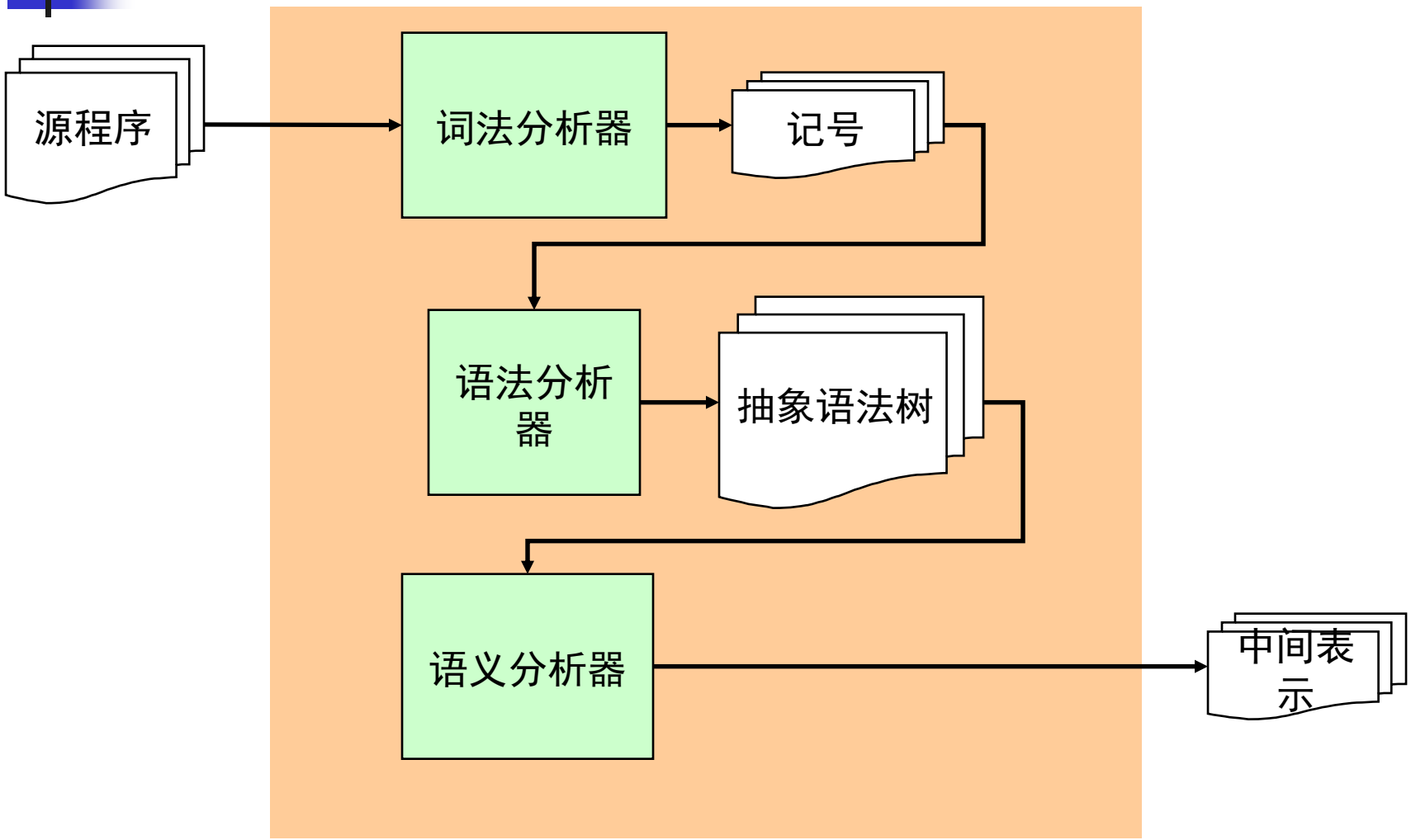
中间表示：
中间代码的地位和作用

编译原理

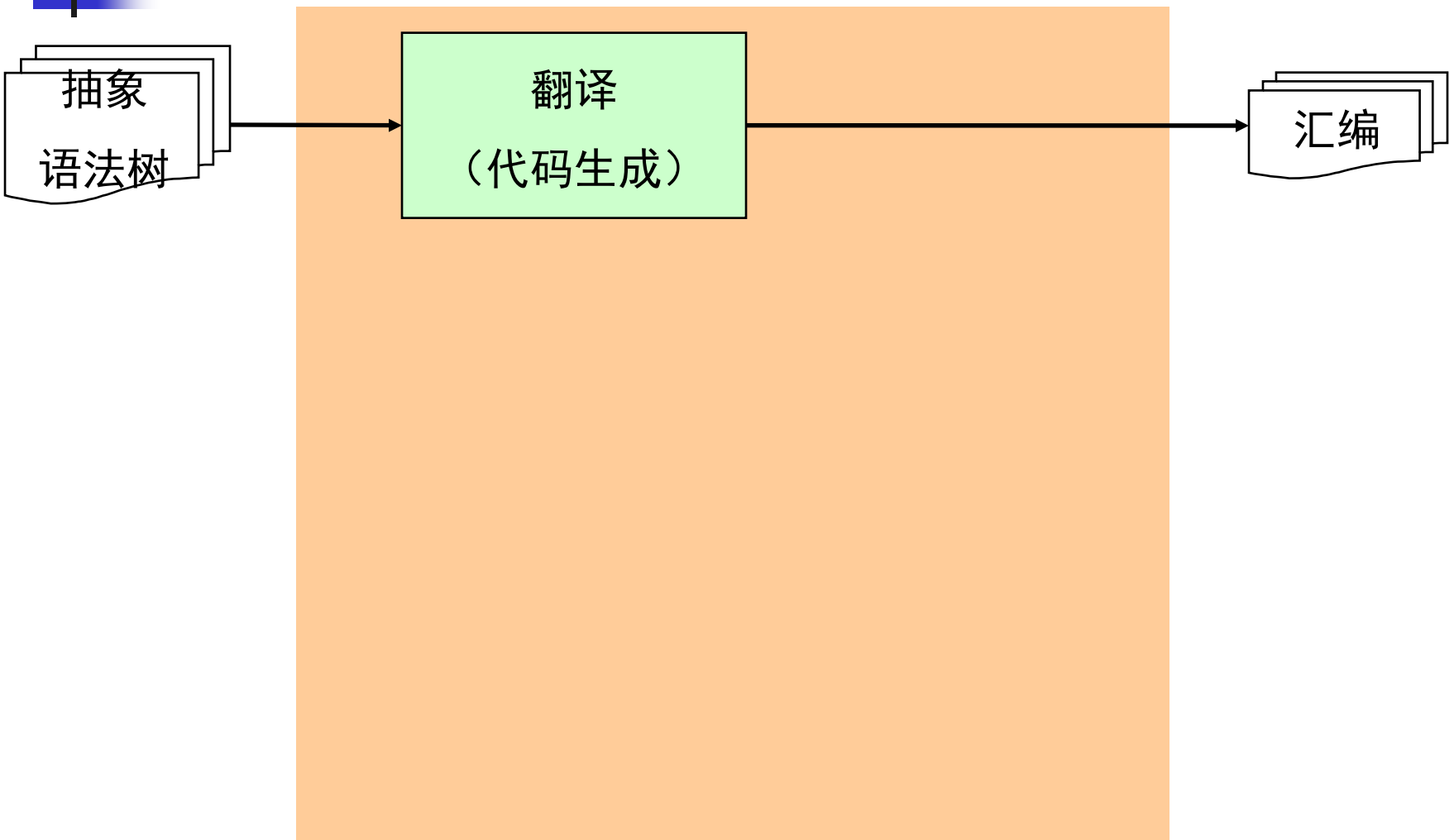
华保健

bjhua@ustc.edu.cn

