**1.1 JVM字节码指令**

JVM指令主要包含了一下几种类型：

加载和存储指令、运算指令、类型转换指令、对象创建与访问指令、操作数 栈管理指令、控制转移指令、方法调用和返回指令、异常处理指令、同步指令等。

## 1.1.1基于栈的解释器执行过程

下面看一下一个简单的代码片段，如下所示：



用JavaP工具 执行命令 Javap -c -s -v -l StackTest.class



## 1.1.2 下面来结婚的例子说明一下整个方法的执行过程：

婚纱公司当日计划

婚纱公司

新郎

新娘

新郎出场

等候室

新娘入洞房

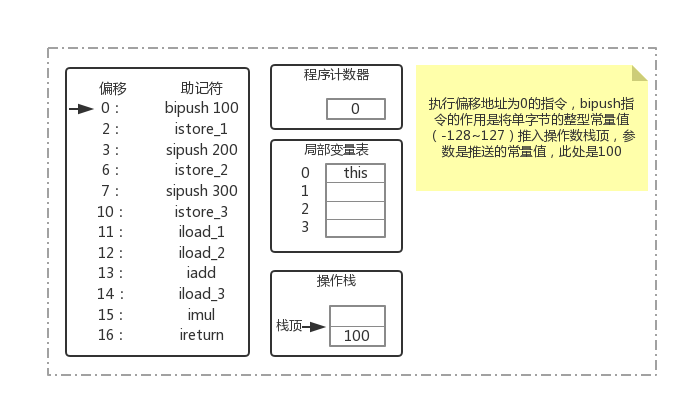
新娘出场

洞房

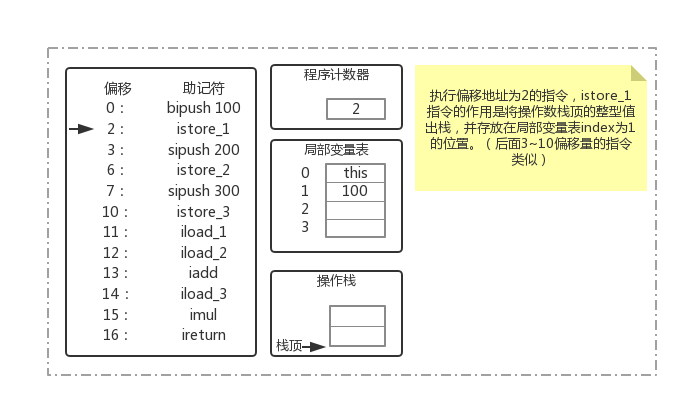
# 1.2 Class字节码中方法执行过程

下面来具体的说明一下整个方法的执行过程：

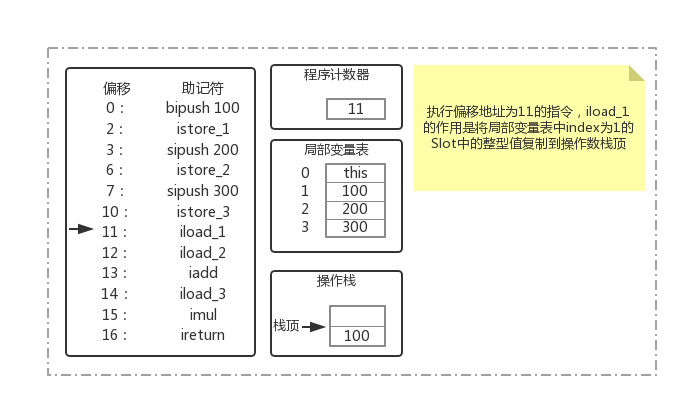
## 1.2.1 将整形常量压入栈顶



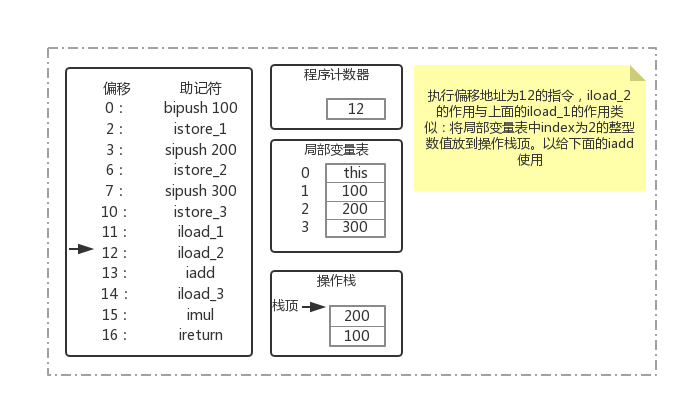
## 1.2.2 出栈并储存在局部变量表中



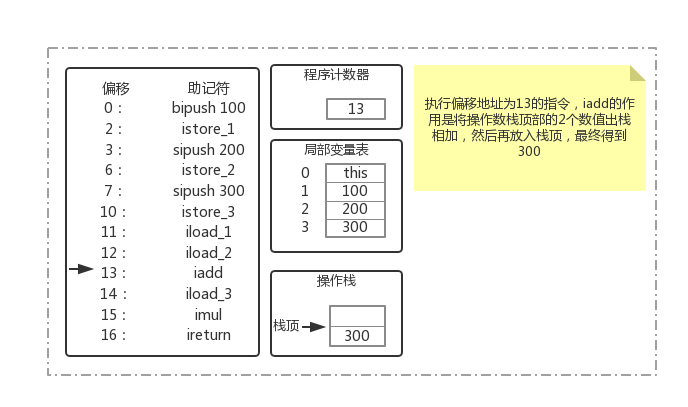
