

本节内容

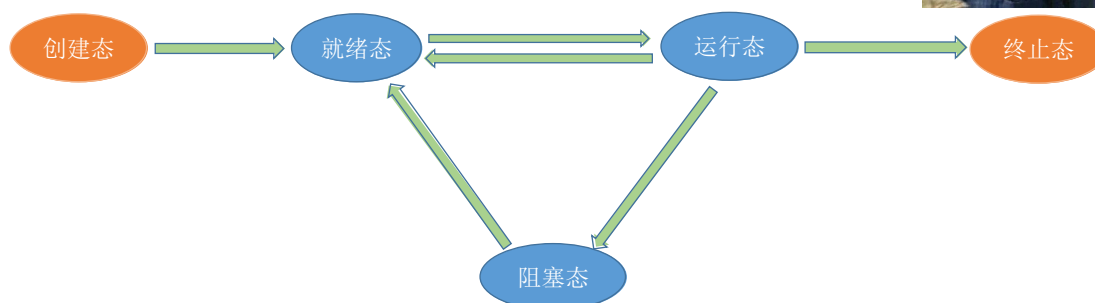
进程控制

王道考研/CSKAOYAN.COM

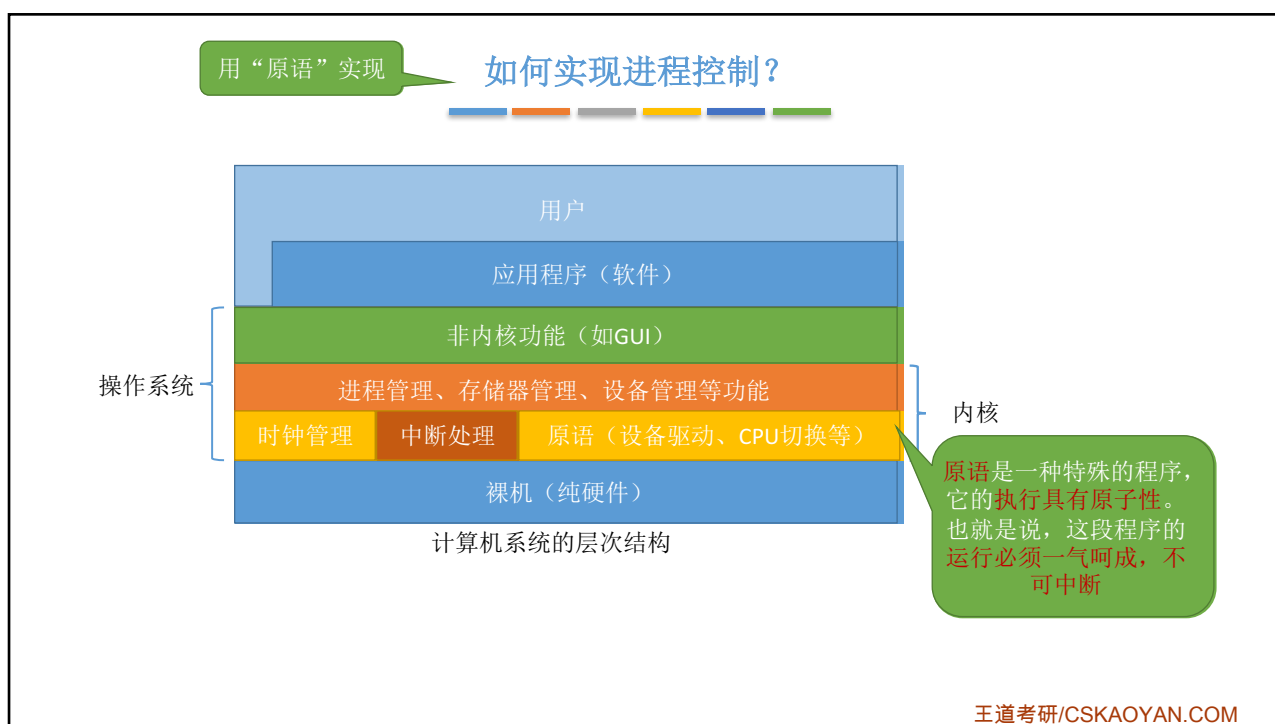
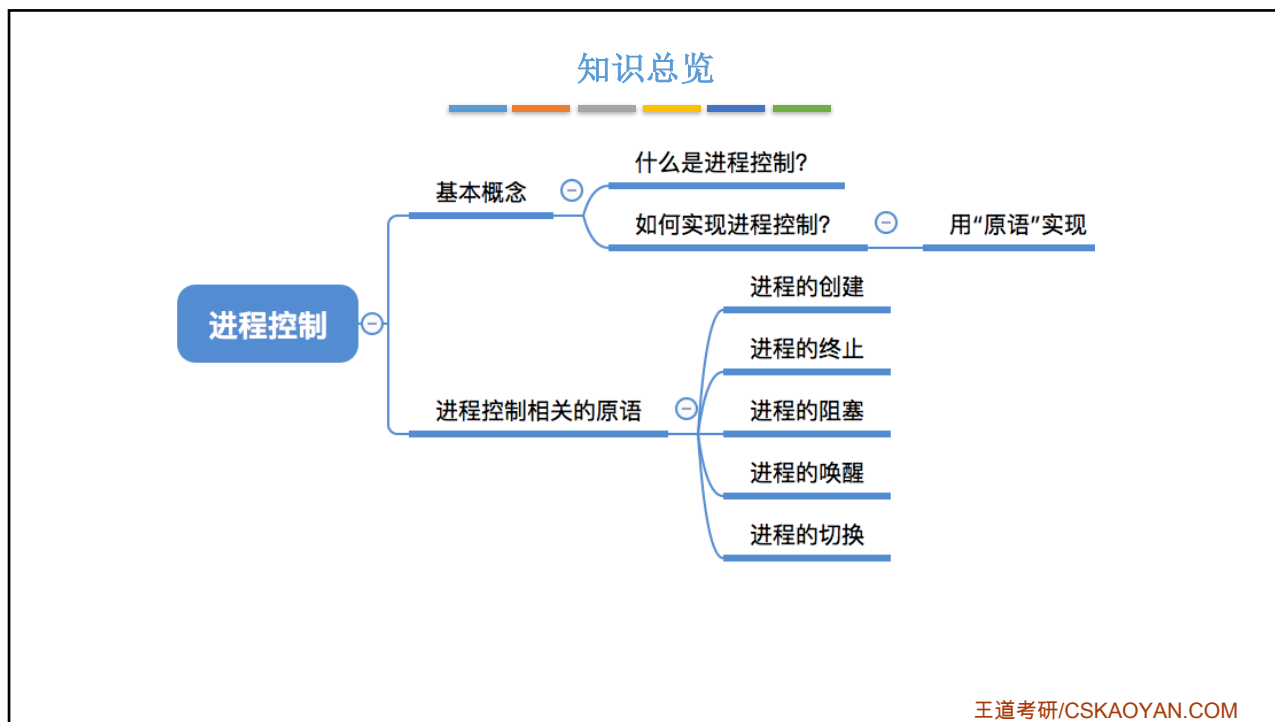
什么是进程控制?

