你真的了解 i++, ++i 和 i+++++i 以及 i+++i++ 吗？

原创： 帅地 [苦逼的码农](javascript:void(0);) 昨天



本文字数：2355字

阅读本文大概需要：6 分钟

我想大部分都知道 i++ 和 ++i的区别，i++ 就是先拿i来使用，之后再自增加1，而++i则是先自增加1，在拿i来使用，例如对于下面这两个语句，我敢保证大部分人都会做：

int i = 1;  
System.out.println(i++)

int i = 1;  
System.out.println(++1)

答案分别为 1,2。对于这个答案我猜大多数人都能答出来。不过 i++ 和 ++i 这两个操作，在内部是如何实现的呢？

我们先来看另外一个问题：

public static void main(String[] args) {  
   int i = 1;  
   System.out.println(i+++i++);  
   System.out.println(i);  
}

这个比刚才那个难了点，答案分别是3，3。假如你对这个答案的由来了如指掌，那么你大不可必往下看，假如你不大理解或者想从底层的汇编指令的来了解这个操作，那么你可以看看我的解释。

首先我们先来看看 i++ 的题，主要是为了后面好解释点。

int i = 1;  
System.out.println(i++);

这两行代码的部分汇编指令如下，注意，我只列出了几个重点的汇编语句：

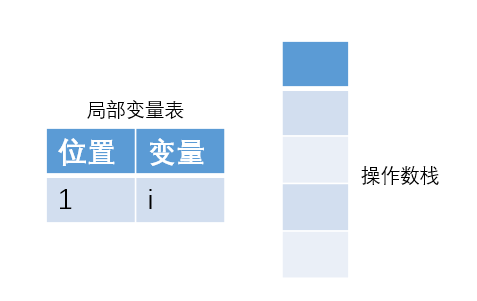
ICONST\_1 *//把常量 1 加载到栈顶*  
ISTORE 1 *//把栈顶的元素弹出，并赋值给局部变量表中位置为“1”的变量，此时指变量i。这两句就相当于 int i = 1;*  
  
*//接下来执行第二行代码*  
ILOAD 1  *//把局部变量表中位置为“1”的变量加载到栈顶，即把i的值加载到栈顶*  
IINC 1 1  *//直接把局部变量表中位置为“1”的变量加1，即把 i 加1。注意，这条指令并没有修改操作数栈就把 i 加1了。*  
INVOKEVIRTUAL java/io/PrintStream.println (I)V  *//把栈顶的元素打印出来，此时栈顶的元素是 1。所以打印的是 1*

