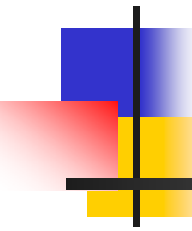


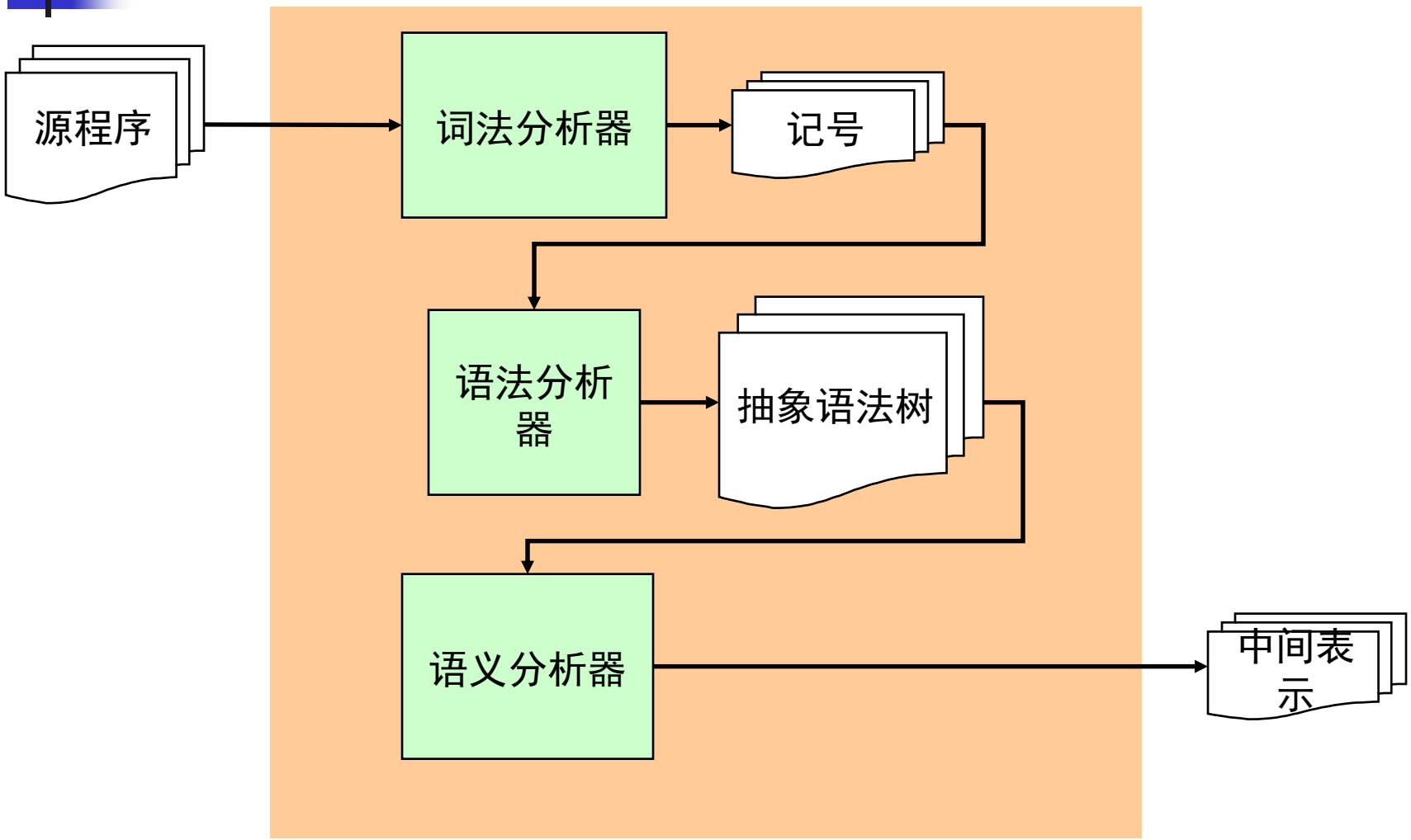
代码优化：基本概念

编译原理

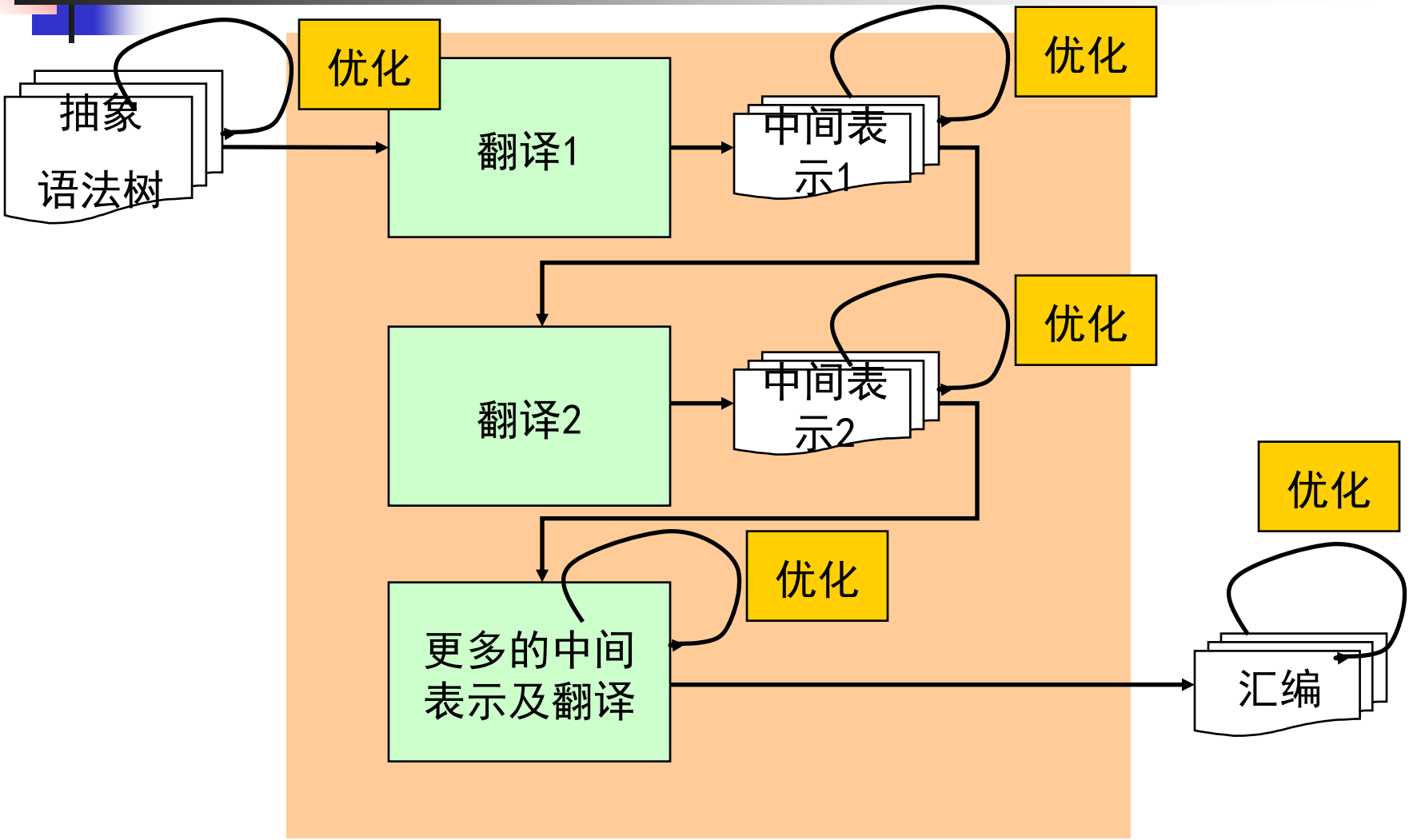
华保健

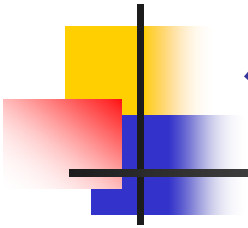
bjhua@ustc.edu.cn

前端



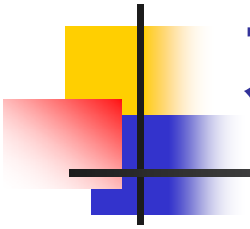
优化的位置





什么是“代码优化”？

- 代码优化是对被优化的程序进行的一种语义保持的变换
- 语义保持：
  - 程序的可观察行为不能改变
- 变换的目的是让程序能够比变换前：
  - 更小
  - 更快
  - cache行为更好
  - 更节能
  - 等等

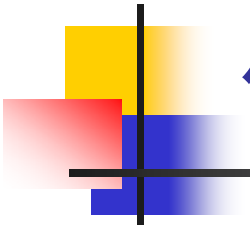


不存在“完全优化”

- 等价于停机问题:
  - 给定程序p，把Opt(p) 和下面的程序比较:

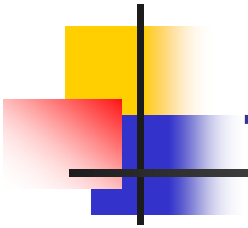
```
L:
    jmp L
```

- 这称为“*编译器从业者永不失业定理*”



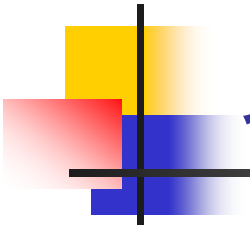
代码优化很困难

- 不能保证优化总能产生“好”的结果
- 优化的顺序和组合很关键
- 很多优化问题是非确定的
- 优化的正确性论证很微妙



正确的观点

- “把该做对的做对”
  - 不是任何程序都会同概率出现
    - 所以能处理大部分常见情况的优化就可以接受
- “不期待完美编译器”
  - 如果一个编译器有足够多的优化，则就是一个好的编译器



路线图

- 前端优化
  - 局部的、流不敏感的
  - 常量折叠、代数优化、死代码删除等
- 中期优化
  - 全局的、流敏感的
  - 常量传播、拷贝传播、死代码删除、公共字表达式删除等
- 后端优化
  - 在后端（汇编代码级）进行
  - 寄存器分配、指令调度、窥孔优化等