进程控制的主要功能是对系统中的所有进程实施有效的管理,它具有创建新进程、撤销已有进程、实现进程状态转换等功能。

简化理解: 反正进程控制就是要实现进程状态转换



王道考研/CSKAOYAN.COM




如何实现进程控制?

用“原语”实现

原语的执行具有“原子性”，一气呵成

思考：为何进程控制（状态转换）的过程要“一气呵成”？


如果不能“一气呵成”，就有可能导致操作系统中的某些关键数据结构信息不统一的情况，这会影响操作系统进行别的管理工作



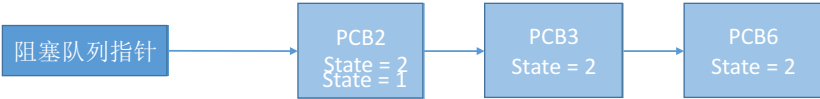
成熟的思考者

Eg: 假设PCB中的变量 **state** 表示进程当前所处状态，1表示就绪态，2表示阻塞态...

就绪队列指针



阻塞队列指针



假设此时进程2等待的事件发生，则操作系统中，负责进程控制的内核程序至少需要做这样两件事：

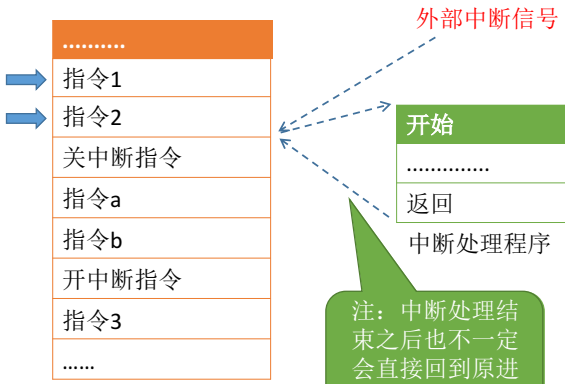
- ①将PCB2的 **state** 设为 1
- ②将PCB2从阻塞队列放到就绪队列

完成了第一步后收到中断信号，那么PCB2的state=1，但是它却被放在阻塞队列里

王道考研/CSKAOYAN.COM

如何实现原语的“原子性”？

原语的执行具有**原子性**，即执行过程只能一气呵成，期间**不允许被中断**。
 可以用“**关中断指令**”和“**开中断指令**”这两个**特权指令**实现原子性



(内核程序，运行在核心态)

