编译原理实验报告: 目标代码生成

141270022 刘少聪

141270037 汪值

1. 实验任务

实验四任务是在词法分析、语法分析、语义分析和中间代码生成基础上将 C--源代码翻译成 MIPS32 指令序列，并在 SPIM Simulator 上运行。本次实验需要完成指令选择，寄存器选择和栈管理。

2. 编译环境

运行环境： Ubuntu14.04、gcc 4.8.4、flex 2.5.35、bison 3.0.2、SPIM 8.0

编译方式： 进入程序所在的 Code 文件夹 键入 make 执行编译过程

然后键入 ./praser + 测试文件 + 目标代码文件

3. 功能实现

目标代码生成的主要文件是 objectcode.h 和 objectcode.c 这两个文件。

在 objectcode.h 中有变量描述符表、寄存器描述符表和栈描述符表三个数据结构，其中标量描述符表用链表形式表示，寄存器描述符表和栈描述符表用数组形式描述。

在 objectcode.c 中实现了每条中间代码到目标代码的转换。

首先在 writeAllObject 函数中对寄存器描述符的初始化，打印出程序头部信息、read 函数和 write 函数的 mips 代码。然后将实验三的每条中间代码 IRList 逐条转为目标代码。

函数irToObject定义了一条中间代码到目标代码的转换规则，首先判断每条中间代码的类别，程序中实现了实验三中的14种类型：LABEL_IR、FUNCTION_IR、ASSIGN_IR、PLUS_IR、MINUS_IR、STAR_IR、DIV_IR、GOTO_IR、IF_GOTO_IR、RETURN_IR、ARG_IR、CALL_IR、READ_IR、WRITE_IR。

函数getReg实现了了寄存器分配算法——局部寄存器分配算法：如果当前代码中有变量需要使用寄存器，就从当前空闲的寄存器中选一个分配出去；如果没有空闲的寄存器，不得不将某个寄存器中的内容写回内存时，则选择那个包含本基本块内将来用不到或最久以后才用到的变量的寄存器。

程序中可以使用自由使用的寄存器共有20个：$t0至$t9和$s0至$s7可以任意使用，$v1和$fp也可以使用。

栈和函数：数据结构struct StkDescipter来进行栈管理，进行函数调用前先将返回地址$ra寄存器的内容压入到栈中，然后根据参数个数将参数$a0-$a3压入栈中，然后将所有变量保存到StkDescipter中，并且将变量压入栈中，将对应的寄存器清零，然后将变量链表VarList清零。函数调用结束后将之前保存的变量恢复到VarList中，并且恢复之前的寄存器，将StkDescipter清零。然后恢复参数寄存器$a0-$a3，恢复返回地址寄存器$ra，最后将返回值从$v0中取出即可。在函数体中，先将所有的变量列表和寄存器列表清零，然后将参数从$a0-$a3中取出来，在函数返回时，只需要将返回值移动到$v0中，并且跳转到$ra指定的地址即可。

在函数storeAllVar中，将所有变量寄存器中的内容压入到栈中，并且将变量保存到StkDescipter中，然后将变量链表VarList清零。

在函数loadAllVar中，先将变量链表清零，然后将StkDescipter中内容回复刀VarList中，并且将栈中的内容恢复到寄存器中。

4. 存在的问题

由于时间关系，程序中没有设计到局部变量的栈管理，只实现了函数调用的栈管理，因此程序只支持20个寄存器的使用，变量一多，就会出现寄存器使用冲突的bug，导致程序出现错误，因此无法实现很多变量的同时使用。

程序只支持最多4个函数参数的传递，因为只设计了简单的栈管理，无法将多余4个的参数压入栈中，因此在调用参数多于4个的函数时，会导致程序出现错误。

程序中由于没有设计好栈管理，无法支持数组变量的所有功能，因此无法支持数组。