## 1.2.3 将局部变量表中索引为1 入栈



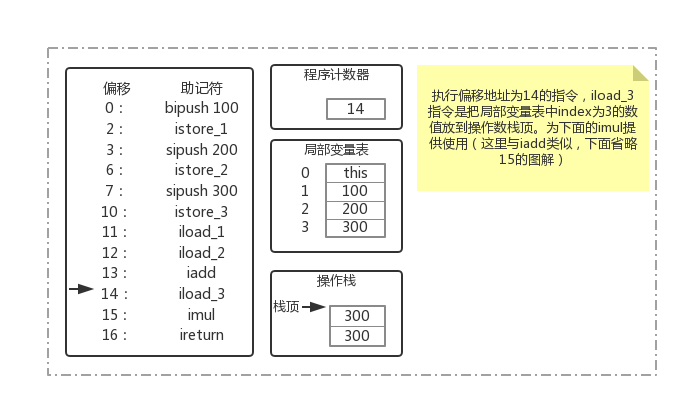
## 1.2.4 将局部变量表中索引为2 入栈



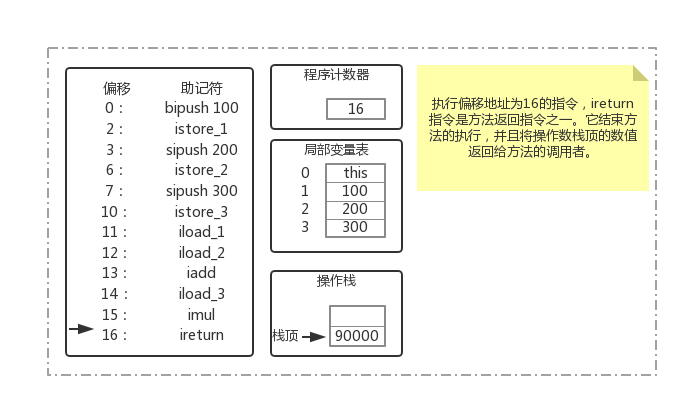
## 1.2.5 将栈中两个数字相加 class指令 iadd



## 1.2.6 加载操作数3 进行入栈操作



## 1.2.7 返回操作操作数



上面的指令执行过程只是一个概念模型，JVM会对过程做一些优化来提高性能，JVM在实际运行时可能执行过程差距比较大，并且不同虚拟机的执行也不尽相同。

# 1.3 指令说明

## 1.3.1加载和存储指令

**定义:**加载和存储指令用于数据在栈帧中的局部变量表和操作数栈之间的来回传输。

        将一个局部变量加载到操作数栈： iload\_、 lload\_ fload\_、dload、aload

。

        将一个数值从**操作数栈存**储到**局部变量**表：

istore\_、 lstore\_、 fstore\_、 dstore\_ 、astore\_。

        将一个常量加载到操作数栈：

bipush、sipush、ldc、ldc\_w、ldc2\_w、aconst\_null、iconst\_ml、iconst\_、lconst\_、fconst\_、dconst\_。