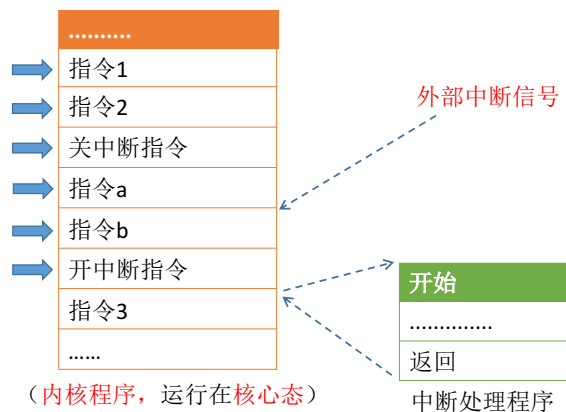
正常情况：CPU每执行完一条指令都会例行检查是否有中断信号需要处理，如果有，则暂停运行当前这段程序，转而执行相应的中断处理程序。

注：中断处理结束之后也不一定会直接回到原进程执行

王道考研/CSKAOYAN.COM

如何实现原语的“原子性”？

原语的执行具有原子性，即执行过程只能一气呵成，期间不允许被中断。可以用“关中断指令”和“开中断指令”这两个特权指令实现原子性



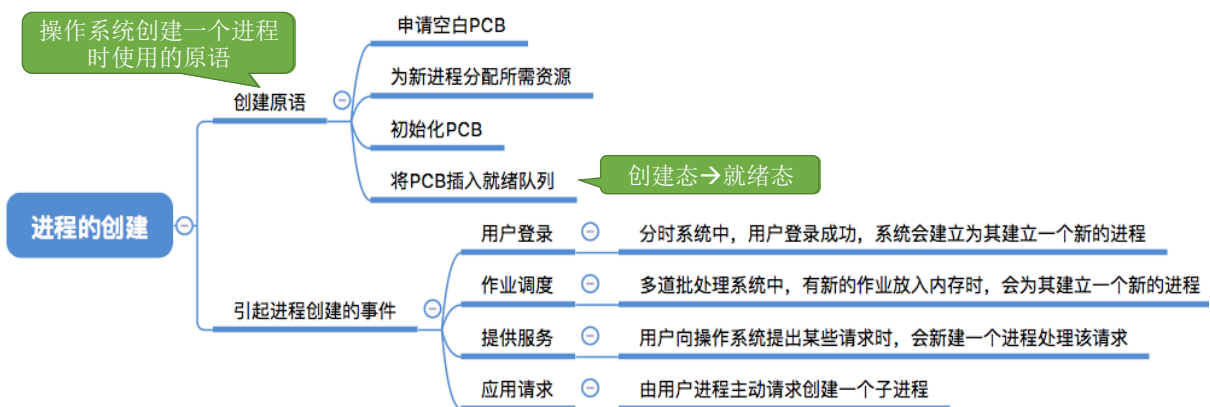
CPU执行了关中断指令之后，就不再例行检查中断信号，直到执行开中断指令之后才会恢复检查。

这样，关中断、开中断之间的这些指令序列就是不可被中断的，这就实现了“原子性”

