编译原理实验四-实验报告

141220041 何宇豪 [141220041@smail.nju.edu.cn](mailto:141220041@smail.nju.edu.cn)

141220037 杭 诚 [141220037@smail.nju.edu.cn](mailto:141220037@smail.nju.edu.cn)

1. 程序实现的功能

首先，我们的程序目前可以把上次得到的中间代码文件转换成汇编伪指令，从而在SPIM模拟器上运行。

1. 编译方法

只用在Code文件夹下，输入make, 即可得到parser可执行文件，假设测试文件叫test.cmm，运行./parser test.cmm out.s可以得到最终的汇编代码out.s。运行命令spim -file out.s可以看到模拟器上的运行结果。

1. 实现思路
2. 总体框架：

我们是基于线性中间代码表示（IR）来翻译最后的目标代码，具体来讲，就是逐条从中间代码翻译到目标代码，每次指令执行都需要确定操作数的位置，确保操作数在寄存器中，然后再做运算。

1. 指令选择：

指令选择我们基本上是采用的一对一或一对多的方式来翻译，一对一是对一些简单的指令的翻译，比如LABEL main:语句，汇编代码中可以直接翻译成main: ；一对多是指一些稍有复杂的代码，比如

x = \*y ,可能需要翻译成三条指令：lw $t2, y; lw $t1, 0(%$t2); st $t1, x;。

1. 寄存器选择：

寄存器我们是写了几个辅助函数，用来为中间代码的操作数分配寄存器。当需要从内存载入操作数op到寄存器reg中时，使用get\_var(Operand op, int reg)这个函数；当需要把寄存器的内容存储到操作数中的时候，使用store\_var(Operand op, int reg)这个函数。

除此之外，我们是把代码按块划分，每一个代码块的开头使用get\_var把需要的参数载入到寄存器，然后每个代码块的结尾使用store\_var把所有的寄存器中的运算结果存回内存中去。

1. 栈的实现：