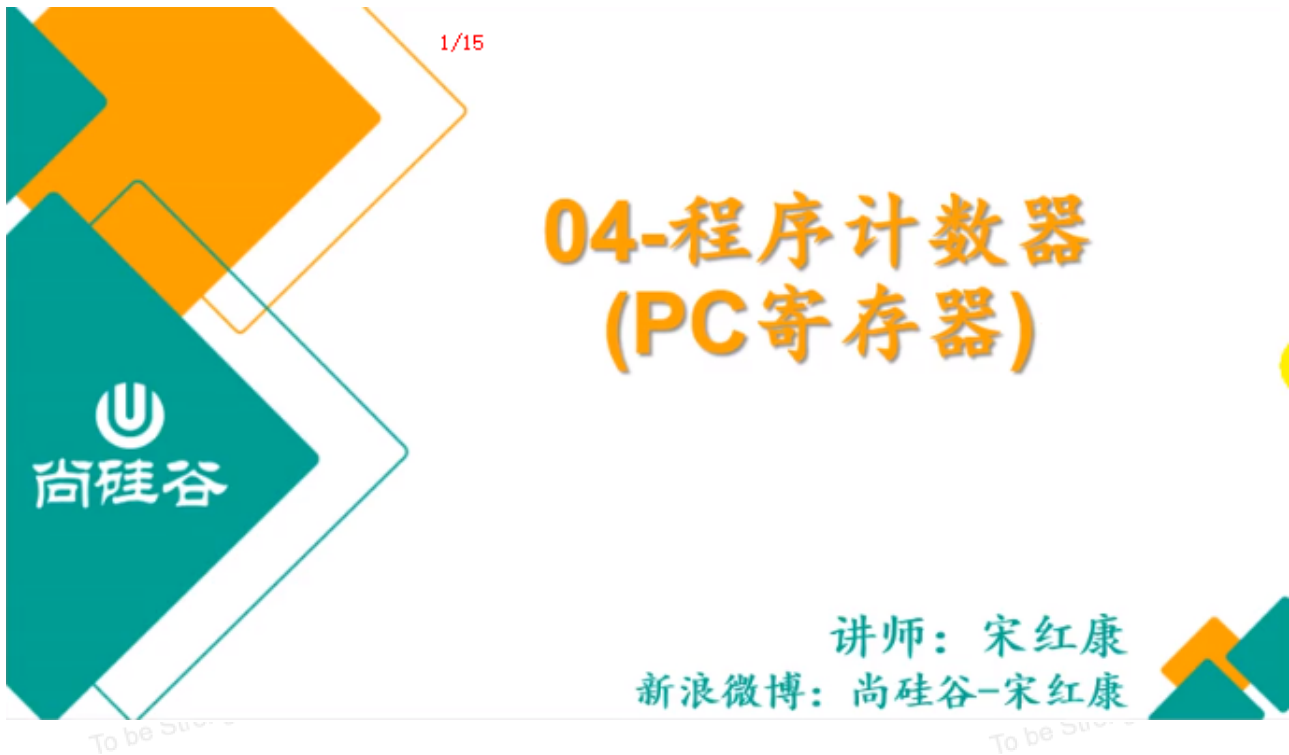


04_程序计数器.pptx



1.PC Register介绍



3/15

1- PIC Register介绍

让天下没有难学的技术



4/15

The Java® Virtual Machine Specification

Java SE 8 Edition

Tim Lindholm

Frank Yellin

Gilad Bracha

Alex Buckley

2015-02-13

<https://docs.oracle.com/javase/specs/jvms/se8/html/>

让天下没有难学的技术

To be Stronger

To be Stronger

PC Register介绍

5/15

The diagram shows the JVM Runtime Data Areas. On the left are the Method Area (red) and Heap Area (blue). In the center is the Stack Area, which contains three threads (Thread 1, Thread 2, Thread 1). Each thread has its own stack frame, which includes local variables (LVA), operand stack (OS), and frame data (FD). To the right of the Stack Area are the PC Registers, which store the program counter for each thread (Thread 1, Thread 2, ..., Thread n). Further right is the Native Method Stack (purple). The entire area is labeled 'Runtime Data Areas' at the bottom.

JVM中的程序计数寄存器（Program Counter Register）中，Register 的命名源于CPU的寄存器，寄存器存储指令相关的现场信息。CPU只有把数据装载到寄存器才能够运行。

这里，并非是广义上所指的物理寄存器，或许将其翻译为PC计数器（或指令计数器）会更加贴切（也称为程序钩子），并且也不容易引起一些不必要的误会。**JVM中的PC寄存器是对物理PC寄存器的一种抽象模拟。**

让天下没有难学的技术

作用

6/15

The diagram illustrates the role of the PC Register. On the left, a box labeled '当前线程' (Current Thread) contains an '执行引擎' (Execution Engine) and a 'PC寄存器' (PC Register). The '执行引擎' is connected to the 'PC寄存器'. The 'PC寄存器' is connected to the 'Java栈' (Java Stack). The 'Java栈' is shown as a vertical stack of frames. The top frame is the '当前栈帧' (Current Frame), which contains a '局部变量表' (Local Variable Table), '操作数栈' (Operand Stack), '动态链接' (Dynamic Link), and '方法返回值' (Method Return Value). Below the current frame are other frames labeled '栈帧n', '栈帧2', and '栈帧1'. The 'Java栈' is connected to the '方法区' (Method Area) and the 'Java堆区' (Java Heap Area).

