编译原理

2020年3月

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

编译原理

上海交通大学

张冬茉

Email:zhang-dm@cs.sjtu.edu.cn

2020年3月

课程目的

编译原理是计算机专业设置的一门重要的专业课程。虽然只有少数人从事编译方面的工作，但是这门课在理论、技术、方法上都对学生提供了系统而有效的训练，有利于提高软件人员的素质和能力。

课程教材和参考资料

- 教材：《编译原理》赵建华 郑滔 戴新宇 译 机械工业出版社 2009年
- 参考资料：《程序设计语言编译原理》（第3版）陈火旺 刘春林 等编著 国防工业出版社 2014年

第1章 引论

- 编写编译器的原理和技术具有十分普遍的意义，以致于在每一从事计算机专业的人员，在理论研究和技术开发中，本书中的这些原理和技术都会反复用到。
- 编译器的编写涉及到程序设计语言、计算机体系结构、语言理论、算法和软件工程等学科。
- 本章通过描述编译器的组成、编译器的工作环境以及简化编译器建造过程的软件工具来介绍编译。

1.1 语言处理器

- 翻译程序(translator)：描述算法的语言可有很多种，将一种语言写的程序转换成另一种语言写的程序，这就是翻译（translation），实现这种功能的程序便是翻译程序(translator)，显然翻译前的程序与翻译后的程序两者应等价。
- 语言处理器就是一种翻译程序，主要包括：
- 编译程序
- 汇编程序
- 解释程序

1.1 语言处理器——编译器(compiler)

- 编译器(compiler)是一个程序，它读入用某种语言（源语言）编写的程序并将其翻译成一个与之等价的以另一种语言（目标语言）编写的程序（如图1 - 1所示）。作为这个翻译过程的一个重要组成部分，编译器能够向用户报告被编译的源程序中出现的错误。

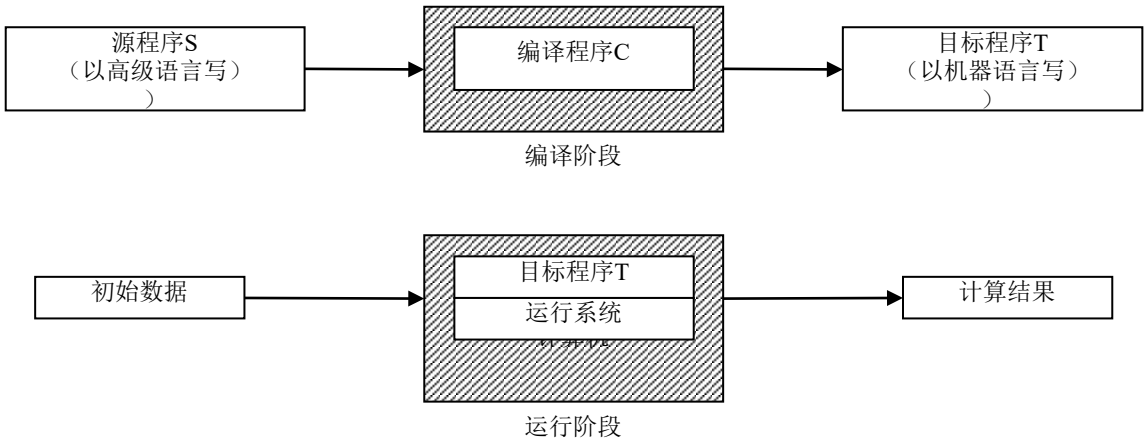


图1.1 程序的编译执行

1.1 语言处理器——汇编程序(assembly program)

- 汇编程序：将可直接执行的机器语言的指令系统符号化，这便是汇编语言，当然汇编语言程序也必须转换成机器语言程序才能执行，这种转换程序便是汇编程序(assembly program)。汇编程序也是一种翻译程序。

(如图1－2所示)