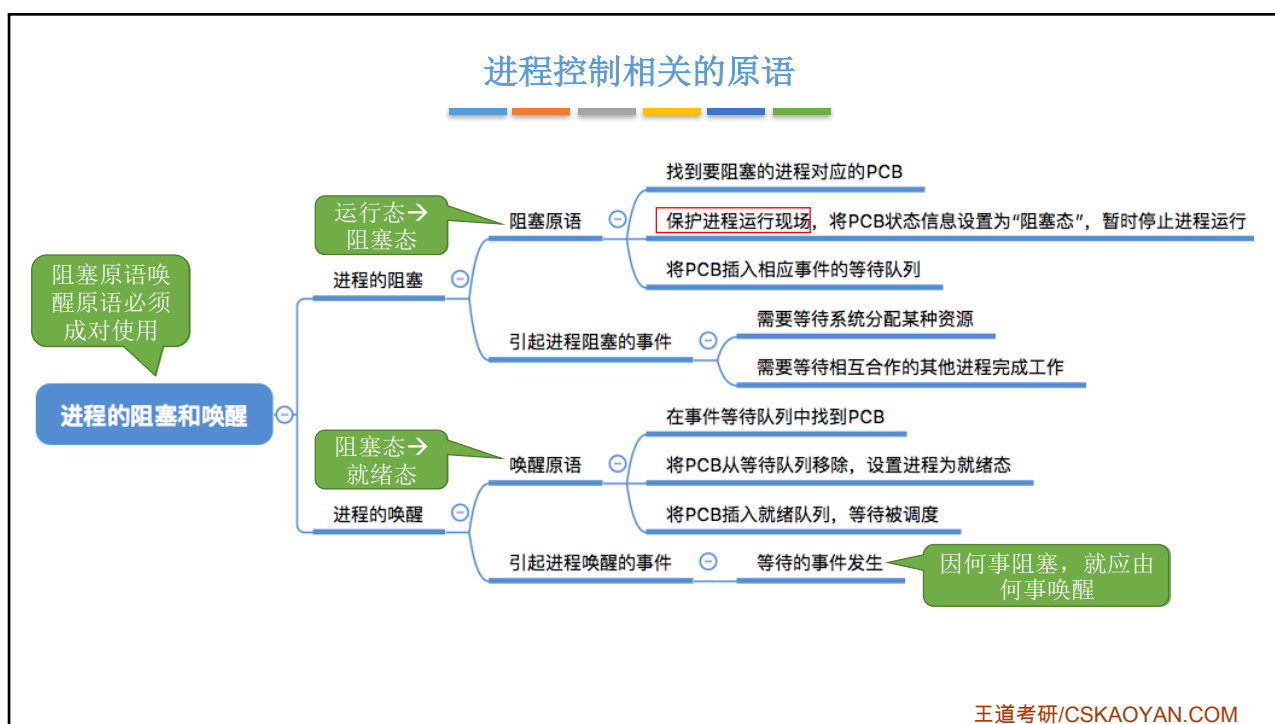
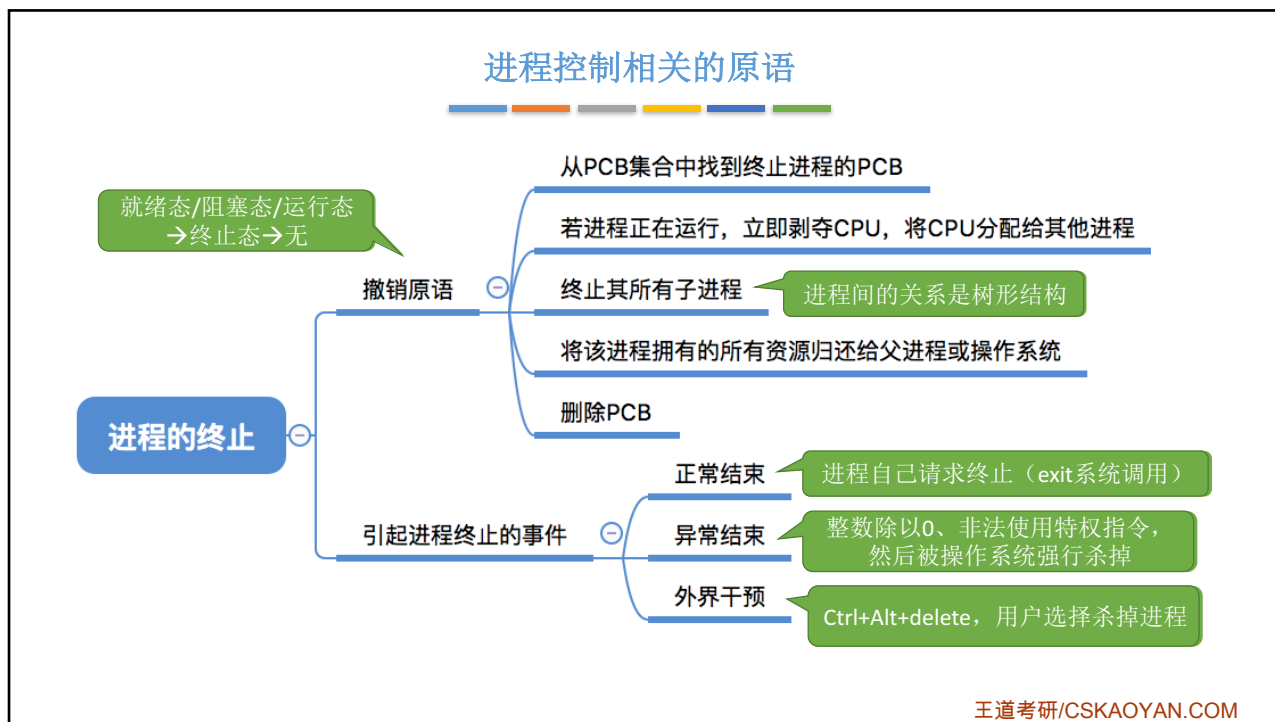
思考：如果这两个特权指令允许用户程序使用的话，会发生什么情况？