作用：

PC寄存器用来存储指向下一条指令的地址，也即将要执行的指令代码。由执行引擎读取下一条指令。

让天下没有难学的技术



PC Register介绍



7/15

- 它是一块很小的内存空间，几乎可以忽略不计。也是运行速度最快的存储区域。
- 在JVM规范中，每个线程都有它自己的程序计数器，是线程私有的，生命周期与线程的生命周期保持一致。
- 任何时间一个线程都只有一个方法在执行，也就是所谓的**当前方法**。程序计数器会存储当前线程正在执行的Java方法的JVM指令地址；或者，如果是在执行native方法，则是未指定值（undefined）。

让天下没有难学的技术



PC Register介绍



8-15

- 它是程序控制流的指示器，分支、循环、跳转、异常处理、线程恢复等基础功能都需要依赖这个计数器来完成。
- 字节码解释器工作时就是通过改变这个计数器的值来选取下一条需要执行的字节码指令。
- 它是唯一一个在Java 虚拟机规范中没有规定任何OutOfMemoryError情况的区域。

让天下没有难学的技术

To be ~~~

To be ~~~

2.举例说明

To be Stronger

To be Stronger







9-15

2- 举例说明

让天下没有难学的技术



举例说明



10-15

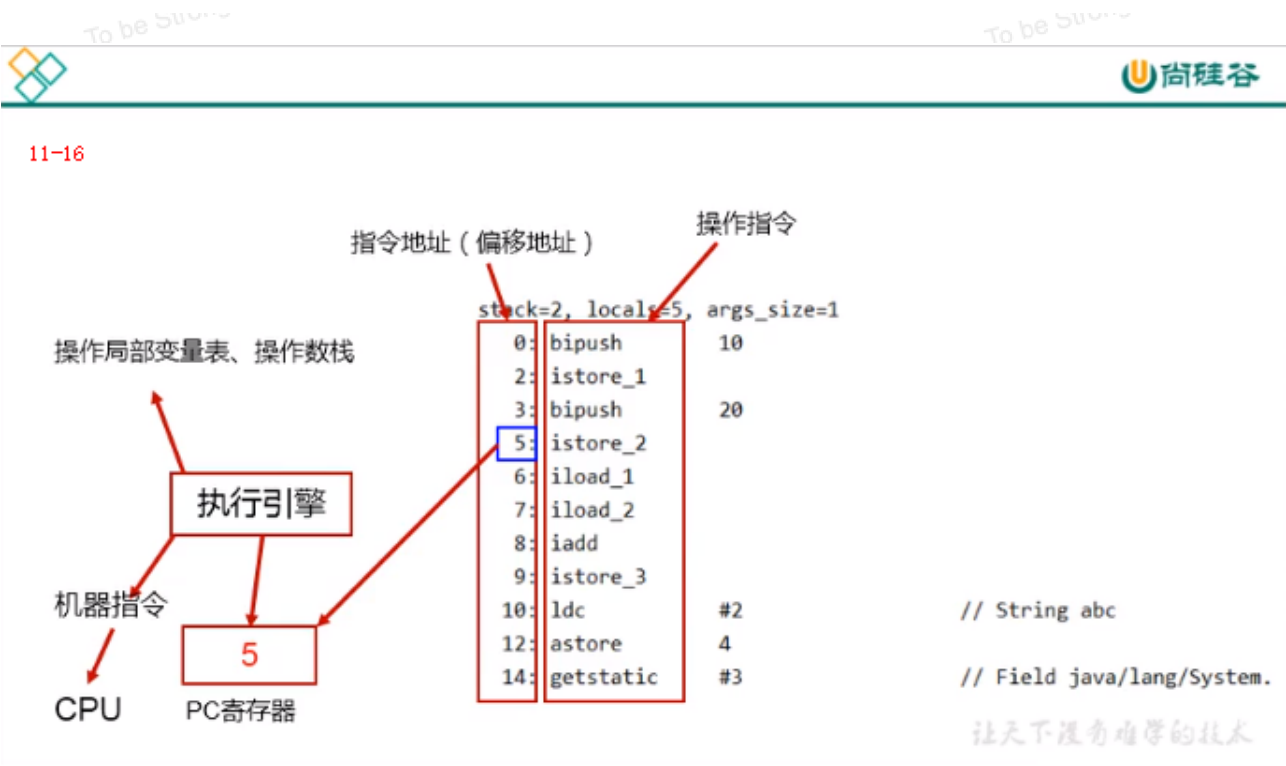
```
public int minus(){  
    int c = 3;  
    int d = 4;  
    return c - d;  
}
```