图1.2 汇编程序的执行

1.1 语言处理系统

由编译器和汇编程序组成的语言处理系统结构

（如图1 - 3所示）

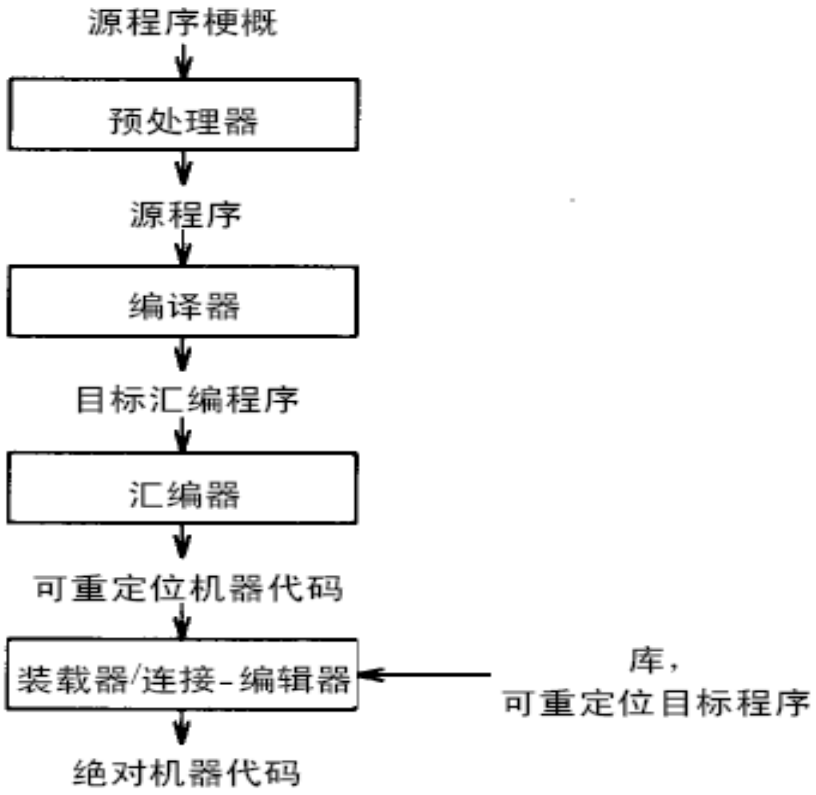


图1-3 一个语言处理系统

1.1 语言处理器——解释程序(interpreter)

- **解释程序**(interpreter)：**解释执行是将源程序中的语句按动态顺序，逐句逐段翻译成可执行代码，一旦具备执行条件(获得必要的初始数据等)立即将这一段代码执行得到部分结果.完成这样功能的程序称为解释程序**(interpreter)（如图1 - 4所示）

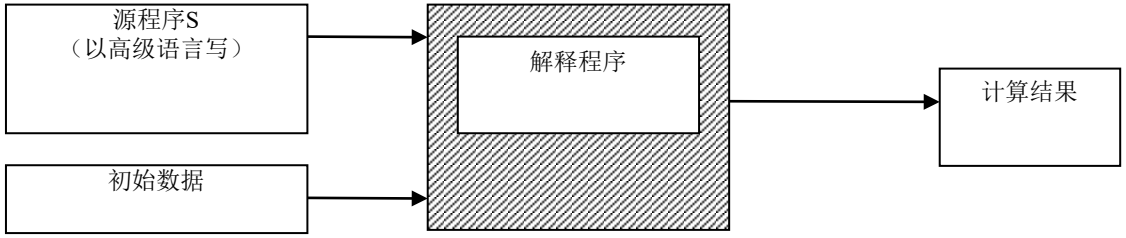


图1.4程序的解释执行

1.2 一个编译器的结构

编译由两部分组成：分析与综合。其中：

- 分析部分将源程序切分成一些基本块并形成源程序的中间表示。主要包括：词法分析，语法分析和语义分析三个部分；
- 综合部分把源程序的中间表示转换为所需的目标程序。主要包括：中间代码生成，机器无关的代码优化和目标代码生成三个部分；综合部分需要大量的专门化技术。

