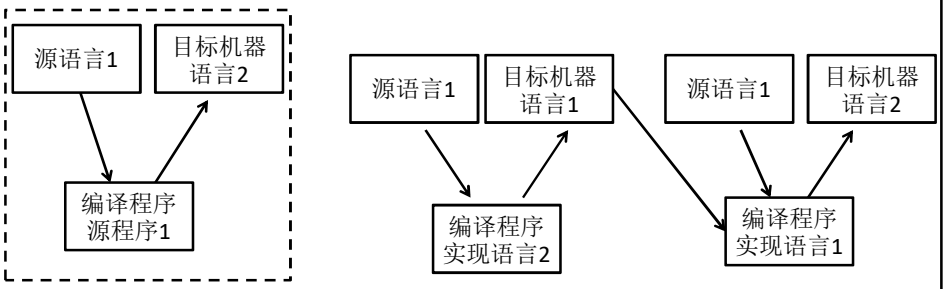
- 编译程序的移植
  - 相同机器，不同高级语言的编译程序生成



58

1.5 编译程序的生成

- 编译程序的移植
  - 不同机器，相同高级语言的编译程序生成



该源语言1可以生成目标机器语言2，则目标机器语言1也可以生成目标机器语言2。目标机器语言1被作为源语言1的编译程序，即编译程序实现语言1。这是自翻译的体现。

59

1.5 编译程序的生成

- 自编译方式

  先对语言的核心部分构造一个小小的编译程序，再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。如此扩展下去，就像滚雪球一样，越滚越大，最后形成人们所期望的整个编译程序。

  例如，词法分析器的自动生成程序。

60

1.5 编译程序的生成

- 词法分析器的自动生成程序

词法规则说明 ⟶ **LEX** ⟶ 词法分析程序
(C程序)

输入：
词法（正规式）
识别动作（ C程序段）
输出：
yylex( ) 函数

61

1.5 编译程序的生成

- 词法分析器的自动生成程序

语法规则说明 ⟶ **Yacc** ⟶ 语法分析程序
(C程序)

输入：
语法规则（产生式）
语义动作（ C程序段）
输出：
yyparse( ) 函数

62

1.5 编译程序的生成

- 并行编译

  把串行的源程序或尚未并行化的并行源程序自动转换成并行计算机上运行的并行目标程序或它能接收的并行源程序。

1.5 编译程序的生成

- 翻译程序的编写系统

  翻译编写系统(TWS, Translator Writing System）：有助于减轻编写翻译程序工作的任何软件系统工具包。如：建立、查找符号表的操作，生成目标代码、出错处理操作。

  TWS分为：

  1. 自动产生编译程序的“编译程序的编译程序”
  2. 面向语法的符号加工程序
  3. 可扩充语言组成的集合，允许用现有类型定义新类型。

第一章 小结

第一章 引论

- 1.1 什么是编译程序：编译程序和解释程序；语言处理系统；源语言程序(高级语言、脚本语言)；汇编语言程序与目标机器语言程序；
- 1.2 编译过程概述：编译过程五个阶段(词法分析、语法分析、语义分析与中间代码产生、优化、目标代码生成)
- 1.3 编译程序结构：表格与表格管理；出错处理；遍 ；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：编译程序的移植；词法分析器的自动生成程序；并行编译

Coursework

- 1.1 请给出编译程序的总体结构图，并简述各个部分的主要功能。
- 1.2 何谓源程序、目标程序、翻译程序、编译程序和解释程序？它们之间可能有何种关系？