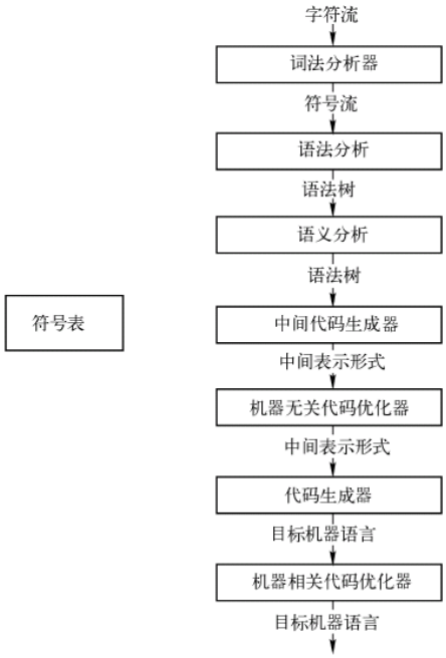
第1章 概论

分析部分和综合部分

- 分析部分（Analysis）
  - 源程序 - 语法结构 - 中间表示
  - 搜集源程序中的相关信息，放入符号表
  - 分析、定位程序中可能存在的错误信息（语法、语义错误）
  - 又称编译器的前端（front end），是与机器无关的部分
- 综合部分（Synthesis）
  - 根据符号表和中间表示构造目标程序
  - 又称编译器的后端（back end），是与机器相关的部分

编译器的步骤



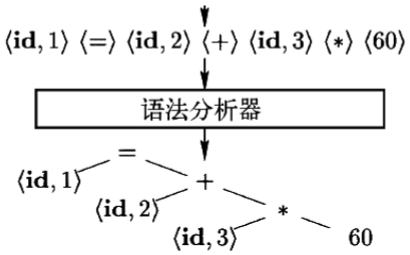
语法树：编译原理里最核心的数据结构

词法分析

- 词法分析/扫描（lexical analysis, scanning）
  - 读入源程序的字符流，输出有意义的词素(lexeme)
  - 基于词素，产生词法单元：
    <token-name, attribute-value>
  - token-name由语法分析步骤使用
  - attribute-value指向相应的符号表条目，由语义分析/代码生成步骤使用

语法分析

- 词法分析后，需要得到词素序列的语法结构
- 语法分析/解析（syntax analysis/parsing）
  - 根据各个词法单元的第一个分量来创建树形中间表示形式。通常是语法树（syntax tree）。
  - 指出了词法单元流的语法结构。



语义分析

- 语义分析（semantic analysis）
  - 使用语法树和符号表中的信息，检查源程序是否满足语言定义的语义约束。
  - 同时收集类型信息，用于代码生成。
  - 类型检查，类型转换。

中间代码生成、代码优化、代码生成

符号表

记录源程序中使用的变量的名字，收集各种属性：名字的存储分配、类型、作用域、过程名字的参数数量、参数类型等等。符号表可由编译器的各个步骤使用。

程序设计语言的基础概念

静态与动态

静态分析：不执行这个程序的情况下即可完成分析，对应编译时刻。（优点：快，可以得到全局信息；缺点：信息可能错误/不精确）

动态分析：在运行程序的情况下进行分析，对应运行时刻。（优点：得到的结果一定是正确的；缺点：只能走一个分支，得到的信息是片面的）

环境与状态

环境：是从名字到存储位置的映射。状态：从内存位置到它们的值的映射。环境储存在符号表中。

参数传递机制

值调用 (call by value)：对实在参数求值/拷贝，再存放到被调用过程的形参的内存位置上。

引用调用(call by reference)：实际传递的是实在参数的地址。