编译的两个大的阶段：前端和后端，其中：

- **前端：**包括依赖于源语言并在很大程度上独立于目标机器的某些阶段或者某些阶段的某些部分。前端一般包括词法分析、语法分析、符号表的建 立、语义分析、中间代码生成以及相关的错误处理。相当一部分代码优化工作也在前端完成。

- **后端：**包括编译器中依赖于目标机器的阶段或某些阶段的某些部分。一般来说，后端完成的任务不依赖于源语言而只依赖于中间语言。后端主要包括代码优化、代码生成以及相关的错误处理和符号表操作。

1.2.1 词法分析——线性分析

- 线性分析：从左到右地读构成源程序的字符流，而且把字符流分组为多个记号（token），而记号是具有整体含义的字符序列。
- 在编译器中，线性分析被称为词法分析或者扫描。

  例如：在词法分析中，赋值语句 position := initial + rate * 60

  中的字符将被分组为以下记号组：

  1. 标识符 position
  2. 赋值符号 :=
  3. 标识符 initial
  4. 加号 +
  5. 标识符 rate
  6. 乘号 *
  7. 整形常量 60

- 在词法分析过程中，分隔这些记号的字符的空格将被删除。

1.2.2 语法分析——层次分析

- **层次分析：**字符串或记号在层次上划分为具有一定层次的多个嵌套组，每个嵌套组具有整体的含义。
- **在编译器中，**层次分析被称为语法分析（parsing 或者syntax analysis）。它把源程序的记号进一步分组，产生被编译器用于生成代码的语法短语。

1.2.3 语义分析

- **语义分析阶段：**检测源程序的语义错误，并收集代码生成阶段要用到的类型信息。语义分析利用语法分析阶段确定的层次结构来识别表达式和语句中的操作符和操作数。

- 语义分析的一个重要组成部分是类型检查。类型检查负责检验每个操作符的操作数是否满足源语言的说明。例如，很多程序设计语言都要求每当一个实数用于数组的索引时都要报错。程序设计语言可能允许一些操作数的强制类型转换。例如，一个二元算术操作符的操作数可以是一个整数和一个实数。在这种情况下，编译器将把整数强制转换成实数。

在机器内部，整数的二进制表示形式不同于实数的二进制表示形式。两个具有相同数值的整数与实数的机器内部表示也不相同。

例如：假定图1-5中的所有标识符都被声明为实数，而60自己却被假定为整数。在对图1-5进行类型检查时编译器会发现*被应用到实数rate和整数60上。一般的解决方法是将整数转换成实数。图1-5b给出了整数转换为实数的方法，即创建一个额外节点inttoreal，显式地将一个整数转换成一个实数。解决类型转换的另一种方法是用一个等值的实数常数来替代整数，因为inttoreal的操作数是常数。

