并行编译与优化
Parallel Compiler and Optimization

计算机研究所编译系统室

Lab Five: Register Allocation

实验五：寄存器分配

内容

1. 实验介绍
2. 实验思路
3. 实验验证

1 实验介绍

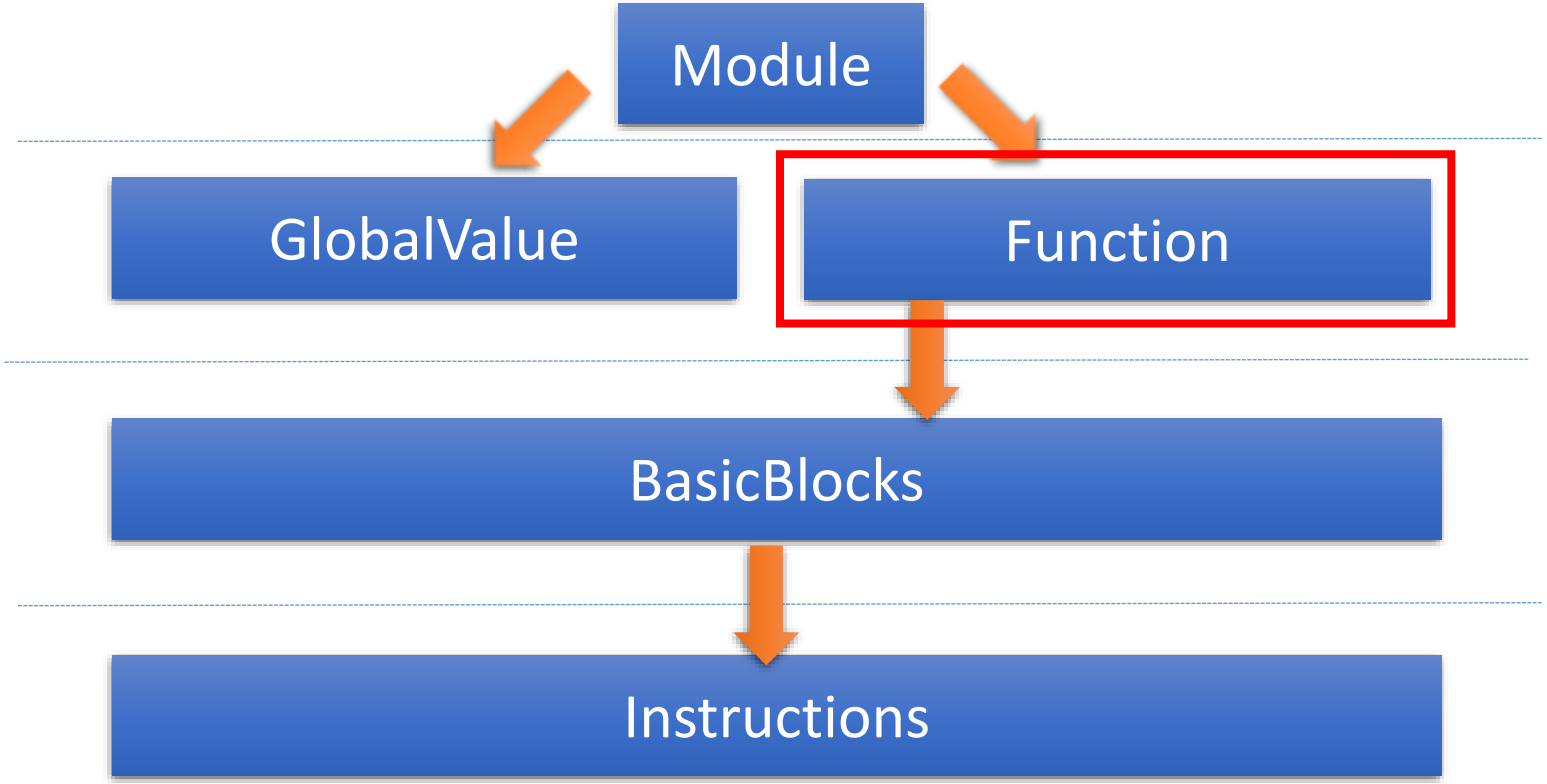
- 实验任务
  - 在编译后端增加对寄存器分配的支持
- 实验目标
  - 通过实验，掌握编译后端寄存器分配方法
- 实验方法
  - 使用计数寄存器分配方法
  - 图着色寄存器分配算法
  - 线性扫描寄存器分配算法

内容

1. 实验介绍
2. 实验思路
3. 实验验证

2.1 回顾: 后端代码生成

- 从IR Module开始，自顶向下遍历
- 将IR逐条翻译成对应的汇编代码（宏扩展的指令选择方法）
- 遍历过程中构建相关符号表



2.1 回顾: 后端代码生成

- 遍历所有Function, 在每个Function内部
  - 做寄存器分配
  - 为该Function使用的每个GlobalValue建立LocalLabel
  - 检查该Function是否有子函数调用，获取参数个数(课堂实验不考虑)
  - 完成该Function头部工作
    - 保存上一级Function的FP, 设置该Function的FP，更新该Function的SP(开辟栈空间)
  - 遍历该Function的所有BasicBlock
  - 完成该Function尾部工作
    - 恢复上一级Function的FP和SP，返回上一级Function

2.2 回顾: 简化的寄存器模型

- 基于ld/st栈维护LocalValue的值
- 所有LocalValue的最新副本保存在栈上
- 维护StackTable符号表，记录每个LocalValue在栈上的位置
- LocalValue的使用和定值
  - 每次使用前，从栈上加载到寄存器
  - 每次定值后，从寄存器存储回栈上，然后所占用的寄存器立即释放

2.3 ARMv7寄存器

- 通用寄存器
  - R0-R3: 传递函数参数和传出函数返回值
  - R4-R10: 通用寄存器
  - R11: FP (栈帧指针寄存器，指向栈底)
  - R12: IP (内部调用暂时寄存器)
  - R13: SP (栈指针寄存器，指向栈顶)
  - R14: LR (链接寄存器，保存返回地址)
  - R15: PC (程序计数器)

2.4 寄存器分配思路

- 将R0~R10, R12作为可用寄存器资源
- 维护RegTable符号表
  - 记录哪些LocalValue分配到寄存器，以及具体寄存器编号
- 维护StackTable符号表
  - 记录哪些LocalValue分配到栈上，以及栈上的位置

2.4 寄存器分配思路

- 可用(但不局限于)所学寄存器分配方法
  - 使用计数方法
  - 图着色方法
  - 线性扫描方法(推荐)
  - 自己设计...

内容

1. 实验介绍
2. 实验思路
3. 实验验证

实验验证

- 功能验证
  - 查看sysyc生成的汇编代码，检验是否实现了寄存器分配
  - 交叉编译sysyc生成的汇编代码，通过qemu运行，查看运行结果，验证寄存器分配功能实现正确
- 性能验证
  - 开发版上测试对比原始sysyc和支持寄存器分配的sysyc生成的程序的性能