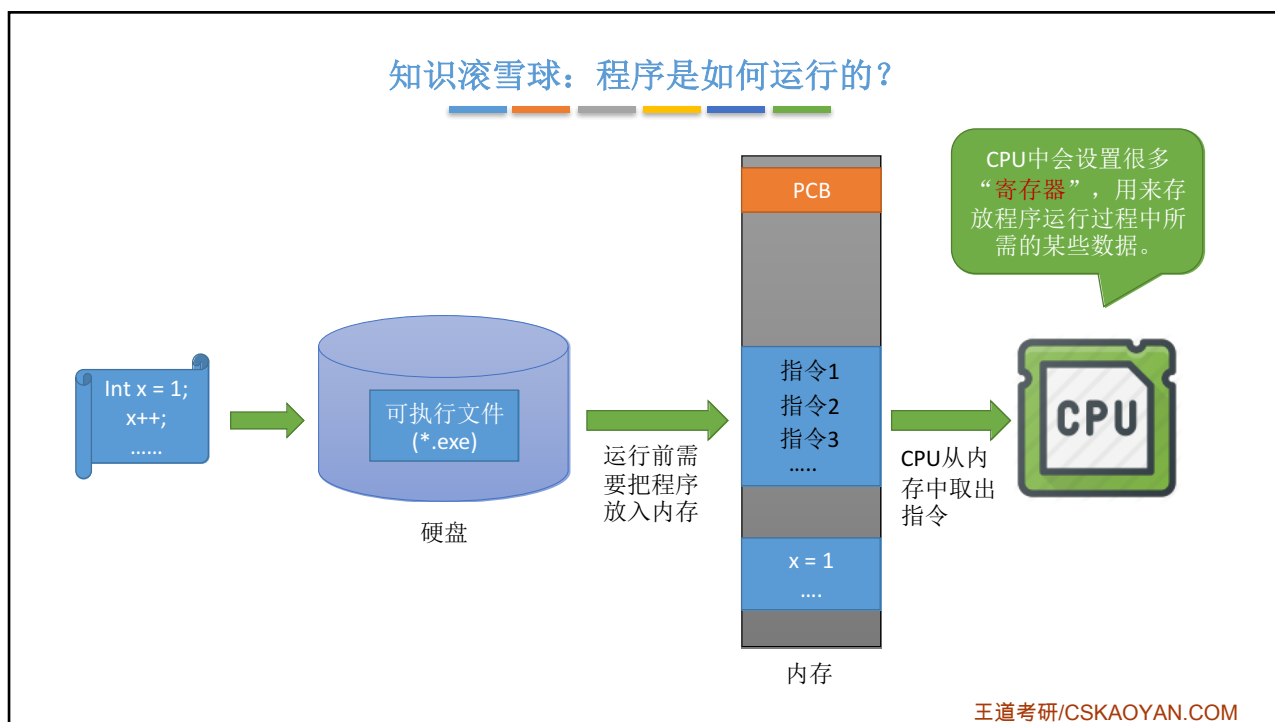
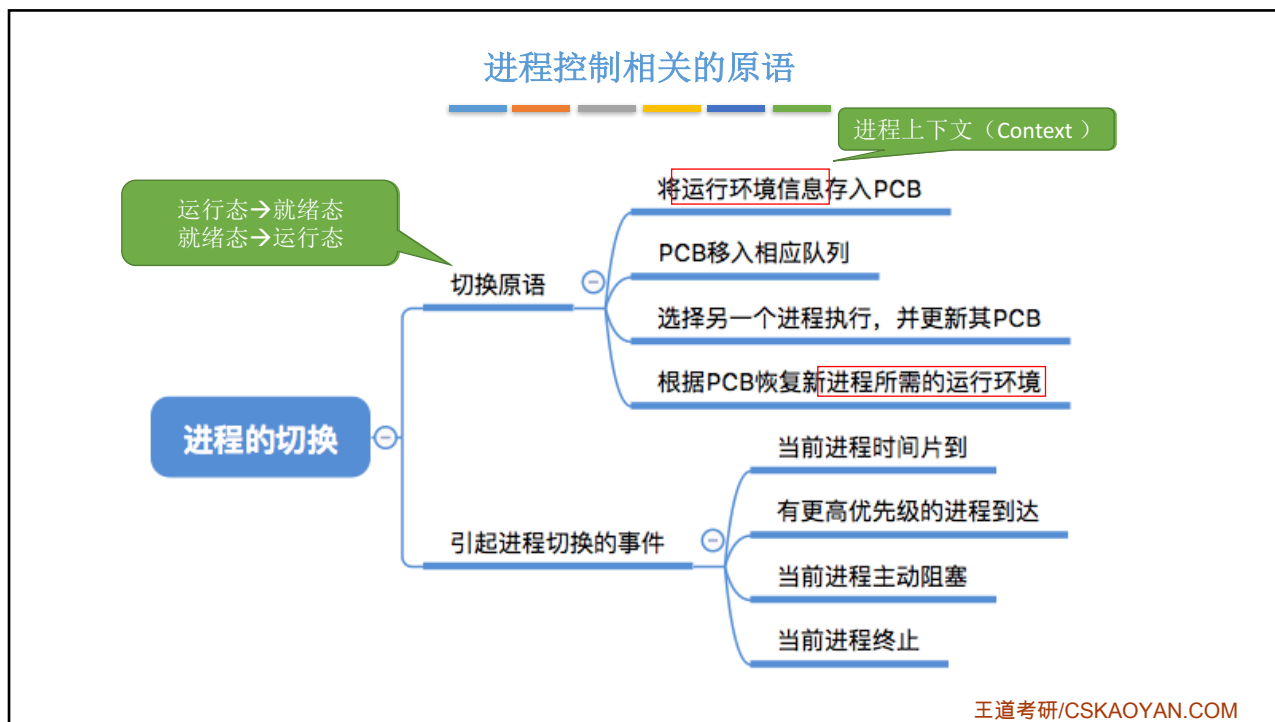
王道考研/CSKAOYAN.COM

