Atitit.虚拟机与指令系统的设计

[1. 两种计算模型 ，堆栈机和状态机（基于寄存器的虚拟机 1](#_Toc501)

[1.1.1. 堆栈机 1](#_Toc10760)

[1.1.2. 状态机 2](#_Toc22547)

[2. 为什么状态机比堆栈机快呢？ 3](#_Toc10711)

[2.1. Stack based vm的指令 范例 4](#_Toc29903)

[3. 参考 5](#_Toc5965)

# **两种计算模型** ，堆栈机和状态机（基于寄存器的虚拟机

有了上面的基础只是，我们就知道，堆栈机和状态机不过是两种不同的图灵完整的计算模型而已。

### 堆栈机

所谓堆栈机，就是计算机的状态是存在于堆栈之中，通过对堆栈中的元素进行运算和调整，来实现计算功能的计算机。   
例如，要进行一个1+2的加法运算，那么就：

| 操作 | 堆栈状态 |
| --- | --- |
| 初始状态 |  |
| 将1压入栈中 | 1 |
| 将2压入栈中 | 1, 2 |
| 调用加法运算 | 3 |

其典型代表就是Python的虚拟机，代码如下：

**push** 1**push** 2

Add

作者:: 绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ） 汉字名：艾龙，  EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

### 状态机

状态机的基本原理就在于，它可以有有限种状态，指令能够让它在不同的状态之间进行转换。   
听起来很抽象？   
但其实，我们大部分写代码的时候都是对状态机编程，比如c代码：

**int** a = 1;**int** b = 2;**int** c = a + b;

其实这个状态机有2^96种状态（假设int是32位的），因为变量a有2^32种状态（-2147483648~2147483647），b、c亦然。

|  |  |  |  |
| --- | --- | --- | --- |
| 操作 | a的状态 | b的状态 | c的状态 |
| 初始状态 | 0 | 0 | 0 |
| a=1 | 1 | 0 | 0 |
| b=2 | 1 | 2 | 0 |
| c=a+b | 1 | 2 | 3 |

典型代表就是Lua的虚拟机，应的代码就是：

loadk 0 1

loadk 1 2

add 2 0 1

意思就是：

register[0] = 1register[1] = 2register[2] = register[0] + register[1]

# 为什么状态机比堆栈机快呢？

既然他们是图灵等价的，那大家一定会很疑惑，为何状态机比堆栈机快呢？   
那么我们要深入到虚拟机内部，看看这些指令都是怎么实现的。   
为了便于大家理解，我所有的代码都不是vm中的实际代码，而是伪代码。   
首先来看看堆栈机：

**switch**(op) {**case** PUSH:

STACK\_ADJ(1);

STACK\_TOP = oprand;

**break**;**case** ADD:

STACK\_SECOND = STACK\_TOP + STACK\_SECOND;

STACK\_ADJ(-1);

**break**;

}

我们可以看到，大部分情况下，执行一条指令，除了原始的赋值操作外，还需要调整堆栈的栈顶指针（那些STACK\_ADJ宏定义），再看看状态机的实现：

**switch**(op) {**case** LOADK:

REGISTER[oprand0] = oprand1;

**break**;**case** ADD:

REGISTER[oprand0] = REGISTER[oprand1] + REGISTER[oprand2];

**break**;

}

大家可以看到，在执行大部分指令时，状态机虚拟机会比堆栈机要少一次调整堆栈的操作，这对性能会有很明显的影响。   
当然这也主要适用于Interpreting的情况，在Jit的情况下，会有很多深度优化，从而使得堆栈机的性能也能和状态机一样。

## Stack based vm的指令 范例

一般都是在当前stack中获取和保存操作数的。比如一个简单的加法赋值运算:a=b+c，对于stack based vm，一般会被转化成如下的指令:

**[plain]** [view plain](http://blog.csdn.net/yuanlin2008/article/details/8423951" \o "view plain) [copy](http://blog.csdn.net/yuanlin2008/article/details/8423951" \o "copy" \t "http://blog.csdn.net/yuanlin2008/article/details/_blank) [print](http://blog.csdn.net/yuanlin2008/article/details/8423951" \o "print" \t "http://blog.csdn.net/yuanlin2008/article/details/_blank)[?](http://blog.csdn.net/yuanlin2008/article/details/8423951" \o "?)

1. push b; // 将变量b的值压入stack
2. push c; // 将变量c的值压入stack
3. add;    // 将stack顶部的两个值弹出后相加，将结果压入stack
4. mov a;  // 将stack顶部结果放到a中

# 参考

高效动态语言虚拟机的设计（二） – 堆栈机vs状态机 - 推酷.htm