***注：可以左右拉动***

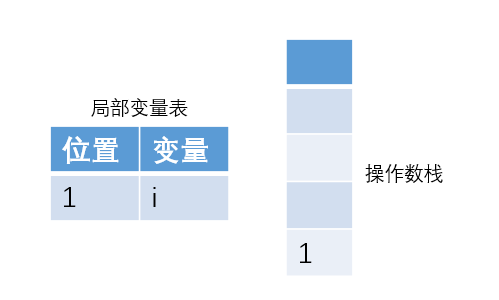
所以，此时打印的是1。

有些人可能没弄过汇编会有点蒙蔽，没事，我花个时间画个图来模拟(注：省略很多细节)。

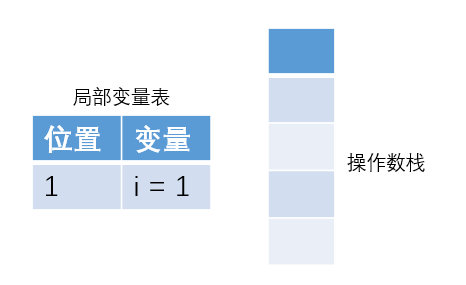
刚开始时的局部变量表和操作数栈如图所示：



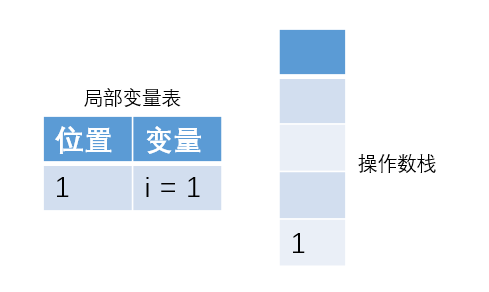
1、执行  ICONST\_1，常量 1 进栈



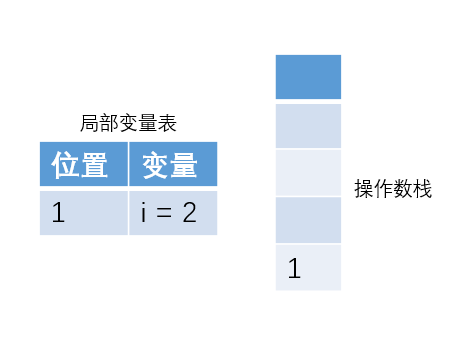
2、执行 ISTORE 1，栈顶元素出栈存到位置“1”



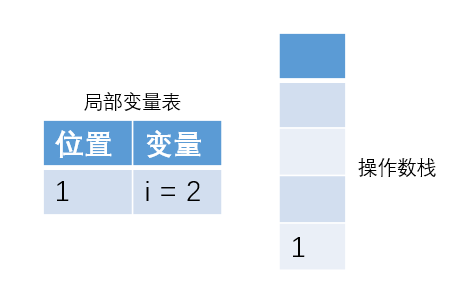
3、执行  ILOAD 1，把位置“1”的变量值存到栈顶



4、执行 IINC 1 1 ，直接把局部变量表中位置为“1”的变量加 1



5、执行 INVOKEVIRTUAL java/io/PrintStream.println (I)V  ，把栈顶的元素打印出来，此时栈顶的元素是 1.



所以虽然i已经等于2了，但此时栈顶的元素却是i之前的值 1 ，所以打印的是1。

这下关于 i ++ 的懂了吧？

那我们来看看 ++ i 与  i ++  的汇编指令有什么不同。

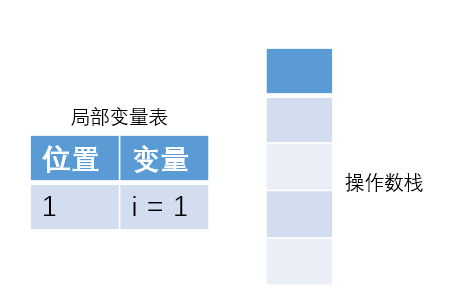
int i = 1;  
System.out.println(++i);

对应的部分重点汇编指令如下：

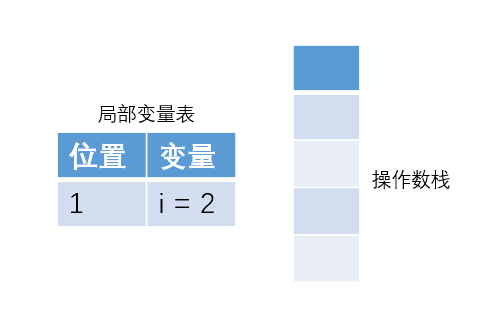
*//和上面i++差不多，不过IINC 1 1 和ILOAD 1这两句的顺序调换了。*  
ICONST\_1  
ISTORE 1  
IINC 1 1 *//直接把局部变量表中位置为“1”的变量加1*  
ILOAD 1  *//把位置“1”的变量压到栈顶，此时栈顶的元素是 2*  
INVOKEVIRTUAL java/io/PrintStream.println (I)V *//所以打印的是2*

再画下图演示一下：

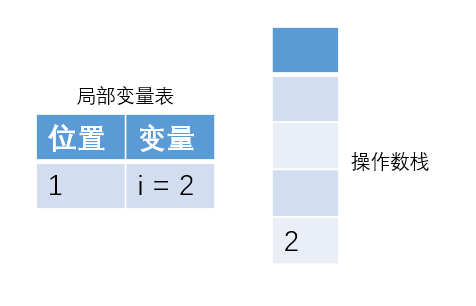
1、执行了ICONST\_1 和ISTORE 1这两句过后的局部变量和栈的情况如下