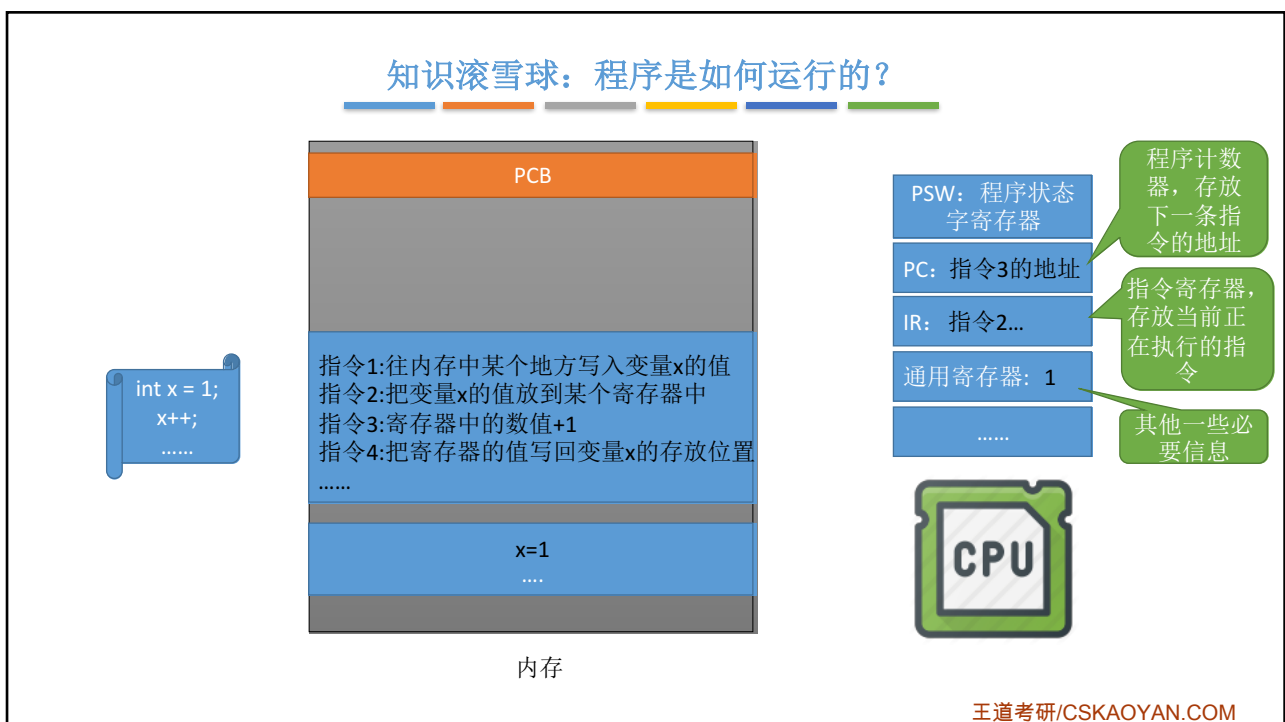
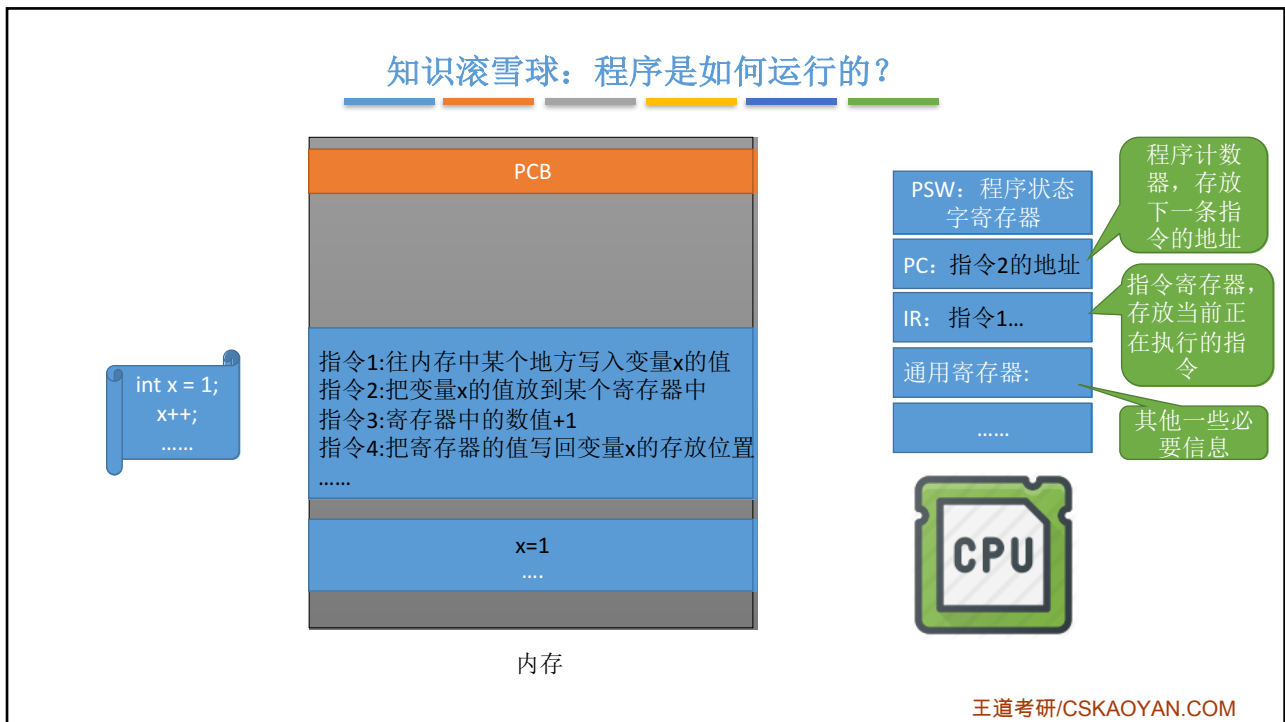
进程控制相关的原语

