精通JVM，不是从调优开始！！！

前言

很多Java开发人员，还不清楚JVM是什么，就直接涉足JVM调优。

很多Java开发人员，说起JVM，只知道运行时数据区域（或称之为JVM内存布局或JVM内存结构）。

还有很多Java开发人员，在被问起：“你知道Java内存模型吗？”，答：“知道，Java内存模型分为方法区、堆、...”。

...

虽说“概念”这个词发明出来是为了方便交流和说明问题的，不用太过于较真。但是，答非所问就是你的不对了。

如果你属于其中一者，我劝你看完这篇文章。

本文就一个目的：认识JVM。

从 Java 的跨平台特性说起

> 认识JVM应该从哪里入手?

可以问自己几个问题，为什么有JVM？没有JVM会怎样?

Java编程语言设计的初衷就是要解决一个问题：**程序员编写一次程序，可以在任何提供Java运行时环境的机器上运行**。

也就是Java语言的跨平台特性“一次编写，到处运行”。

这一点可以在Oracle官方提供的文档里证实：

[docs.oracle.com/javase/spec...](http://cxyroad.com/”https://docs.oracle.com/javase/specs/jvms/se8/html/jvms-1.html”)

可能有些Java开发人员对跨平台特性不是很理解：“不应该就是编写一次到处运行吗？”。

当然不是，如果了解C、C++语言的，就知道，想要在不同平台上运行，需要分别编译，甚至重新编写。

当然，这是所有编程语言都存在的问题。原因下面有说明。

> 那么，Java是如何解决的?

Java 跨平台的实现：JVM

Java之所以能实现跨平台，离不开JVM的支持。

下面对此展开说明。

JVM 的定义

JVM，全称：Java Virtual Machine （Java 虚拟机）。

下面是官方对JVM的定义：

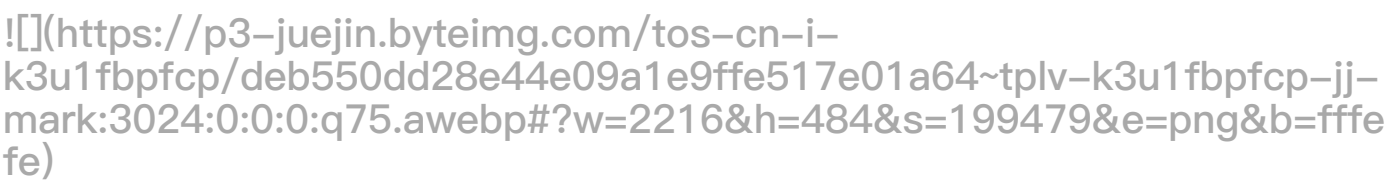
> Java虚拟机是一种抽象的计算机。就像真正的计算机一样，它有一个指令集

，并在运行时处理各种内存区域。

这个说明已经非常简单明了了。如果不理解的继续看下文。

跨平台问题说明

任何一个高级编程语言编写的程序，在最终执行前，都会被翻译成计算机可以理解的语言：机器码，也就是诸如0101的二进制数。



这些机器码就是指令的实际表示，计算机的工作原理就是通过执行这些指令来完成具体的任务。
（每一条指令中明确规定了计算机从哪个地址取数，进行什么操作，然后送到什么地址去等步骤）

而同一个操作，在不同的平台，指令可能会有所不同。

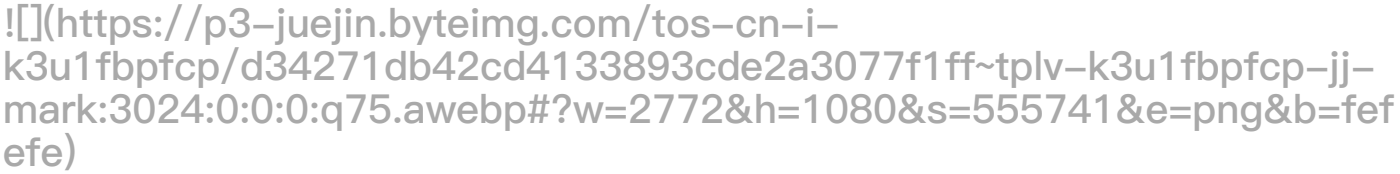
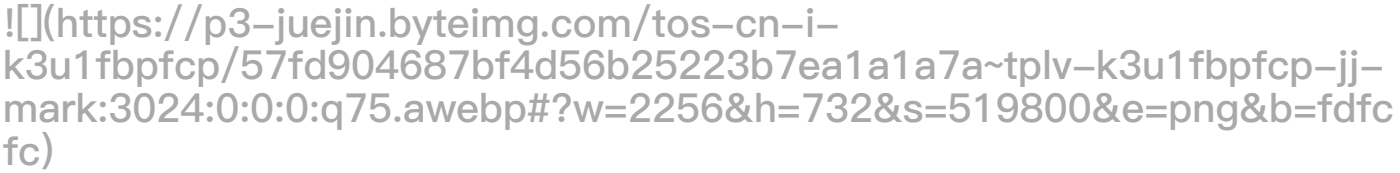
例如，“将数据从内存加载到寄存器”的操作，在x86架构和ARM架构中，指令就不一样：