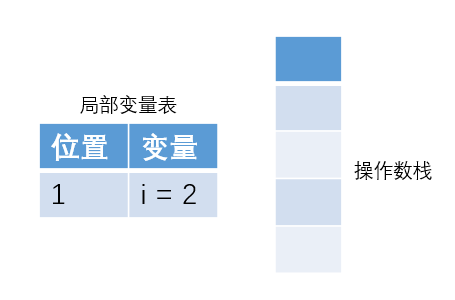
2、执行 IINC 1 1。注意，执行这条指令，操作数栈不会发生变化。



3、执行 ILOAD 1，把位置“1”的变量值压入栈顶



4、执行 INVOKEVIRTUAL java/io/PrintStream.println (I)V  ，把栈顶的元素打印出来，此时栈顶的元素是 2



所以，对于 i++ 和 ++i的区别彻底懂了吧。

接下来我们来分析这个程序

int i = 1;  
System.out.println(i+++i++);  
System.out.println(i);

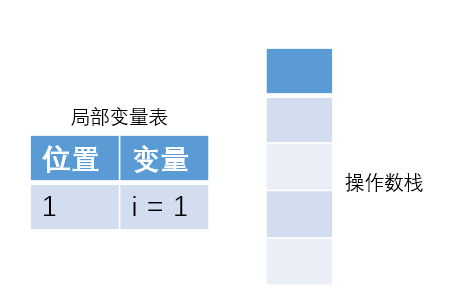
这里先说一下，按照运算符号的优先顺序，i+++i++等价于 (i++) + (i++)。

对应的部分汇编指令如下：

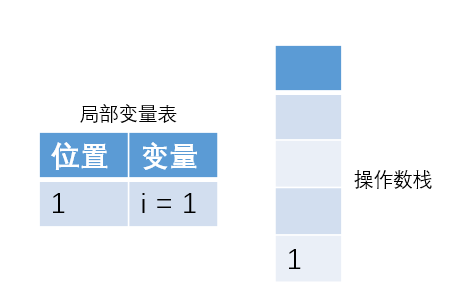
//第一行  
ICONST\_1  
ISTORE 1  
//第二行  
ILOAD 1  
IINC 1 1  
ILOAD 1  
IINC 1 1  
IADD  *//把栈顶的两个元素弹出相加之后在把结果放回栈顶*  
INVOKEVIRTUAL java/io/PrintStream.println (I)V  
//第三行  
ILOAD 1  
INVOKEVIRTUAL java/io/PrintStream.println (I)V

