『编译技术』SysY-Mips编译器设计——总体设计概述

本编译器源自『编译技术』课程设计，针对C语言子集`SysY`文法的源代码生成`Mips`体系架构目标代码。

2024年编译实验教程

章节目录

零. 支持文法

2023年编译技术实验文法说明

一. 总体结构

本编译器采用经典的前端-中端-后端架构，前端负责词法分析，语法分析，符号表建立，错误处理部分，中端负责生成中间代码LLVM以及中间代码优化，后端负责生成目标代码MIPS以及后端代码优化。

二. 接口设计

本编译器秉持着高内聚，低耦合的思想，将核心功能完整封装在了各个子模块，而主类Compiler仅保留调用子模块的接口，编译步骤跟随调用的接口依次为：

1. `clearAnnotation`：删除代码中的注释
2. `parseCompUnit`：语法分析，依据语法规则生成语法树，同时伴随着词法分析与生成符号表，并进行错误处理
3. `printWrongInfo`：若代码出错，则打印报错信息并暂停编译过程
4. `genLLVM`：生成中间代码LLVM，并将其逻辑结构保存在IrModule中
5. `optimizeLLVM`：进行中间代码优化，通过调节IrModule中的FLAG可随时关闭/开启优化
6. `setMipsModule`/`genMips`：将中间代码的结构依次装填入后端Mips的结构中，并生成最终的目标代码Mips

三. 文件说明

项目源码仓库

文件树如下所示：

- `FrontEnd`：前端代码包
  - `Core`：内含语法分析器与词法分析器，是前端解析的精髓
  - `Info`：存储解析中遇到的错误，单词等相关信息

    - `NonTerminal`：存储文法内各个非终结符节点
    - `SymbolTable`：存储符号表
- `MidEnd`：中端代码包
    - `IrInstructions`：存储LLVM的相关指令
    - `SymbolTable`：代码优化时用到，存储为mem2reg而重新构建了中间代码符号表
    - `Tools`：一些工具类
    - `IrCore`：内含Value, User, IrGlobal, IrFunction, IrBasicBlock, IrModule等LLVM中端架构类
- `BackEnd`：后端代码包
    - `MipsInstructions`：存储Mips的相关指令
    - `MipsCore`：完全模仿LLVM架构，构建出的Mips下的一套架构，用于将LLVM架构装配到后端，并进行Mips目标代码生成，其中MipsInstructionBlock为核心，它的作用是将一条LLVM语句翻译为多条Mips语句。
    - `Compiler`：编译器启动类

四. 各章节目录