编译原理实验报告-Lab4

Name: 傅小龙 Dept: CS

Grade: 3 ID: 191220029

一、实验名称

目标代码生成.

二、实验内容

本次实验中我完成了必做要求.

在生成目标代码时, 按序遍历中间代码, 依照中间代码的类型(例如: 赋值、四则运算、函数调用等)和操作数的取值方式生成相应的目标代码.

寄存器分配方式采用实验手册5.2.4节所描述的朴素寄存器分配算法. 具体实现思路如下:
每个函数自然地把代码分段, 依据假设4和假设5, 我们可以直接根据函数中的变量和临时变量的数量来计算应该分配给该函数的栈空间大小, 并且依据变量和临时变量出现的顺序安排它们在栈中的位置. 这里用两个数组'int* var_offset', 'int *tmp_offset'分别记录当前函数中要用到的变量和临时变量相对于栈顶'$fp'的偏移量. 两个数组的下标分别和中间代码中变量、临时变量的序号一一对应. 在生成目标代码时通过访问这两个数组即可得到操作数在栈中的位置. 相关实现详见'./Code/mipGen.c' - 'int funcStackSpace(...)', ' opStackSpace(...)', 'int findOffset(...)'.

根据朴素寄存器分配方式的特点, 我们可以让所有指令的操作数所用到的寄存器都确定下来, 然后利用sw指令将运算结果保存至栈中相应的位置.附表给出了本实验中间代码的翻译方式.

三、编译本程序

在提交文件的根目录下, './Code'文件夹内为实现实验手册中所要求完成的目标代码生成器的相关代码。请使用这个文件夹下的 Makefile 编译生成可执行对象。

在 /Code 目录下的终端输入'make'即可生成可执行目标'parser'，用形如'./parser ¡test file¿ [dstPath]'的指令生成'test file'的目标代码.'dstPath'是中间生成的目标代码文件的路径，这一参数 是可选的，缺省值为 ./output/output.ir . 注意：如果生成文件的路径采用缺省值，请确保'paser'的同目录下文件夹'/output/'是存在的，否则运行 ./parser 时将导致 Segmentation Fault .