编译原理
实验四：目标代码生成

规格严格，功夫到家

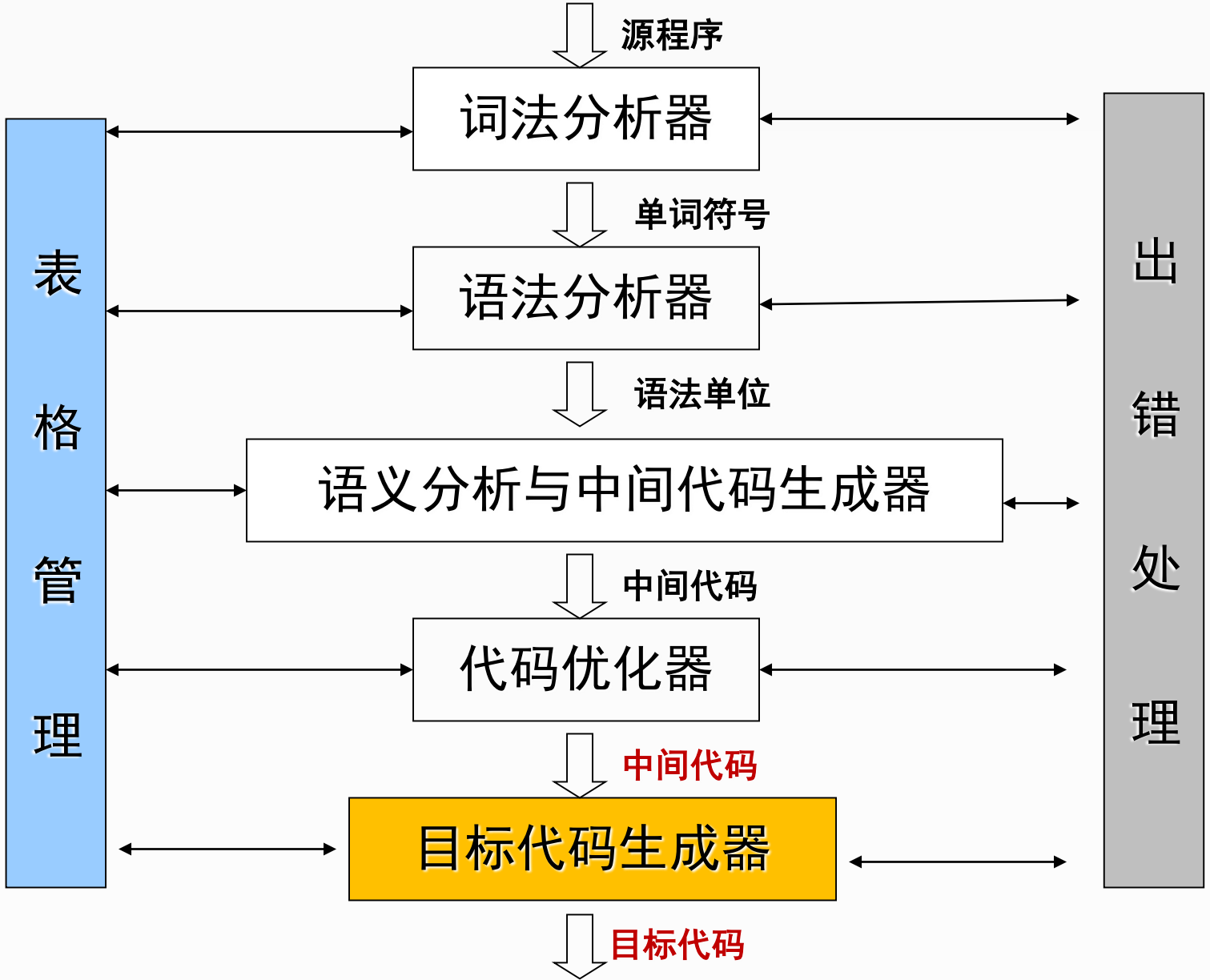
编译程序的总体结构

✔实验一：

✔实验二：

✔实验三：

✔实验四：



实验目的

1. 加深对编译器总体结构的理解与掌握；

2. 掌握常见的RISC-V指令的使用方法；

3. 理解并掌握目标代码生成算法和寄存器选择算法。

实验学时数：2学时

实验内容

1. 将实验三生成的中间代码转换为目标代码（汇编指令）；

2. 使用RARS运行生成的目标代码，验证结果的正确性。

注：

- 实验四不检查out/assembly_language.asm和std/assembly_language.asm是否一致，而是验证通过rars运行汇编指令后得到的结果是否正确。
- Rars使用参考指导书