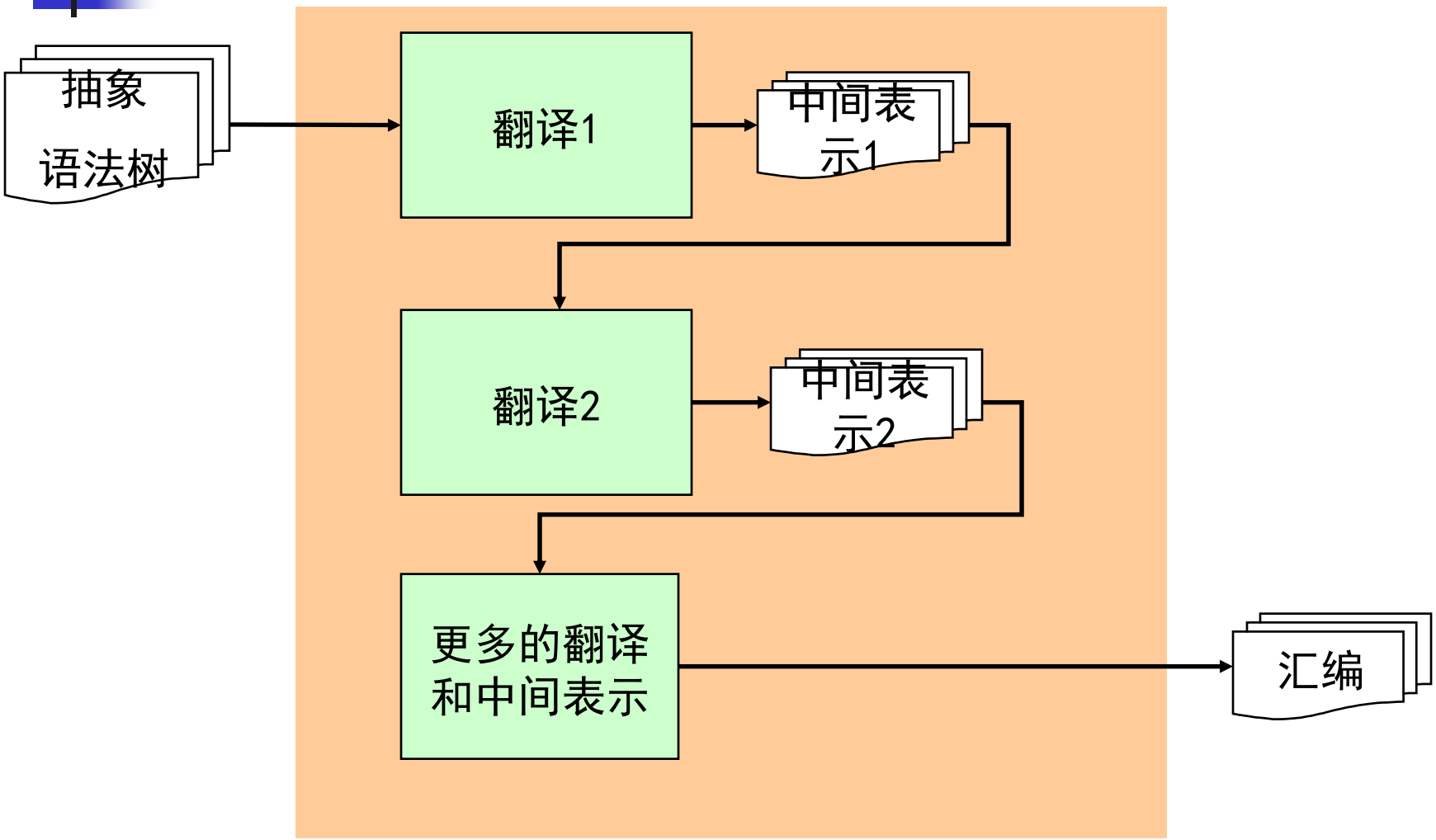
前端

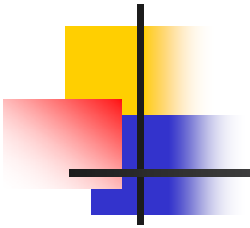


最简单的结构



中间端和后端

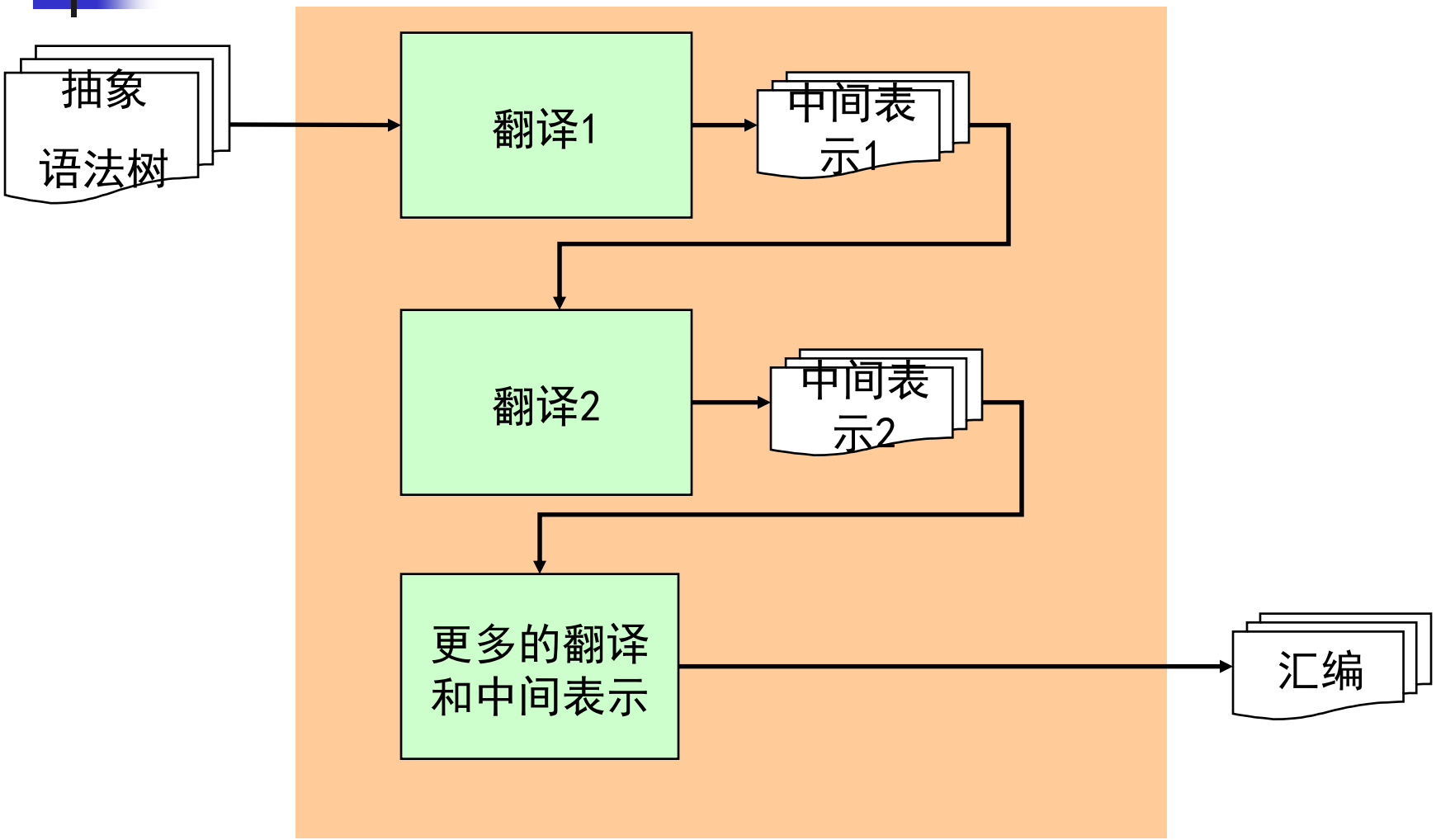




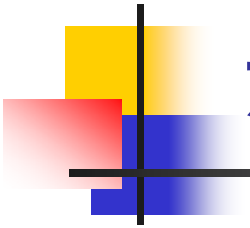
中间代码

- 树和有向无环图（DAG）
  - 高层表示，适用于程序源代码
- 三地址码（3-address code）
  - 低层表示，靠近目标机器
- 控制流图（CFG）
  - 更精细的三地址码，程序的图状表示
  - 适合做程序分析、程序优化等
- 静态单赋值形式（SSA）
  - 更精细的控制流图
  - 同时编码控制流信息和数据流信息
- 连续传递风格（CPS）
  - 更一般的SSA