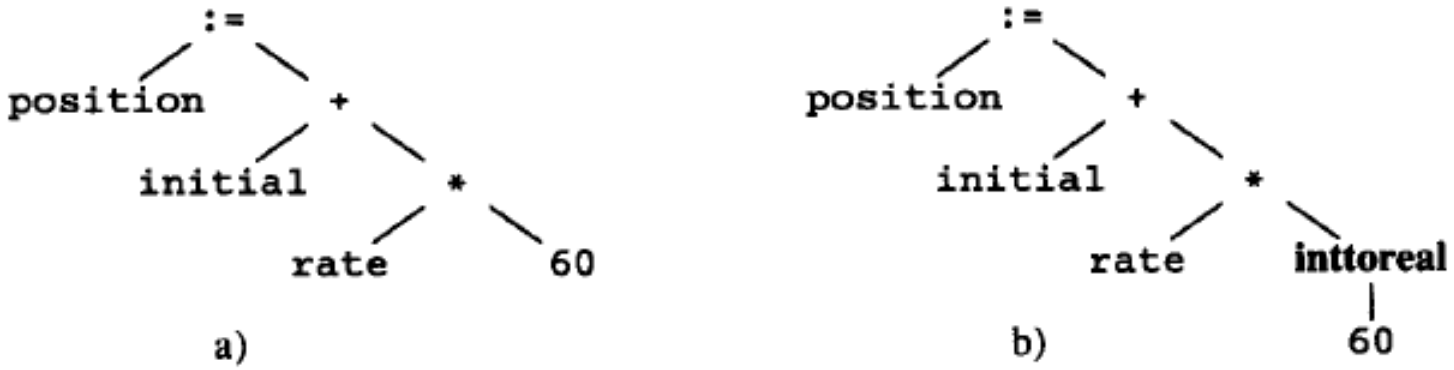


图1-5 语义分析插入了一个整数到实数的转换

1.2.4 中间代码生成

- 某些编译器在完成语法分析和语义分析以后，产生源程序的一个文本形式的中间表示（例如：三地址码）。我们可以将这种中间表示看成是某种抽象机的程序。
- 源程序的中间表示应该具有两个重要性质：
  1、是易于产生；
  2、是易于翻译成目标程序。
- 例如： position : = initial + rate * 60的源程序的三地址码如图1.6所示：

```
temp1 := inttoreal(60)
temp2 := id3 * temp1
temp3 := id2 + temp2
id1 := temp3
```

图1.6程序的三地址码

1.2.5 代码优化

前一阶段产生的中间代码是以语法单位这样的局部区间为单位而产生的，不能保证效率是高的，有必要进行等价变换，使程序占用空间少，运行时间短。代码优化阶段试图改进中间代码，以产生执行速度较快的机器代码。

常用的优化措施有：

1. 删除冗余运算；
2. 删除无用赋值；
3. 合并已知量；
4. 循环优化等；

有些优化措施效果很明显，例如循环中参于运算的运算量，如其值并不随着循环而发生变化，这类运算称为循环不变运算，它完全不必每次循环都计算一次，将它们提到进入循环前计算一次即可。

1.2.5 代码优化

代码优化阶段试图改进中间代码，以产生执行速度较快的机器代码。当然，生成中间代码的一个很自然的算法是对语义分析后的树的每个运算符产生一条指令。

如图1.6所示的四条指令变为如图1.7所示的两条指令:

```
temp1 := inttoreal(60)
temp2 := id3 * temp1
temp3 := id2 + temp2
id1 := temp3
```

图1.6程序的三地址码

```
temp1 := id3 * 60.0
id1 := id2 + temp1
```

图1.7程序的三地址码

1.2.6 代码生成

- 编译的最后一个阶段是目标代码生成，生成可重定位的机器代码或者汇编代码。在这一阶段，编译器为源程序定义和使用的变量选择存储单元，并把中间指令翻译成完成相同任务的机器代码指令序列。这个阶段的一个关键问题是变量的寄存器分配。
- 例如，使用寄存器R1和R2，如图1.7所示的中间代码可以翻译成如图1.8所示的目标代码:

```
LDF R2, id3
MULF R2,R2,#60.0
LDF R2, id2, R1
ADDF R1,R1,R2
STF  id1,R1
```

图1.8程序的目标代码

1.2.7 符号表管理

- 编译器的一个基本功能是记录源程序中使用的标识符并收集与每个标识符相关的各种属性信息。标识符的属性信息表明了该标识符的存储位置、类型、作用域（在哪段程序中有效）等信息。当一个标识符是过程名时，它的属性信息还包括诸如参数的个数与类型、每个参数的传递方法（如传地址方式）以及返回值的类型等信息。

- 符号表是一个数据结构。每个标识符在符号表中都有一条记录，记录的每个域对应于该标识符的一个属性。这种数据结构允许我们快速地找到每个标识符的记录，并在该记录中快速地存储和检索信息。

