编译原理实验报告

221240073 李恒济*

2025 年 4 月 28 日

1 完成进度

完成了所有的必做和选做任务，主要包括中间代码生成 (ir.c) 的实现.

2 编译方式

使用了课程网站上指定的 Makefile，进入 Code/文件夹后键入 **make** 即可在该文件夹下生成 parser.

3 实现细节

3.1 修改实验二的实现

本次实验中首先排查了实验二的 bug，然后在语义分析阶段多创建并维护了一个全局符号表，并为符号表条目和数组类型添加了一些属性以方便中间代码的生成.

- 修改了实验二语义分析的实现，修复了实验二中未能实现语句块作用域的 bug.
- 考虑到本次实验的**假设 4**(没有全局变量的使用，并且所有变量均不重名)，在实验二的基础上多维护了一个全局符号表 Var，与语义分析中的其他符号表不同，Var 插入时不进行同名条目的检查（基于**假设 4** 的考虑）.
- 为符号表条目添加了 isAddr 属性，用于函数形参的传址操作.
- 为数组类型添加了 width 属性，存放数组某一层元素的宽度.
- 在 field 中添加了的 offset 属性，维护结构体成员在**最近结构**的偏移量, 如 a.b.c 中 c 的 offset 存放的是相对 b 的偏移量.

*Email:221240073@smail.nju.edu.cn

3.2 中间代码生成

遍历语法树，**自顶向下**的将源代码翻译成中间代码，有一些值得一提的实现细节如下：

- 实现了一维数组的直接赋值: 在翻译 $Exp \rightarrow Exp\ ASSIGNOP\ Exp$ 时对等号两侧的 Exp 进行特判，如果都为一维基本数组（基于无需实现多维数组直接赋值和不存在结构体直接赋值的假设），则按照相对地址逐个赋值，直至所有数组元素赋值完毕或目标数组域已被填满.
- 实现了 getSize 函数用于翻译 VarDec 时分配空间，并在求分配空间大小的过程中，计算维护好结构体域中的 offset 和数组某一层元素的 width.
- 计算多维数组 or 嵌套结构体的基本地址时，巧用递归，将求址问题转化为求 $base$ 、求 $offset$ 和计算 $base + offset$ 等子问题，显著降低求解的难度.
- 使用双向循环链表维护中间代码，并在生成完所有中间代码后统一输出到目标文件，从而方便后续进行优化.

4 不足 & 可改进的地方

- 生成的中间代码有**很大**的优化空间, 存在大量重复、无用的中间代码...
- 回看实验二的代码时发现实现的不够优雅，存在很多臃肿丑陋可优化的地方.
- 建立太多符号表对空间占用太大，或许可以进行整合.

5 思考感悟

- 处理复杂任务时，可以先从 high-leval 设计、思考，**自顶向下**逐渐完善，过早扎到细节里容易深陷泥潭、寸步难行，对心态也会造成挑战.
- 对于存在多处复用的代码，一定要避免 copy-paste，将其封装成函数，否则 debug 时会出现灾难.
- 防御性编程永远不会错，不要在 assert 断言上吝啬，这能确保程序的行为符合预期，毕竟谁都不想面对 segmentation fault 陷入沉思.