为什么要划分成不同的中间表示？

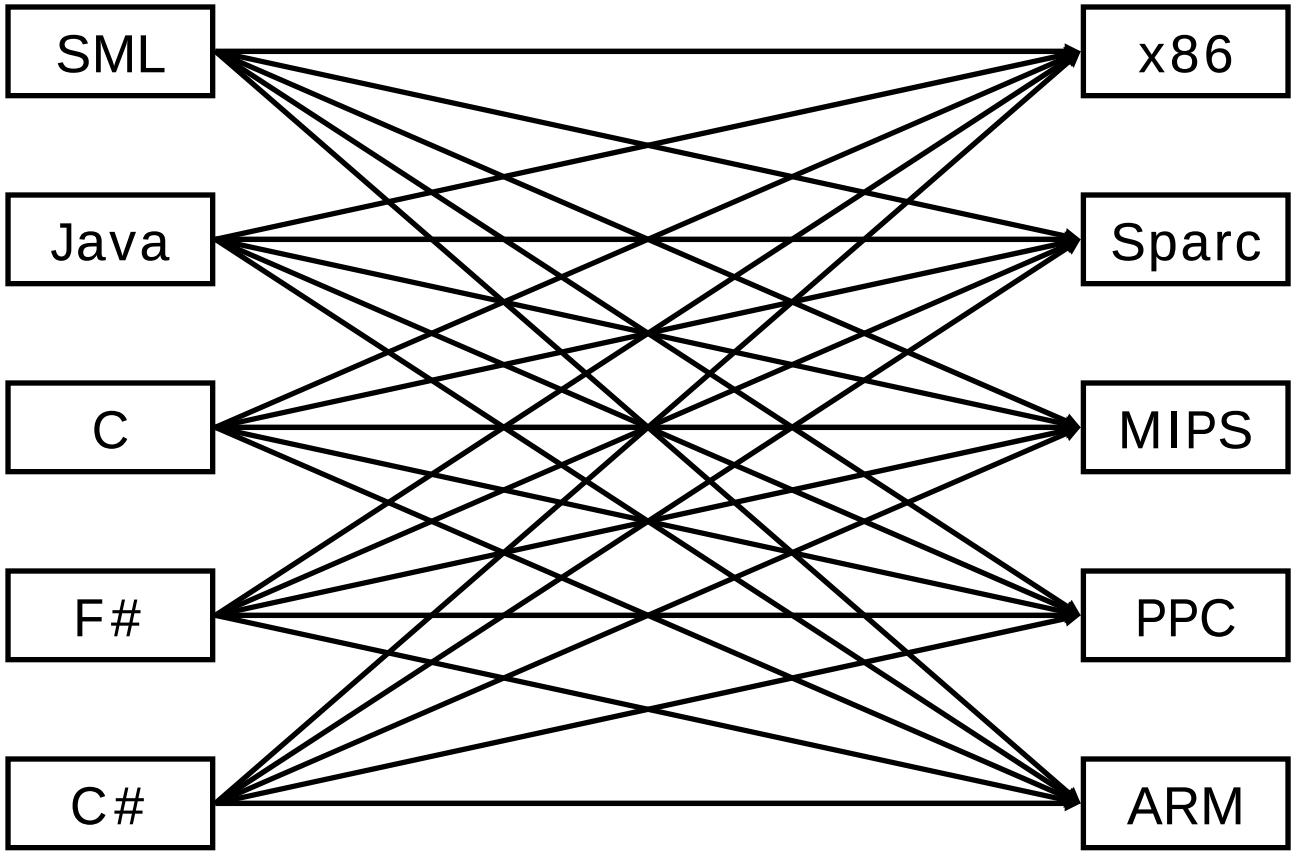


为什么要划分成不同的中间表示？



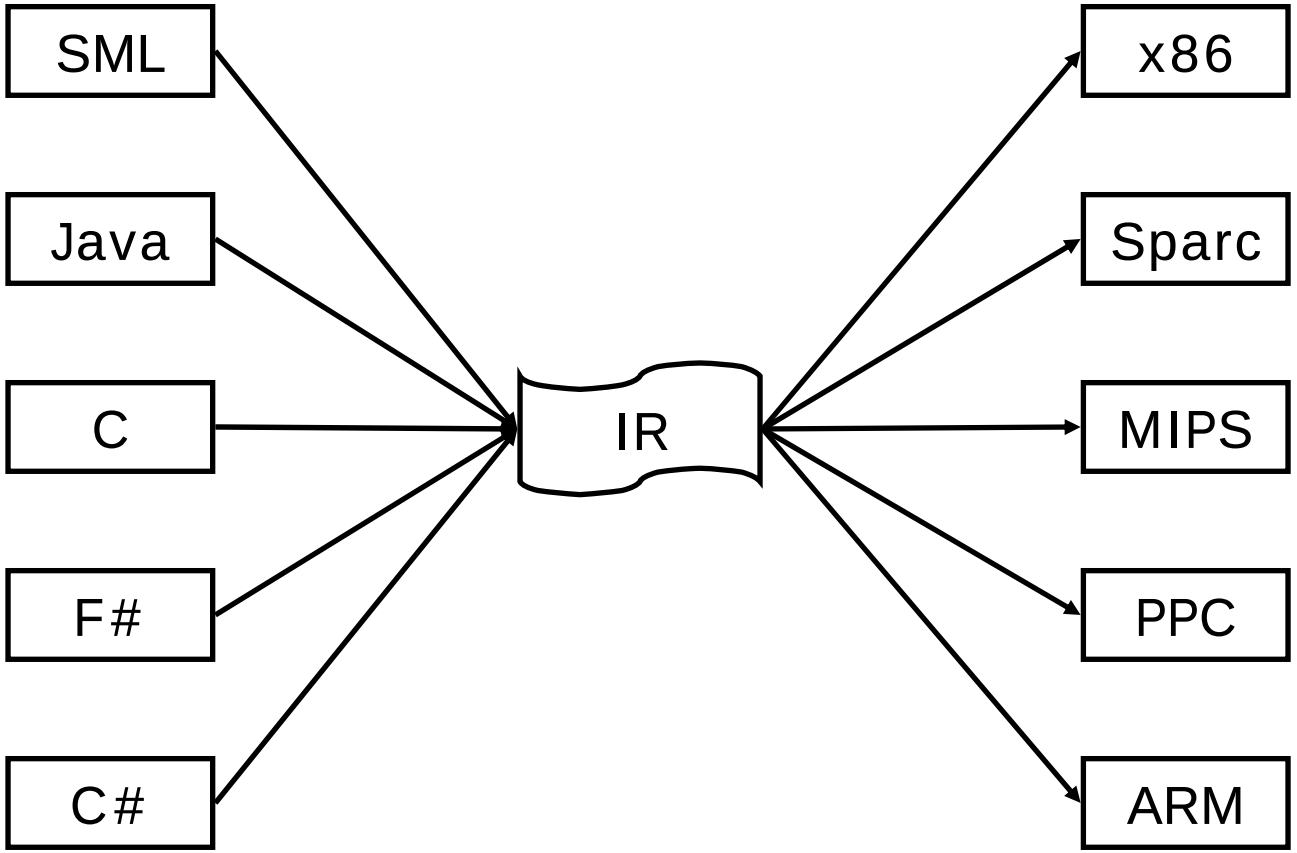
- 编译器工程上的考虑
  - 阶段划分：把整个编译过程划分成不同的阶段
  - 任务分解：每个阶段只处理翻译过程的一个步骤
  - 代码工程：代码更容易实现、除错、维护和演进
- 程序分析和代码优化的需要
  - 两者都和程序的中间表示密切相关
    - 许多优化在特定的中间表示上才可以或才容易进行
  - 课程后续我们继续深入这一话题

通用编译器语言？

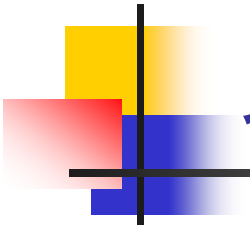


n×m 个编译器

通用编译器语言？



n+m 个编译器



路线图

- 全面讨论中间表示涉及的重要问题和解决方案
  - 详细介绍现代编译器中几种常用的重要中间表示
    - 三地址码
    - 控制流图
    - 静态单赋值形式
  - 详细介绍在中间表示上做程序分析的理论和技术
    - 控制流分析
    - 数据流分析
- 为下一部分讨论代码优化打下基础