1.2.8 编译器的遍（趟）

- 编译的若干个阶段通常是以一遍(趟)来实现的，每遍读一次输入文件、产生一个输出文件。编译器的阶段组合为遍的方式千差万别，因此我们趋向于按阶段而不是按遍来讨论编译器。
- 如上所述，多个编译阶段可以被组合为编译的一遍，并且每一遍中的各编译阶段的工作是相互交错的。例如:词法分析、语法分析、语义分析以及中间代码的生成可以被组合为一遍。这样，词法分析形成的记号流可以被直接翻译成中间代码。更详细地说，我们可以认为该编译遍是在语法分析器的管理下进行的。语法分析器根据它读到的记号识别语法结构。当它需要下一个记号时，它通过调用词法分析器获得所需的记号。一旦语法结构找出来了，语法分析器就调用中间代码生成器完成语义分析并生成中间代码的一部分。

1.2.9 编译器的构造工具

人们已经设计出了一些自动设计编译器特定构件的软件工具。这些工具使用了特殊的语言来说明和实现特定程序设计语言构件。很多工具使用了非常复杂的算法。成功的工具都把生成算法的细节隐藏起来而且所产生的构件很容易与编译器的其他构件集成在一起。下面是一些有用的编译器的构造工具:

- 语法分析器生成器
- 扫描器生成器
- 语法制导翻译引擎
- 代码生成器的生成器
- 数据流分析引擎

1.3 编译器的各阶段及结果

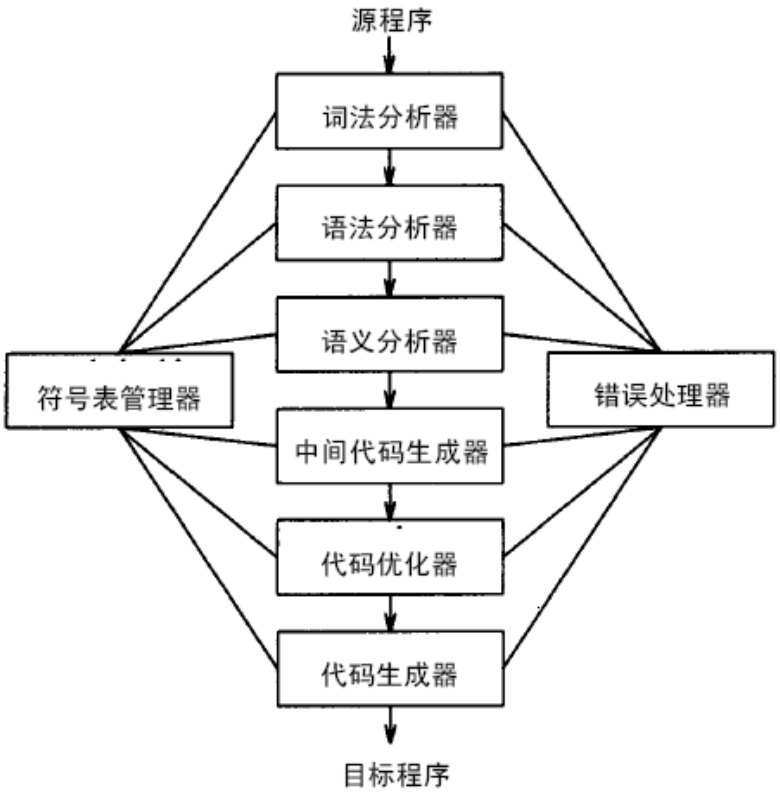


图1-9 编译器的各阶段

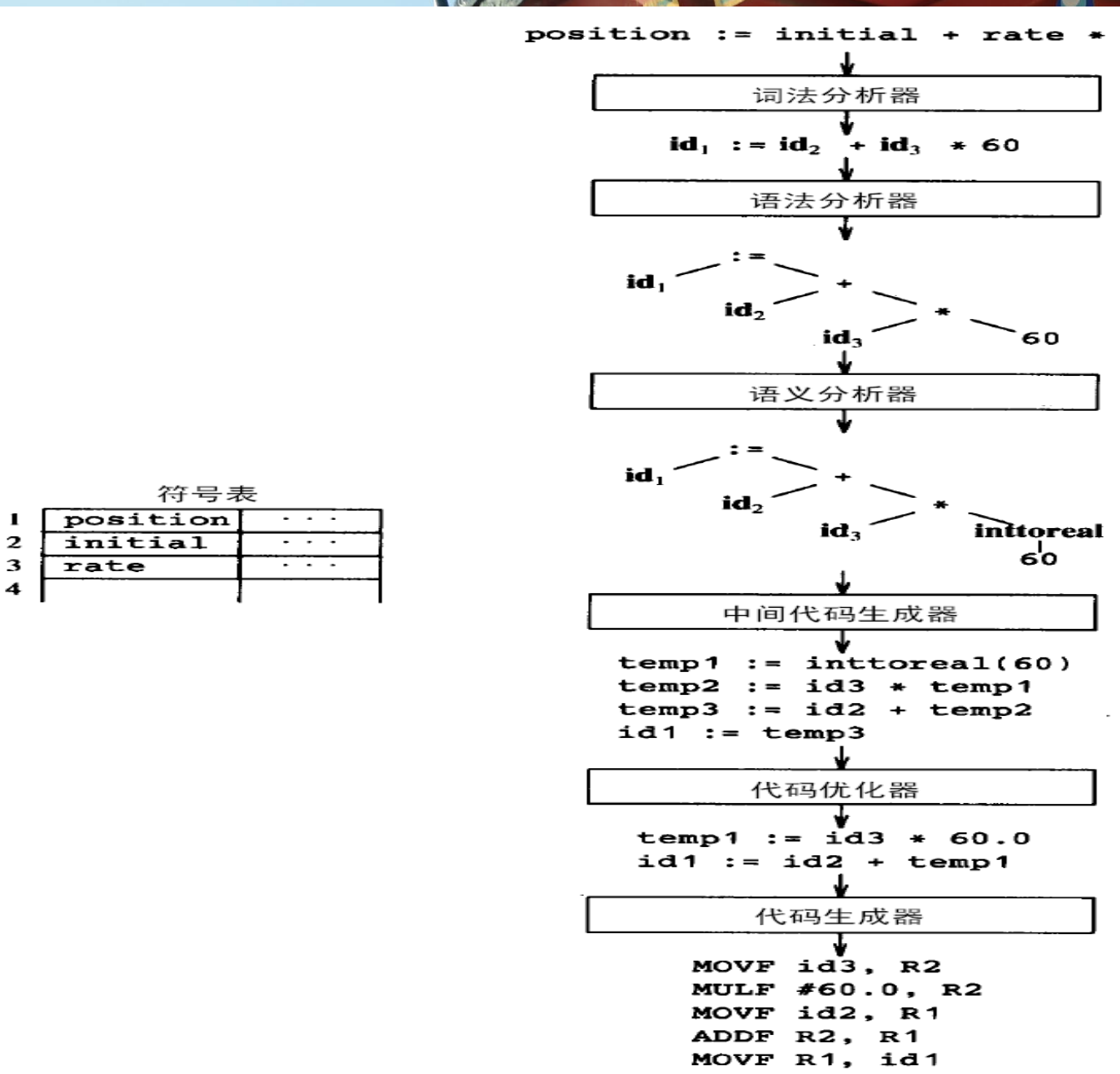


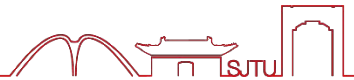
图1-10　一个语句的翻译

第一章 总结

- 语言处理器及其结构
- 编译器的结构
- 编译器的组成部分及其组成部分

  阅读了解P7-23内容

第一章 作业



你认为一个商用软件是应该用具有编译程序的语言开发还是用具有解释程序的语言开发，为什么？

谢 谢!