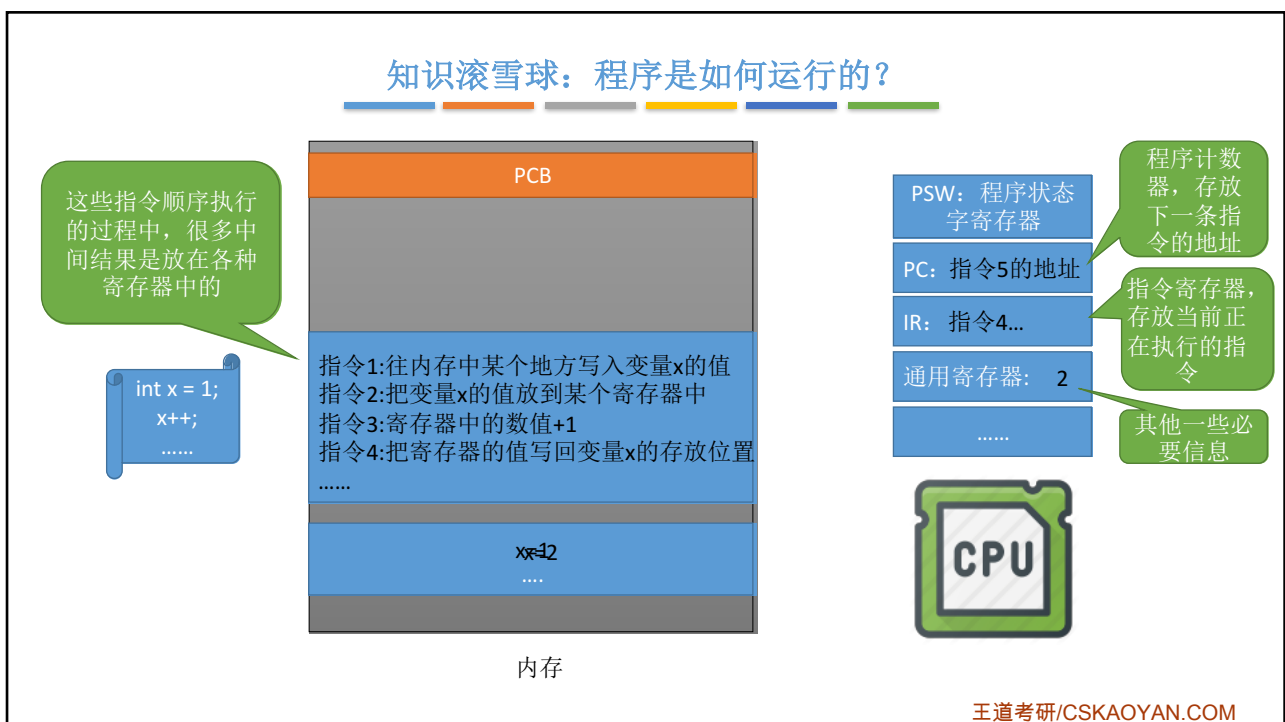
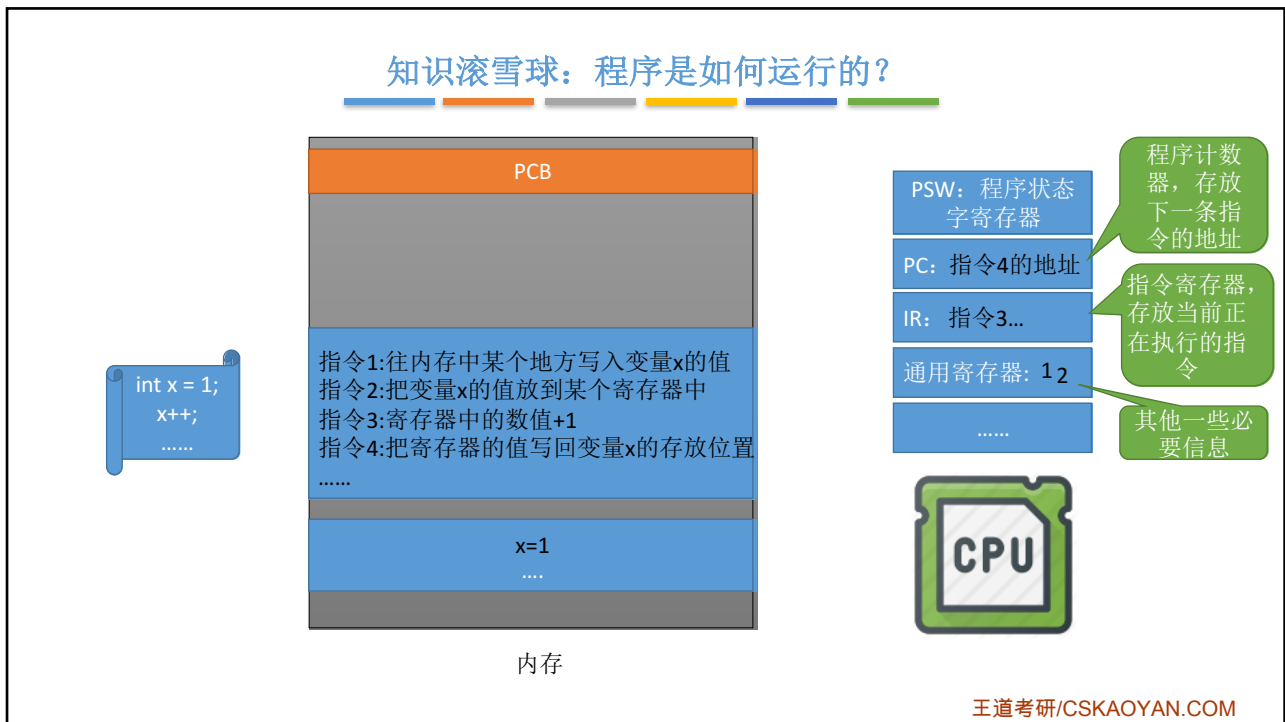


