编译原理实验四实验报告

191810140沈炫辰

一、实验环境

Ubuntu 18.04.3 LTS

VirtualBox

win10电脑

vs code、gcc、gdb

二、实现内容

我的实验四和实验三是一起交的，实验四出现的问题可能和实验三有关，所以把实验三可能出现的问题复制在下面：

实验三中有几个残留缺点没有解决，一个是程序中定义的变量使用了原名，而临时变量使用了tn，所以如果程序中定义的变量出现了tn，那么会产生重名，程序不会判断重名，就会出问题。

第二个问题是不够优化。虽然我对程序进行了多趟局部优化（包括处理局部的公共子表达式，复制传播和消除死代码），但是测试不多，可能会出现bug，而且没有全局优化。如果测试时出现的bug过多，可以在sdt.c的sdt_main函数里将优化函数optimize_intr_code注释掉。

本次实验我采取最简单的线性翻译中间代码，并且按基本块进行寄存器分配的策略。

在分配寄存器时发现可用寄存器满时，采取随机分配一个寄存器的方法。

下面是可能出现的问题：

没有区分t寄存器和s寄存器（均视为t寄存器，函数不负责保存）。t8、t9两个寄存器用作常数寄存器，其它时候不能使用（mips32中大多数指令没有两个常数作为参数的情况，所以用这两个寄存器装载常数进行指令运算）。v0用作返回值，v1不用。a0-a4只用作传参数，其它时候不用。倒数第二个寄存器用作fp函数栈底指针。

函数参数只设置了4个寄存器，没有为额外参数实现压栈。

生成代码和中间代码一样，不同的是编译时只有一趟编译，但是运行时可以运行多次（循环、递归），在多次运行中怎么保证每次代码运行都是按预期不变的是需要考虑的问题：

第一个问题是栈内存的分配。如果每次分配栈内存的时候将sp-4并分配是一定不行的，所以在第一次将栈分配给某个数据时，要记录下该数据在栈中的位置，因为sp会动，所以记录下相对于fp的位置，fp在一次函数调用中一定不会动。每次读取和写入时，也是读取或写入相对于fp的位置。

第二个问题是函数调用的参数。不知道为什么mips32要给函数参数设置寄存器，可能大部分函数都是四个参数以下，执行更有效率，但这样就有了另一个问题：在函数中调用函数会复写四个参数寄存器，所以在调用之前，要把四个寄存器中的有效数据压入栈中，并且记录好相对位置。

第三个问题是基本块的保存和读取策略与循环冲突。有一些保存读取要提出到循环外面，有一些保存读取要插入到条件判断之中，所以我还要修改中间代码，标记这个条件判断是什么。

感觉这个实验一开始还是很有自信的，做着做着就开始怀疑人生了，各种组件之间如何协调完美运行是真的难（自己想的程序逻辑很难保证不出现问题），只能不断测试找出bug然后思考修改，但是这学期我实在太忙了，最近两个实验都没怎么测试就交上去了。

三、编译方法

自己写了一个特别简单的Makefile文件，终端输入make编译，然后输入./parser 文件名 可以生成单一cmm文件的中间代码到out.ir并生成汇编代码到out.s