实验报告

151160055 吴宇昊 地理与海洋科学学院

程序如何被编译：

在终端中输入 make 完成编译，make test1，make test2，make test3 分别完成对 test1，test2，test3 的目标代码生成，生成文件分别储存在 test 里的 out1.s，out2.s，out3.s 中

程序功能：

目标代码的生成使用了中间代码的代码链，为了方便将 lab3 中的中间代码改为双向链表
相关代码置于 objectcode.h 与 objectcode.c 里
目标代码生成的入口在 intercode.c 中
整体框架十分简单，由变量描述符，寄存器描述符，栈描述符三个数据结构以及目标代码生成，目标代码插入，变量储存和加载，选择寄存器几个函数构成
变量描述符有当前所在寄存器的位置，其代表的变量，以及 next 构成
寄存器描述符有当前寄存器所存储变量，寄存器使用时长，寄存器名字构成
栈描述符由其长度，储存变量的的使用时长，以及变量描述符数组构成
因将语义分析部分注释掉，所以可能会出现语义错误，没有在语义分析里插入 write，read 函数的描述，所以报错便将其注释掉
目标代码生成函数负责生成开头以及 read，write 函数，将寄存器的名字初始化，逐条将中间代码变为目标代码
寄存器为实验指导所给的MIPS体系结构中的寄存器，声明了全局的一维数组储存寄存器描述符
中间代码插入函数负责对中间代码进行模式的选择，没有完成对一维数组的声明，使用的相关代码生成
寄存器分配采用的是局部寄存器分配算法，没有进行基本块的划分与活跃变量的分析，虽然使用了变量描述符和寄存器描述符，实际上并没有很好的实现这两个描述符的使用，在寄存器分配的问题上只是进行了简单的操作，按照函数的结构来说大体是这样—①通过infunction和anow两个全局变量判断当前函数体是否在拥有形参的函数当中，如果在此函数当中，则将形参分配到$a0~$a3中，多于四个形参不予以考虑，并将其插入到变量描述符链表当中，返回该寄存器的位置②如果不在上述函数当中，若链表表头为空，则malloc一个新的变量描述符作为链表表头③若表头不为空，则对链表进行搜索，若与当前变量匹配，说明该变量在寄存器中，返回寄存器位置④搜索失败，则malloc一个新的变量描述符，遍历至链表末端并插入⑤如果上述操作皆失败，说明寄存器已满。我所使用的寄存器是$t0~$t9，$s0~7。首先对临时变量进行搜索，将其中使用时长最长的释放掉⑥若操作⑤失败，那么寄存器中均为局部变量或全局变量，将寄存器中使用时长最长的先保存回内存，再释放掉。⑦返回寄存器位置
Ld和St两个函数即是在函数调用时将原先寄存器中的变量描述符压到栈中，待函数调用结束

将变量描述符放回，但是其实只是在程序内部进行，在目标代码中并没有得到体现
测试结果正确，不过可能过于简单，我认为程序中的漏洞还是有很多的

总结

这次实验和前几次实验相比可能并没有那么愉快了。实验指导不如前几次实验那么详细，只有在 github 上找他人的代码借鉴。但是其他人也没有做的能像传说中的基本块划分，活跃变量分析，栈的操作一系列等等。我自己对于汇编了解的也甚少，目标代码有时候数量庞大没有像中间代码那样便于纠错。结果也只是半抄半写，但还是发现许多漏洞。这次实验还发现一个远古错误—在 lexcial.l 中将 star 写成了 plus，找了半天竟然是 lab1 的错误。Anyway 这次实验也宣告了这门课程的结束。现在只觉得脑子一片混乱。