↓

字节码文件

→

```
public int minus();  
descriptor: ()I  
flags: ACC_PUBLIC  
Code:  
    stack=2, locals=3, args_size=1  
    0: iconst_3  
    1: istore_1  
    2: iconst_4  
    3: istore_2  
    4: iload_1  
    5: iload_2  
    6: isub  
    7: ireturn  
LineNumberTable:  
    line 15: 0  
    line 16: 2  
    line 17: 4  
LocalVariableTable:  
    Start Length Slot Name Signature  
        0      8     0  this  Lcom/atguigu/java/PCRegisterTest;  
        2      6     1    c    I
```





两个常见问题



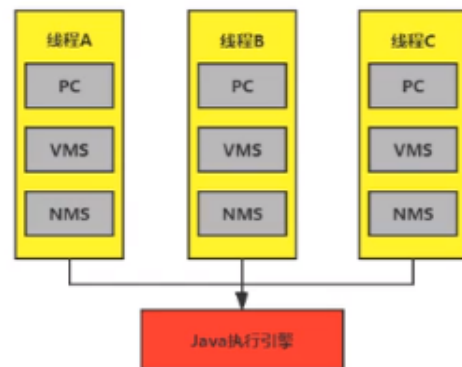
13-16

使用PC寄存器存储字节码指令地址有什么用呢？

为什么使用PC寄存器记录当前线程的执行地址呢？

因为CPU需要不停的切换各个线程，这时候切换回来以后，就得知道接着从哪开始继续执行。

JVM的字节码解释器就需要通过改变PC寄存器的值来明确下一条应该执行什么样的字节码指令。



让天下没有难学的技术

To be Stronger



两个常见问题



14-16

PC寄存器为什么会被设定为线程私有？

我们都知道所谓的多线程在一个特定的时间段内只会执行其中某一个线程的方法，CPU会不停地做任务切换，这样必然导致经常中断或恢复，如何保证分毫无差呢？**为了能够准确地记录各个线程正在执行的当前字节码指令地址，最好的办法自然是为每一个线程都分配一个PC寄存器**，这样一来各个线程之间便可以进行独立计算，从而不会出现相互干扰的情况。

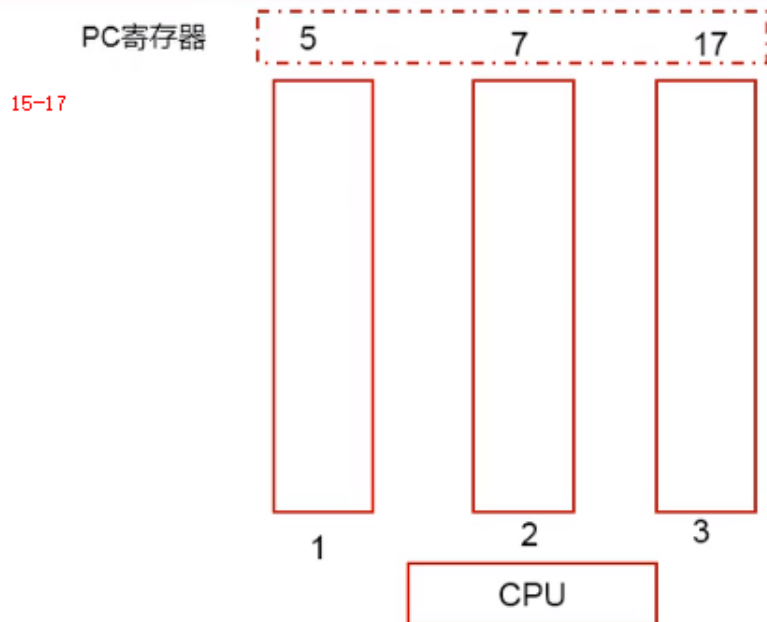
由于CPU时间片轮限制，众多线程在并发执行过程中，任何一个确定的时刻，一个处理器或者多核处理器中的一个内核，只会执行某个线程中的一条指令。

这样必然导致经常中断或恢复，如何保证分毫无差呢？每个线程在创建后，都会产生自己的程序计数器和栈帧，程序计数器在各个线程之间互不影响。

让天下没有难学的技术

To be Stronger

To be Stronger



让天下没有难学的技术



CPU 时间片

16-17

CPU 时间片即 CPU 分配给各个程序的时间，每个线程被分配一个时间段，称作它的时间片。

在宏观上：我们可以同时打开多个应用程序，每个程序并行不悖，同时运行。

但在微观上：由于只有一个 CPU，一次只能处理程序要求的一部分，如何处理公平，一种方法就是引入时间片，每个程序轮流执行。

让天下没有难学的技术
To be Stronger

To be Stronger

To be Stronger

本文来自



尚硅谷学习

参照 <http://www.atguigu.com/download.shtml> 学习

关注