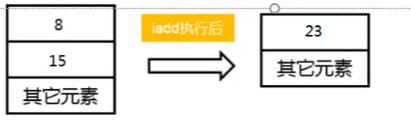
**操作数栈（Operand Stack）：**

1、每一个独立的栈帧中除了包含局部变量表以外，还包含一个后进先出（Last-In-First-Out）的操作数栈，也可以称之为表达式栈（Expression Stack）。

2、操作数栈，在方法执行过程中，根据字节码指令，往栈中写入数据或提取数据，即入栈出栈。

3、某些字节码指令将值压入操作数栈，其余的字节码指令将操作数取出栈，使用他们后再把结果压入栈。

4、比如：执行复制、交换、求和等操作。

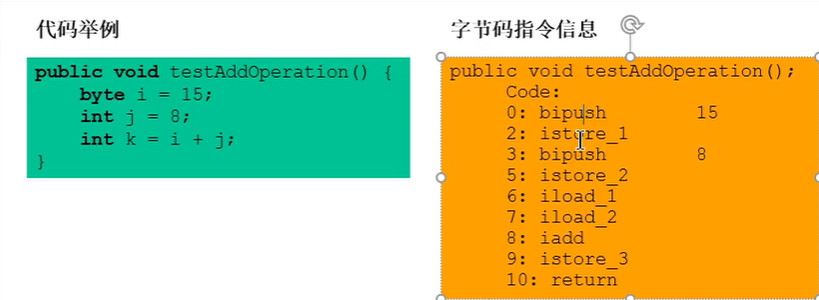


5、如果被调用的方法带有返回值的话，其返回值将会被压入当前栈帧的操作数栈中，并更新PC寄存器中下一条需要执行的字节码指令。

６、操作数栈中元素的数据类型必须与字节码指令的序列严格匹配，这由编译器在编译期间进行验证，同时在类加载过程中的类检验阶段的数据流分析阶段要再次验证。

７、另外，我们说Java虚拟机的**解释引擎是基于栈的执行引擎**，其中的**栈指的就是操作数栈**。（[JVM字节码之整型入栈指令(iconst、bipush、sipush、ldc)\_iechenyb\_新浪博客 (sina.com.cn)](http://blog.sina.com.cn/s/blog_99201d890102x7mc.html)）

（[JVM字节码之整型iconst、istore、iload指令\_kq1983的专栏-CSDN博客](https://blog.csdn.net/kq1983/article/details/113406715)）



８、操作数栈，主要用于保存计算过程的中间结果，同时作为计算过程中变量临时的存储空间。

９、操作数栈就是JVM执行引擎的一个工作区，当一个方法开始执行的时候，一个新的栈帧也会随之被创建出来，这个方法的操作数栈是空的。

１０、每一个操作数栈都会拥有一个明确的栈深度用于存储数值，其所需的最大深度在编译期就定义好了，保存在方法的Ｃｏｄｅ书信中，为ｍａｘ＿ｓｔａｃｋ的值。

１１、栈中的任何一个元素都是可以任意的Java数据类型。（３２位的类型占用一个栈单位深度、６４位的类型占用两个栈单位深度）

１２、**操作数栈并非采用方法索引的方式来进行数据访问的，而是只能通过标准的入栈和出栈操作来完成一次数据访问。**