编译原理实验四

111220174 张赫
mattzhang9@gmail.com

总体说明:

本次实验的内容是通过已经生成的中间代码输出最终对应的MIPS32指令序列（可以包含伪指令),并在 SPIM Simulator 上进行测试。

指令序列采用直接输出的方式，不加以任何优化。寄存器采用固定分配方式，堆栈管理使用$fp。因为在理论上不存在太大问题，以下所述为一些细节上的问题。

翻译细节:

1.对于函数名，不直接采用源代码中的名字，而是使用"_"加函数名。因为如果原来的函数名中有诸如add, sub等的名字会和汇编指令混淆而出错。

2.所有变量都存在栈上，使用链表记录变量名和其相对本函数$fp的位置以达到记录所有变量位置的目的。每次使用变量的时候通过查表得到相对地址，根据 $fp间接寻址即可找到。对于数组类型变量，考虑了多重形态以及因为中间代码优化得较多而出现的各种情况。在赋值、双目运算、参数传递和条件语句都做了足够的考虑以及细分。数组部分占用了绝大部分时间。对于常量型操作符，也经过了大量考虑并进行了大量测试。

3.堆栈管理完全使用不会改变的$fp。因为使用固定寄存器，所以不存在保存寄存器数据的问题。除了打印函数名以及移动没有被使用的$sp外，其他工作均有调用者完成。调用者先用栈保存当前理论上$sp的位置（有一个专门的变量记录现在栈顶到$fp的位置），这样一来函数返回后可以直接回到原来的栈顶而不需要修改。然后调用者对参数进行压栈。

参数在ARG的时候通过stack记录，在此时全部pop出。这样做是因为有些参数可能是一些运算，所有不是全部ARG连在一起。再然后调用者保存当前$fp和$ra的值，改变$fp的值为现在的栈顶，在此之后调用函数。函数返回后，恢复$fp, $ra以及理论上$sp的值，移出$v0中的返回值并赋给结果。

4.为了方便调试、进一步学习等原因，还在机器码的响应位置打印了中间代码。

总结:

本次实验花了大概5天时间，在经历过今年以及去年Lab3的测试用例的测试后终于完成了。这次实验因为没有采用其他寄存器分配方法，所以在理论上没有什么难点。但是在写代码的时候却遇到了很大困难。最开始一直没想好该用什么方式完成堆栈，但在看了gcc的汇编代码、查阅SPIM手册和MIPS官方指南后决定使用$fp后还是解决了。然后就一直纠结于数组的问题。数组可能出现的形式很多，每种的翻译模板都不一样。即使这些在赋值的情况下都能弄对，但其他情况下可能因为考虑得不全而

漏掉（比如操作符的类型为常量）。有些错误并不会使模拟器直接报错，再加上模拟器字非常小而且不会附上注释，所以调试异常艰难。

本次实验也是这个学期编译原理实验的最后一个实验，虽然付出了比较多的时间以及充满对其他班先发测试用例各种水过的羡慕嫉妒恨，但无论是从对编译原理本身的理解的角度还是从写代码的角度都有比较大的提升，一切辛苦还是值得的。

使用方法:

编译`parser`请先进入source文件夹。

输入 `make parser` 来编译 parser。

然后输入 `./parser 输入文件 输出文件` 以获得MIPS32指令序列并存储在输出文件中。