武汉大学计算机学院

本科生实验报告

高级程序设计大作业环节

专 业 名 称 ：计算机科学与技术

课 程 名 称 ：高级程序设计

指 导 教 师 ：梁意文 教授

学 生 学 号 ：2016301040097

学 生 姓 名 ：左亦平

二○一八年四月

摘 要

实现一个基础的lua解释器

关键词：lua

目录

[1实验目的和意义 2](#_Toc513814996)

[1.1 实验目的 2](#_Toc513814997)

[2 实验设计 2](#_Toc513814998)

[2.1 概述 2](#_Toc513814999)

[2.2 实验环境 2](#_Toc513815000)

[2.3 设计思路 3](#_Toc513815001)

[2.4代码规范 4](#_Toc513815002)

[2.5 代码文件结构 4](#_Toc513815003)

[2.6 代码阅读指南 4](#_Toc513815004)

[参考文献 9](#_Toc513815005)

[附录 10](#_Toc513815006)

[教师评语评分 11](#_Toc513815007)

# 1实验目的和意义

## 实验目的

实现一个基础的lua解释器，包含以下lua的特性

1. 赋值语句
2. 表达式求值
3. 函数声明和调用

# 2 实验设计

## 2.1 概述

实验分为两个阶段

1. python原型开发
2. C#移植

其中，python原型开发阶段主要是学习：虚拟机基础，编译原理基础，lua基础语法，lua实现，lua源代码。然后用python按照自己的理解实现一下部分基础的功能，软件的主要执行流程：

1. 解析lua代码文本
2. 生成汇编代码
3. 用虚拟机执行，实现赋值语句，表达式求值，函数声明和调用

移植到C#后，主要是解决强类型语言下的lua数据模型（data model）的实现，其他部分基本是python语法到C#语法的手工翻译。顺带发现了一些用python开发时没发现的错误。

## 2.2 实验环境

Pycharm

Sublime Text3

Python 3.6

Visual Studio 2017

.Net Framework 4.5

C# 7

Antlr4 Runtime（C# 和python）

Antlr4

## 2.3 设计思路

python阶段

1. 解释器前端是antlr4，实现listener接口来生成汇编代码，存储在main\_func对象中
   1. 指令集设计和虚拟机是紧密相关的，由于一些原因，最后选择了基于栈的虚拟机实现，这样实现变得相当简单
2. main\_func对象包装成thread对象后交给虚拟机执行
   1. 虚拟机的基本流程是在一个大循环中：取指，译码，执行
   2. 使用访问者设计模式后取指和译码一行就解决了，所以主要的部分是执行
   3. 执行涉及运行时环境的设计，对于虚拟机而言，只需要一个栈即可，即一个stack<function>对象，每一个元素都是一个栈帧，存储了当前执行的函数的局部信息等

C#阶段

数据模型的实现

1. 实现动态类型一般使用C union，C#没有union，使用多态实现，混合值类型和引用类型的union本应该用装箱拆箱实现，由于性能太差而且语言的类型是非常关键的，所以混合了额外字段和多态，这样的缺点是设计变得更加复杂
2. 最终实现为一个lua\_Tvalue类，封装了对lua类型的操作，访问等
3. 软件写到一半发现设计的仍然不是很好，导致访问的API复杂（加入类型转换的API会更复杂），暂时不重构，所以不详细说明数据模型

## 2.4代码规范

1. 变量蛇形小写（没有完全遵守，太复杂了）
2. 坚持写英文注释（没有完全遵守，因为软件目前是半成品）

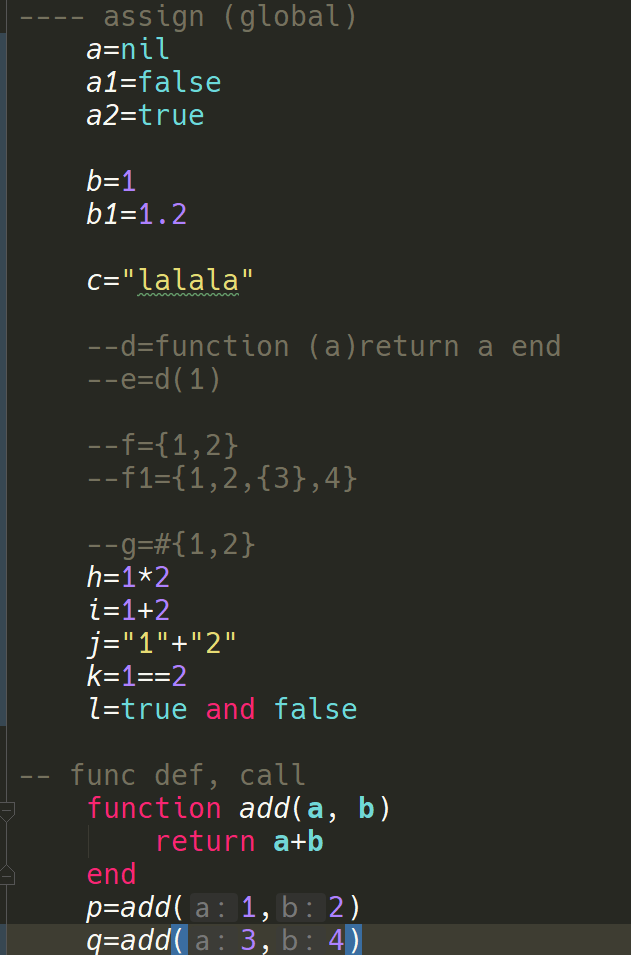
## 2.5 代码文件结构

|  |  |
| --- | --- |
| 文件名 | 功能描述 |
| lvm | 虚拟机 |
| lua | lua对外接口：类型和dofile |
| ISA | 指令集和实现 |
| lobject | lua数据模型实现 |
| Program | 软件main入口，以执行test.lua为例 |
| compiler | 解释器前端，生成汇编代码 |

