编译原理第四次实验报告

201220069 周心同

实验内容

将lab3的中间代码翻译成目标代码，并通过oj测试

编译方式

实验环境：与实验要求相同。

在Code目录下make即可。

程序亮点

中间代码拆分

在翻译成目标代码的过程中，如果遇到类似于 *x = *y, x = &y +z 之类的中间代码，一方面，实验指导上并没有详细指出如何翻译，另一方面，也为了翻译时候代码的简洁统一，且中间代码的debug明显要比目标代码简单的多，所以这次实验一部分精力主要放在将中间代码拆分为*x = y , x = *y, x = &y这三大类，并且把原先依附于OP的*号和&号改为依附于IR，这样可以在IR类别上统一操作。同时修改了一些lab3中的冗余结构，类型。

数据结构

一个大小为32的寄存器数组，每个寄存器保存存储内容和是否为空的标记。

两张链表，一张内存链表，一张寄存器链表（链表将寄存器和操作数对应），用朴素寄存器分配算法，首先将所有要用到的变量存到内存链表里，每条中间代码翻译就将要用到的变量存到寄存器里，翻译完后再写回内存中。如果spill，就把寄存器表头移出，把表头指向的寄存器的存储内容替换为新的内容，然后加入寄存器表尾。

遇到的问题和解决方法

1.我使用Qtspim时不论怎么用都无法读取输入，最后找到一个在线的QtSpim网址，非常好用：

JsSpim - Online MIPS32 Simulator Based on Spim (shawnzhong.github.io)

2.找到了一个lab3的bug，泪目，要不是我改了lab3的ir结构估计这辈子都找不到bug

是与ARG有关的问题，建议把lab4的某几个样例加入lab3的test，或者加一个 arg *t的test（改了之后高了30分 为什么这么多样例在lab3都没有啊），经过近半个小时的确认错误和debug，删去三行对了（貌似是lab3更改数据结构时忘了改ARG的一部分 我甚至还注释了todo 总之lab3乱得很 再也不想碰了 当时debug心惊胆战 还重交了好几次lab3）

3.又有一个lab3的bug，是变量的初始化赋值为变量的情况，谁能想到strcmp写成strcpy还能过lab3

```
if (place->kind != VARIABLE_OP || strcpy(place->u.value, field->name) != 0)
```

又重交了一遍lab3，建议lab3加一个test如 int i2 = i1;

4.如果不用朴素寄存器分配算法，每次翻译完后不写回内存的话会有奇奇怪怪的bug，比较困难，故用朴素寄存器分配解决。

5.一些其他的小bug 例如空指针，代码逻辑问题等，借用之前实验的样例来debug

实验建议和感想

1.不要有过于繁复的架构，按照实验指导书的翻译方式来，更多的去修改中间代码而不是目标代码的翻译方式，可以更好的降低难度，而且debug起来更加安心。

2.遇到段错误，二分法debug找空指针。

3.实验指导书给出的多种方法如寄存器分配方法等其实并不是重点，选一个简单的实现就行，一开始走复杂的反而浪费了很多精力。

4.遇到困难打tag，虽然对代码的数据结构模块性不好，但有利于代码书写时的整合统一，这里我将原本操作数的输出内容统一为name，不然原先那样每次用变量名都需要思考OP类型，十分麻烦。