编译原理TEST虚拟机实验报告

专业班级： 计算机科学与技术2019-3班 日期： 2021年12月7日

|  |  |  |
| --- | --- | --- |
| 小组成员 | 选课序号 | 学号 |
| 胡聿鑫 | 19 | 2220192813 |
| 尹潇怡 | 14 | 2220192462 |
| 郑永利 | 27 | 2220193811 |
| 韩蓉 | 4 | 2220191303 |
| 牛天蕾 | 25 | 2220193392 |

# 实验任务分配情况

胡聿鑫同学负责实现if-else，while和for语句的相关功能，实现了基本的加减乘除功能，同时整合了词法、语法和语义分析程序，并能输出简单的运行栈。

韩蓉和郑永利同学设计可识别的中间代码规则，实现了函数调用功能和键盘输入read和结果输出write，并扩充实现了do-while语句功能，能完成循环嵌套。

尹潇怡同学和牛天蕾同学编写了多个TEST源程序测试用例，覆盖了程序的所有功能，同时负责整理实验所实现的功能，总结程序的优缺点并攥写实验报告。

# 实验目的

设计一个TEST虚拟机，要求读取中间代码并执行，同时可以利用键盘输入，并输出正确的运行结果。

# 实验内容及要求

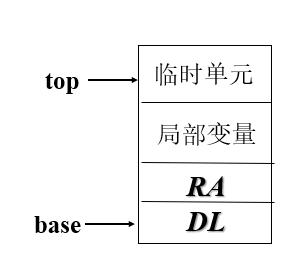
（1）能够完成四则运算；

（2）能够实现函数调用，if-else，while和for语句的相关功能。

# 实验相关理论知识

由于实验过程中需要用操作数栈实现程序的输入和输出，因此首先给出操作数栈的原理：操作数栈用于记录程序运行过程中每个变量的值的变化，也能用来对给定条件进行判断。操作数栈和普通栈相同，所有的操作均在栈顶完成。操作数栈中，以函数调用为单位分配空间，该空间用于存放局部变量、基地址和返回的位置等。每进入一个函数，都需要为其分配空间，退出时则释放开辟的空间，即使是同一函数先后进行两次调用，分配的空间也是不同的。需要开辟空间的具体大小在语义分析阶段确定。

函数数据区具体包括的内容有：局部变量、临时单元，两个联系单元DL和RA（其中DL用于存放主调函数基地址，RA存放函数调用指令的下一条指令地址），其布局如下图所示：



由于指令有很多种，在识别每一个指令时可以使用if或者switch语句进行判断。但switch语句每次只取一次变量和条件进行比较，而if语句中每次都会取出所有变量和条件进行比较，因此从程序复杂度的角度出发，选择switch语句能有更高的效率。而且虚拟机需要识别的指令较多，因此有更多的分支，switch能够在一定程度上对程序进行简化，使代码的易读性更高。

为了能确定下一条需要执行的指令所在，可以借助程序计数器ip，用来确定下一条指令是第几个，即按顺序读取中间代码的指令，每读取一条指令，ip就加一，若遇到跳转类指令（BR、BRF等），将对应的操作数直接赋值给ip，然后再继续按顺序执行。

TEST虚拟机整体实现过程的流程图如下：



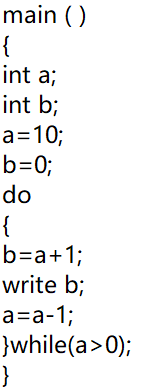
# 实验过程、运行结果

虚拟机的执行过程，实质上就是不断读取中间代码并识别指令，因此本实验的关键就是把每一个指令对应的功能实现，并且能够确定接下来执行的指令所在位置。因此可以得到实验的整体设计思路：逐条读取中间代码，每个指令作为一个分支，识别到某一指令名便执行对应的功能，执行完毕后继续读取代码直至全部读取结束。