如果上面的那两个 i++ 和 ++i你看懂了，那么上面那个汇编应该也差不多能看懂。我用图来逐条分析一下吧。

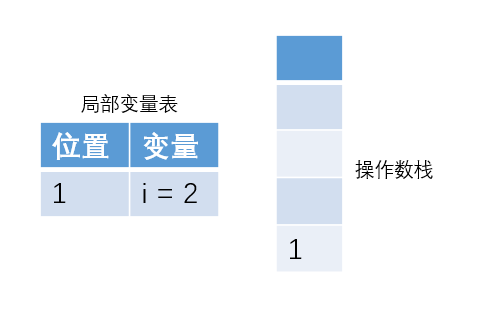
1、执行了 ICONST\_1 和ISTORE 1之后的状态如下



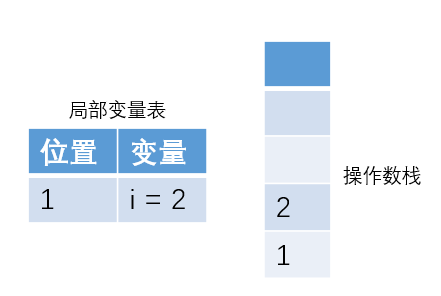
2、执行 ILOAD 1



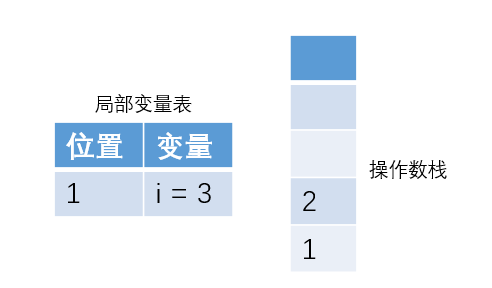
3、执行 IINC 1 1



4、执行  ILOAD 1

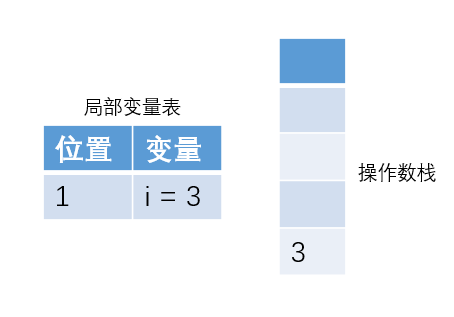


5、执行 IINC  1 1。

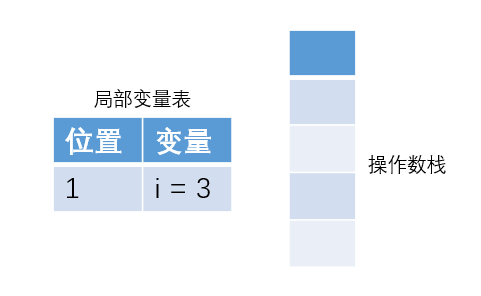


此时实际上 i 的值已经是 3 了，只是栈顶放的都是 i 的旧值。

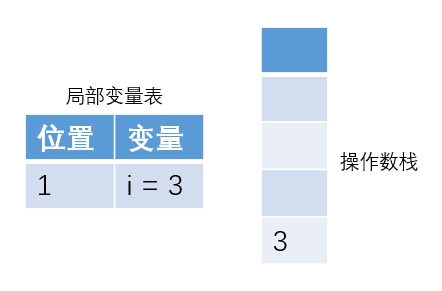
6、执行 IADD ，把栈顶两个元素出栈相加后再把结果入栈



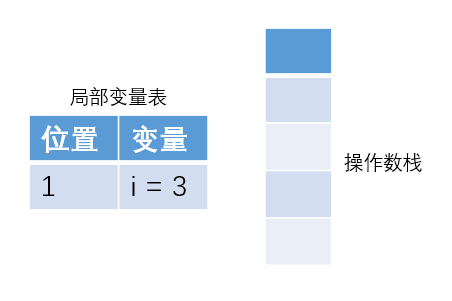
7、执行INVOKEVIRTUAL java/io/PrintStream.println (I)V，此时栈顶元素为3，所以打印的是3



8、执行  ILOAD 1，把局部变量表加载到栈顶



9、执行INVOKEVIRTUAL java/io/PrintStream.println (I)V，此时栈顶元素为3，所以打印的是3



完毕

现在知道了把，对于 i+++++i 的题也知道怎么做以及怎么回事了吧。

这篇文章重点让你理解 i++ 与 ++ i的实现机制，对于上面的汇编指令以及进栈入栈的过程为了更好着说明要解决的问题，所以隐藏了很多细节，而且也删除了部分代码。如有错误的地方，还请见谅。

如果你想了解更多的汇编指令，我这里看到一篇总结的还挺全的：https://blog.csdn.net/hudashi/article/details/7062675

有收获？点击底部卡片加个鸡腿犒劳一下？

https://mmbiz.qpic.cn/mmbiz_gif/b96CibCt70iaa8r7PJoyAtlfHAKe8RosE3wYVKBac55p1HPBJHZS42ywnG4yYtD3jo9A9e5kawBZs4IE6R1C4wibw/640?wxfrom=5&wx_lazy=1

- End -

推荐阅读：

[【算法实战】生成窗口最大值数组](http://mp.weixin.qq.com/s?__biz=MzUxNzg0MDc1Mg==&mid=2247484791&idx=1&sn=555ecf7315a549de05555e5009678c8b&chksm=f9934f68cee4c67eeecd09a63f7212971fcb3e14eaa132c46b033085f562e5051e5a7300ef14&scene=21#wechat_redirect)

[谈谈NAT：什么？全球IP和私有IP是什么鬼？](http://mp.weixin.qq.com/s?__biz=MzUxNzg0MDc1Mg==&mid=2247484741&idx=1&sn=7c18cbb8ab38a57848640bd8397e451d&chksm=f9934f5acee4c64cfe9a8ef5da90cac3925a09290eeecbf194a126b0acab6e2af76deb5da42f&scene=21#wechat_redirect)





微信扫一扫  
关注该公众号