* 在x86架构中，可能会使用MOV指令将数据加载到寄存器EAX中：

```
MOV EAX, [0x12345678]
```

* 在ARM架构中，可能会使用LDR指令将数据加载到寄存器R0中：

```
LDR R0, [0x12345678]
```

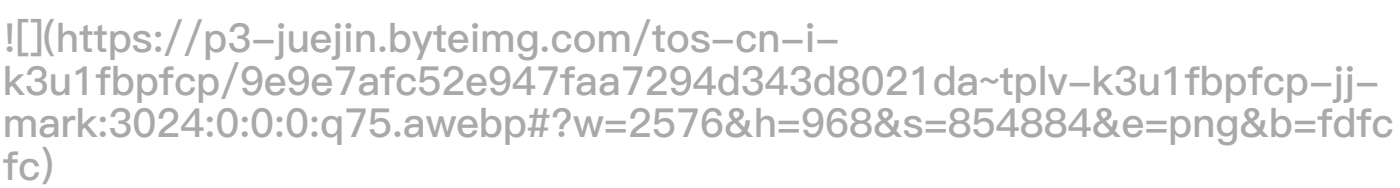
```

所以，在一个平台上编写的程序要在其他平台上运行，就需要重新编译，甚至重写。否则就会出现下图现象。




这也正是跨平台问题存在的根本原因。

从编码到运行
---

而JVM作为一个抽象的计算机，对外提供了一套自己的指令集，作用何在?

作用是：**在不同平台运行符合自己指令的程序时，会将该指令转换成当前机器的本地指令。**

这样就避免了多次编译的操作，从而实现“一次编写，到处运行”。

或者这样理解：JVM替代了多次编译的工作。

> 那么，符合自己指令的程序是什么呢?

没错，就是Java字节码（class文件）。

我们平时在编写完.java源码后，会经过JDK提供的编译工具（javac）编译为.class，而.class文件里面的内容就是JVM指令。


```

JVM 已成为规范
==========

可能会有人有疑问，“为什么javac不直接编译成机器码？”

三个方面回答这个问题：

1. 如果javac直接编译成机器码，就又回到跨平台的问题了。
2. JVM除了解决跨平台问题外，还负责了内存管理和安全性的问题。
3. 发展至今，JVM已经成为了一种规范，只要符合JVM规范，支持任何语言运行在JVM上。



JVM 的后续了解
==========

通过上面的描述，如果理解了JVM是什么、JVM的作用或者JVM存在的意义，那就算是入门了。

如果激发了你对“JVM指令是什么”以及“它们如何被转换成机器指令”的好奇心，或者产生“类加载和JVM的关系”、“垃圾回收器和JVM的关系”等疑问。

那么，请我，后面的文章会带你一起了解。或者，自己可以去官方进行了解，官方文档链接给你放在这。

Java 各版本文档：
[docs.oracle.com/en/java/jav...](http://cxyroad.com/”https://docs.oracle.com/en/java/javase/”)

Java语言&JVM规范文档：
[docs.oracle.com/javase/spec...](http://cxyroad.com/”https://docs.oracle.com/javase/specs/index.html”)

总结

最后总结一下本文内容：

1. 所有高级编程语言编写的程序最终要翻译为机器码（指令的实际表示）才能被计算机理解。
2. 由于计算机的硬件架构、操作系统不同，同一个操作指令可能也会不同。所以，程序要想在不同平台上运行，需要重新翻译或编写。
3. Java最初是为了解决程序跨平台运行的问题，JVM正是实现跨平台特性的关键所在。
4. Java语言编写的程序会被编译为Java字节码，也就是JVM自身的指令。运行时，JVM会将指令转换成本地机器指令，从而实现跨平台。
5. JVM发展至今，已然是一个规范，已支持上百种编程语言。

当了解了JVM的作用、存在的意义，再去了解JVM的工作原理就不会那么困惑了。

原文链接: https://juejin.cn/post/7373577112220532745