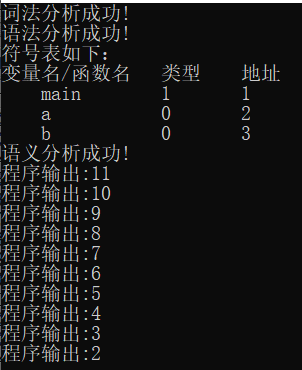
读取中间代码时，采用一个结构体数组code用于存储指令及其操作数，以便程序能够连续执行至结束。由于经过了之前的词法、语法和语义分析阶段，中间代码生成后已经不会出错，因此无需考虑执行中间代码的过程中会遇到错误。为方便操作，执行时采用操作数栈stack。同时以栈顶寄存器top作为栈顶的下一个单元的下标，基址寄存器base为当前函数的数据区在stack中的起始地址。

为了方便调试，将词法、语法和语义分析程序以头文件的形式在虚拟机程序中引用，生成中间代码后直接解释执行。这样无需在语义分析结束后，将中间代码作为输入文件再次输入到虚拟机程序中。同时也能直观地看到词法、语法和语义分析结果，包括识别单词类别，输出语法树，打印符号表和生成中间代码等。

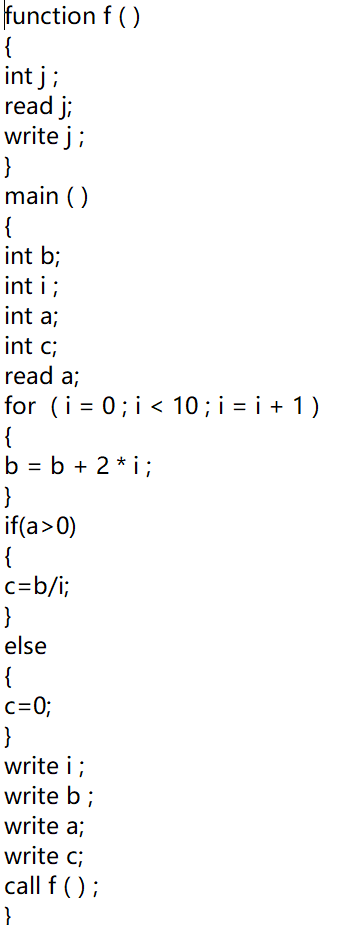
该程序在实现基本功能的基础上，拓展实现了do-while语句，现编写一个简单的TEST程序进行验证：



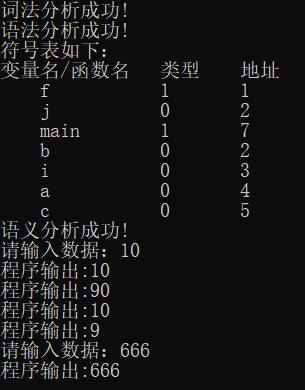
该程序每循环一次均输出一次b的值，a的初值为10因此b第一次输出结果应为11，每次输出递减1，最后输出b=2时程序结束，最终运行后结果如下：



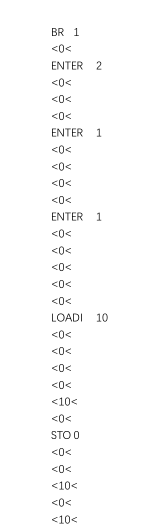
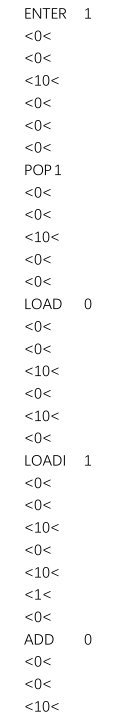
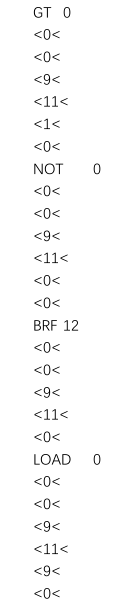
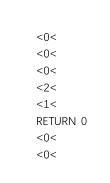
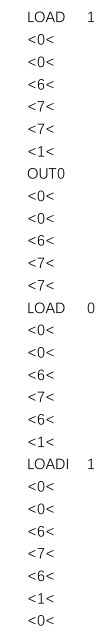
此代码仅简单验证了该虚拟机是可以运行的，而四则运算、函数调用、循环嵌套等功能并没有在程序中体现，为较为完整地测试本虚拟机的功能，现编写如下TEST源程序：