        扩充局部变量表的访问索引的指令：wide。

## 1.3.2运算指令

        运算指令作用于操作数栈上面的2个值的特定运算，并且把**结果重新存入操作数栈顶**。大体上可以分为2类：对**整型、浮点型数值**运算。

因为JVM指令集中没有byte、short、char和boolean类型的算术运算，所以都使用了对应的int类型的指令代替。

        加法指令：iadd、ladd、fadd、dadd

        减法指令：isub、lsub、fsub、dsub

        乘法指令：imul、lmul、fmul、dmul

        除法指令：idiv、ldiv、fdiv、ddiv

        求余指令：irem、lrem、frem、drem

        取反指令：ineg、lneg、fneg、dneg

        位移指令：ishl、ishr、iushr、lshl、lshr、lushr

        按位或指令：ior、lor

        按位与指令：iand、land

        按位异或指令：ixor、lxor

        局部变量自增指令：iinc

        比较指令：dcmpg、dcmpl、fcmpg、fcmpl、lcmp

## 1.3.3类型转换指令

       类型转换指令可以将2种不同类型的数值相互转换，

这些转换一般实现于代码中的显示类型转换，

主要有以下类型：

**int类型**到long、float或者double类型

**long类型**到float、double类型

**float类型**到double类型

        对于显示的类型转换，一般情况下都是窄化类型转换（也就是丢失精度的转化，如：long转为int等）。常见的转换指令有：

**i2b、i2c、i2s、l2i、f2i、f2l、d2i、d2l、d2f**

等。

## 1.3.3对象创建于访问指令

        对于普通对象和数组的创建，JVM分别使用了**不同的指令**去处理。

        创建普通对象的指令：**new**

        创建数组的指令：**newarray、anewarray、multianewarray**

        访问类变量（static类型）和实例变量（非static类型）的指令：

getstatic、putstatic、getfield、putfield

        把一个数组加载到操作数栈的指令：baload、caload、saload、iaload、laload、faload、daload、aaload

        将一个操作数栈的值存储到数组元素中的指令：bastore、castore、sastore、iastore、fastore、dastore、aastore

        取数组长度的指令：arraylength

        检查普通对象类型的指令：instanceof、checkcast

## 1.3.4操作数栈管理指令

        如同一个普通的堆栈一样，JVM提供了直接操作操作数栈的指令。

        将操作数栈顶的1个或2个元素出栈：pop1、pop2

        复制栈顶1个或2个元素，并将副本的1份或者2份重新入栈：dup、dup2、dup\_x1、dup2\_x1、dup\_x2、dup2\_x2

        将栈顶的两个数值互换：**swap**

## 1.3.3控制转移指令

        控制转移指令可以让JVM，跳转到指定的偏移地址的字节码执行。从上面的模型图看来，就是修改程序计数器的值。

        分支条件：**ifeq、iflt、ifle、ifne、ifgt、ifge、ifnull、ifnonnull、if\_icmpeq、if\_icmpne、if\_icmplt、if\_icmpgt、if\_icmple、if\_icmpge、if\_acmpeq、if\_acmpne。**

        复合条件分支：**tableswitch、lookupswitch**

        无条件分支：**goto、goto\_w、jsr、jsr\_w、ret。**

## 1.3.5方法调用和返回指令

方法调用和返回指令

        方法调用包含了以下指令。

        invokevirtual指令：用于调用对象的实例方法，根据对象的实际类型分派（虚方法分派）。

        invokeinterface指令：用于调用接口方法，它会在运行时搜索一个实现这个接口的对象，找出合适的方法调用。

        invokespecial指令：用于调用一些需要特殊处理的实例方法，包括初始化方法、私有方法、父类方法。

        invokestatic指令：用于调用类方法（static方法）。

        invokedynamic指令：用于在运行时动态解析出调用点限定符所引用的方法，并执行。

        上述的前4条指令都是固化在JVM内部的，invokedynamic的分派逻辑是由用户所设定的引导方法决定的。

        方法的调用指令与数据类型无关，而方法的返回指令是根据返回值区分的。包括：ireturn（当返回值是boolean、byte、char、short、int）、lreturn、freturn、dreturn和areturn。return指令提供给：返回值为void的指令、实例方法初始化、接口类方法初始化。

## 1.3.6异常处理指令

        Java程序中显示抛出异常的操作都是由**athrow**指令实现的。

## 1.3.7同步指令

        JVM可以支持方法级的同步和方法内的同步，这两种同步结构都是由管程（Monitor）来实现的。

        方法级的同步是隐式的，无需通过字节码指令来控制。JVM可以从方法常量池的方法表结构中的ACC\_SYNCHRONIZED访问标志，得到其是否为同步方法。

        对于方法中的同步块，JVM中使用monitorenter和monitorexit两条指令来支持。下面