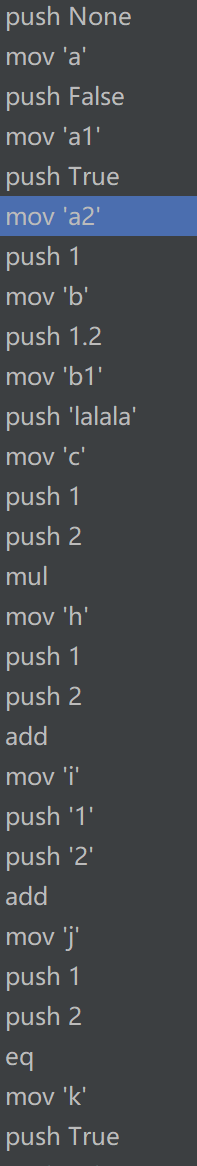
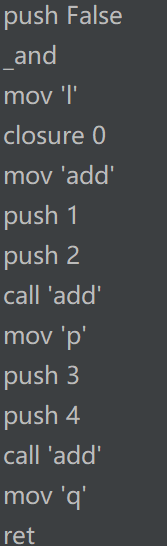
## 2.6 代码阅读指南

无

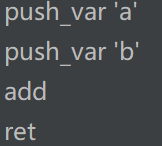
结论

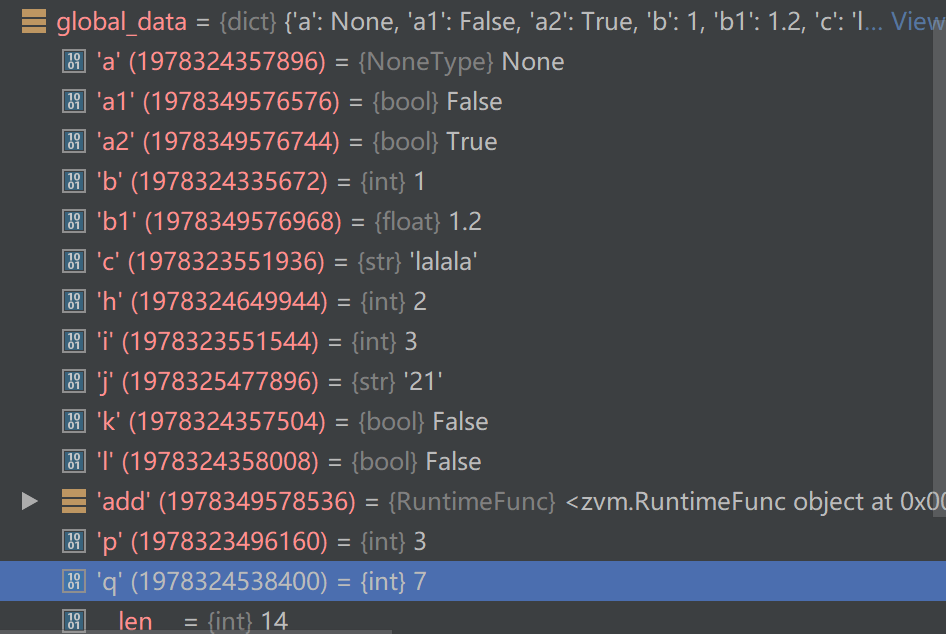
test.lua

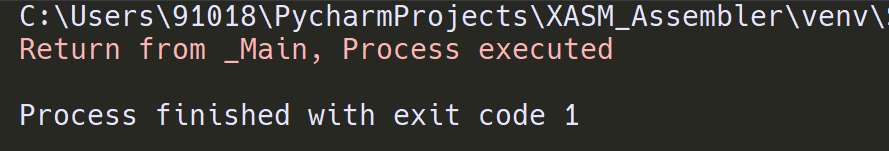
main函数汇编代码



add函数汇编代码



执行后全局变量表

直接运行后

C#版本效果是一样的，不重复了

结论：

成功实现

接下来做什么：

其实只实现了很小的一部分，概括地说就是实现了一个动态类型的脚本语言，它能进行表达式计算和函数调用。软件仍然是半成品，继续写的话会发现更多的问题。此外，编译原理和lua语言的实现确实是很难的主题，很多内容自己现在的理解大概是错的，有待进一步学习。

# 参考文献

1. implementation of lua 5
2. A No Frill s Intro To Lua 51 VM Instructions
3. the evolution of lua
4. lua设计与实现
5. definitive antlr4 reference
6. lua reference manual

# 教师评语评分

评语：

评分：

评阅人：

年 月 日