该程序中涵盖了虚拟机可以实现的全部功能，首先进行词法和语法分析，然后进行语义分析并输出符号表，再生成中间代码。虚拟机直接开始解释执行中间代码，无需再进一步输入中间代码的源文件。该程序首先执行一次键盘输入，然后开始进入for循环，输出时首先输出循环次数i（为10次），然后输出循环结果B（值为90），接着进入if-else判断，输入的数大于0则执行c=b/i，否则c=0，由于输入10，因此输出c=90/10=9。最后，程序还要执行一次函数调用，该函数实现功能为输入一个数，紧接着输出刚刚输入的数，运行后结果如下：



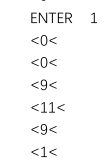
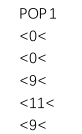
书中借助运行栈帮助理解代码的执行过程，为此在虚拟机程序中也尝试引入了运行栈，由于画出栈的过程比较繁琐，本程序对运行栈进行了简化，只对能修改栈顶元素的操作进行输出，若没有赋值的变量入栈则默认将其置为0，上述TEST程序得到的部分运行栈如下：

# 实验总结

本次实验是小组合作完成的，组内成员分工明确，从规则设计、函数编写、程序整合到报告攥写，大家都各司其职，完成地既高效且准确，因此实现过程并没有太大困难。总体上实现了TEST虚拟机的基本功能。

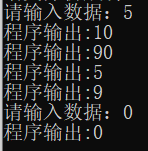
此外，程序也在功能上进行了一定的拓展，首先能够实现do-while循环的功能，也可以和for，while循环并用，实现循环的嵌套。虚拟机本没有输出任何文件，但为了便于理解中间代码的执行过程，程序中扩充了运行栈。不同于书上的运行栈，该栈将对应的中间代码显示，并紧接着输出执行完该指令后栈的当前状态。以下列指令为例，进入函数体后执行POP操作，将栈顶元素1出栈（为了简化代码，输出时栈顶元素优先输出，因而输出文件中底部为实际的栈顶）：

该虚拟机并未涉及错误处理的部分，因为程序已经经过词法、语法和语义分析成功生成了中间代码，执行中间代码正常来说不会出现错误。但这并不意味着虚拟机程序在运行时不会出错，若TEST程序中存在一些错误是之前的程序无法识别的，那么它仍然能够成功生成中间代码，但是却无法通过虚拟机的编译，这也是本次实验的不足之处，今后还需要不断地优化改进以克服这一问题。

该程序仍然有不足之处，未能实现switch-case语句的有关功能。针对这一点，应当在词法分析阶段就将switch和case添加到关键字，在语法分析阶段按switch的结构读{}，case语句后冒号前的数字或字符串被识别成标号，每一个case语句必须检查是否有break，其相应的语义错误也需要进行声明。

程序的可读性也有待加强，例如如下情况：



该页面有多个输入和输出，但格式均相同，若不结合TEST源程序很难判断结果正确与否，若能够在输入和输出时同时给出需要输入和输出的变量名，则可以使得程序更加清晰易读。但本虚拟机采用了操作数栈，对数据的read和write均在栈顶完成，因此若想在栈顶数据读入或写出之前再给出对应的变量名，则需要将变量名的相对地址提前存入另一个栈中并读出，然后才能再一次访问栈执行对应的read或write操作，因而虚拟机函数中每个case分支都需要调用两个栈。若想分时调用两个不同的栈，首先需要确定栈顶元素之间的对应关系，并根据这一关系，当匹配时输出变量名和结果，不匹配则跳过。不仅如此，还需要修改栈的结构，让其能够输出字符串到屏幕，否则栈只能将数字输出，这一功能在本实验中没有实现，今后还需要进一步提升自身代码能力，不断完善相关知识的学习。