目标代码生成算法（x86）

参考代码生成算法（P406）：
对每个形如i: x := y op z的三地址语句，给出总体框架。

1. 调用函数getreg(i:x:= y op z)确定可用于保存y op z的计算结果的位置L。L通常是寄存器，也可能是内存单元。
2. 查看y的地址描述符以确定y值当前的一个位置y′。如果y值当前既在内存单元中又在寄存器中，则选择寄存器作为y'。如果y的值还不在L中，则生成指令MOV y′, L。
3. 生成指令op z′, L，其中z′是z的当前位置之一。
4. 如果y和/或z的当前值没有后续引用，在块的出口也不活跃，并且还在寄存器中，则修改寄存器描述符以表示在执行了x:=y op z 之后，这些寄存器分别不再包含y和(或)z的值。

寄存器选择算法（x86）

函数getreg返回保存x := y *op* z的x值的位置L

选择原则：

1. 选择y独占的寄存器，要求y不活跃，且y在此指令后不再被引用
2. 选择空闲寄存器
3. 抢占非空闲寄存器

目标代码生成举例

三地址码：(add, $1, a, b)　假设a在R0, b在R1中

X86平台　　RISC-V平台

- 如果a独占寄存器R0，且a在此指令后不再被引用，可以使用R0存放$1;

目标代码：ADD R0, R1　（成本更低）

- 如果a接下来还要使用，选择空闲寄存器R2存放$1;

目标代码：MOV R2, R0

ADD R2, R1

目标代码生成举例

三地址码：(add, $1, a, b)　假设a在R0, b在R1中

X86平台　　RISC-V平台

- 为$1选择空闲寄存器R2；

目标代码：ADD R2, R0, R1

结论：算法的目标是降低成本，因此和选择的目标平台密切相关。

目标代码生成算法（RISC-V）

对每个形如（op，result,lhs,rhs）的三地址语句

1. 使用寄存器选择算法为result选择寄存器。
2. 如果左操作数lhs已经在寄存器中，使用当前寄存器，右操作数rhs同理。
3. 否则，如果左操作数lhs是变量且在内存中，使用寄存器选择算法为它选择一个寄存器，将内存中的变量值加载到寄存器中，右操作数rhs同理（rhs不能抢占lhs和result占用的寄存器）。
4. 生成汇编指令。

注：以上算法仅供参考，可以有不同实现。

寄存器选择算法（RISC-V）

1. 如果有空闲寄存器，选择空闲寄存器；
2. 否则，夺取不再使用的变量所占的寄存器；
3. 如果所有寄存器中的变量后面都要使用，自行设计算法从被占用的寄存器中夺取一个，并将被夺取寄存器的变量存回内存。（此步骤可选，不加分）

备注：

- 在代码生成时，约定使用RISC-V临时寄存器：t0-t6
- 同时使用 a0，亦即 x10 存放程序的返回值
- 使用input_code.txt样例只需要实现寄存器选择算法的1,2两个步骤
- 使用data/in/reg-alloc.txt样例需要实现寄存器选择算法的1,2,3三个步骤
- 以上算法仅供参考，可以有不同实现

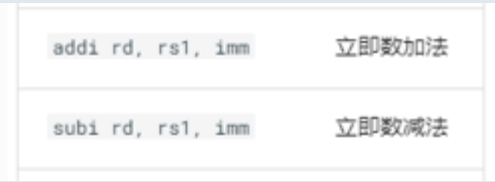
实验步骤

1. 加载前端提供的中间代码，视情况做预处理（预处理思路参考指导书）；
2. 实现寄存器选择算法；
3. 实现目标代码生成算法；
4. 输出生成的目标代码到指定文件中；
5. 使用Rars运行目标代码，验证其正确性。

预处理举例

中间代码：（ADD， $2， 3， b）

RISC-V 指令：



预处理后的中间代码：（ADD， $2， b， 3）

同学们，
独立开始实验