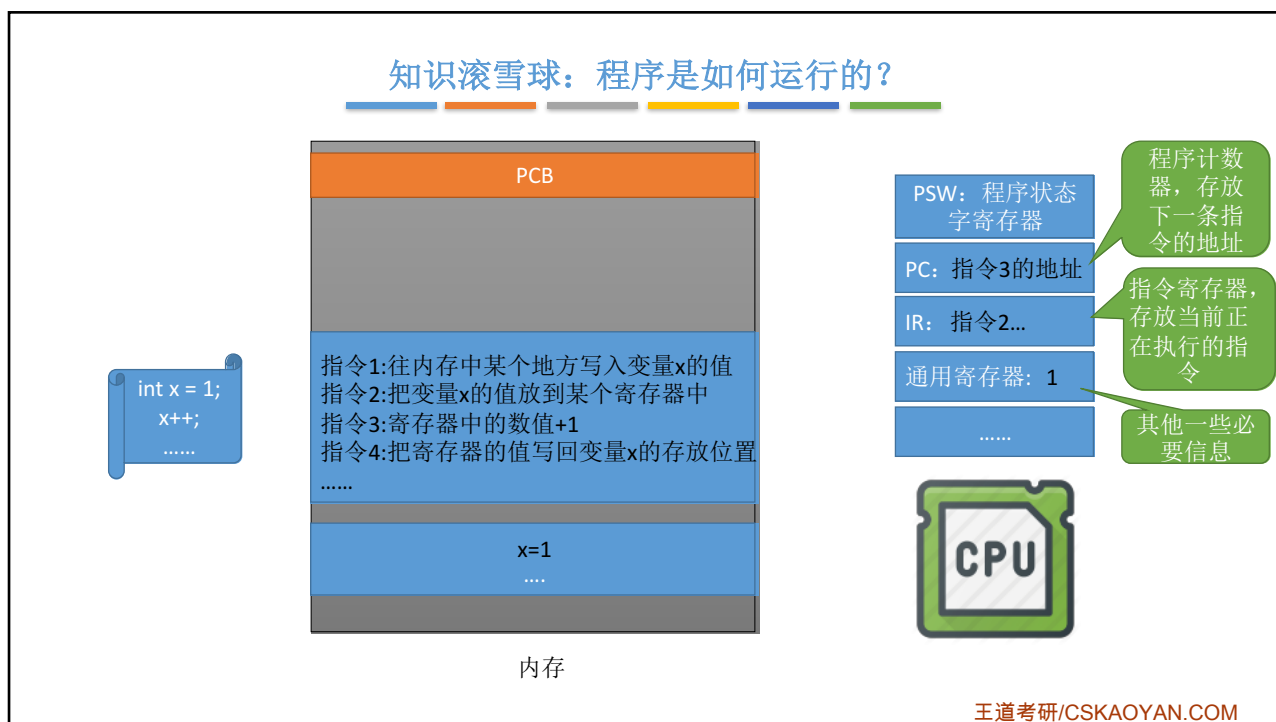
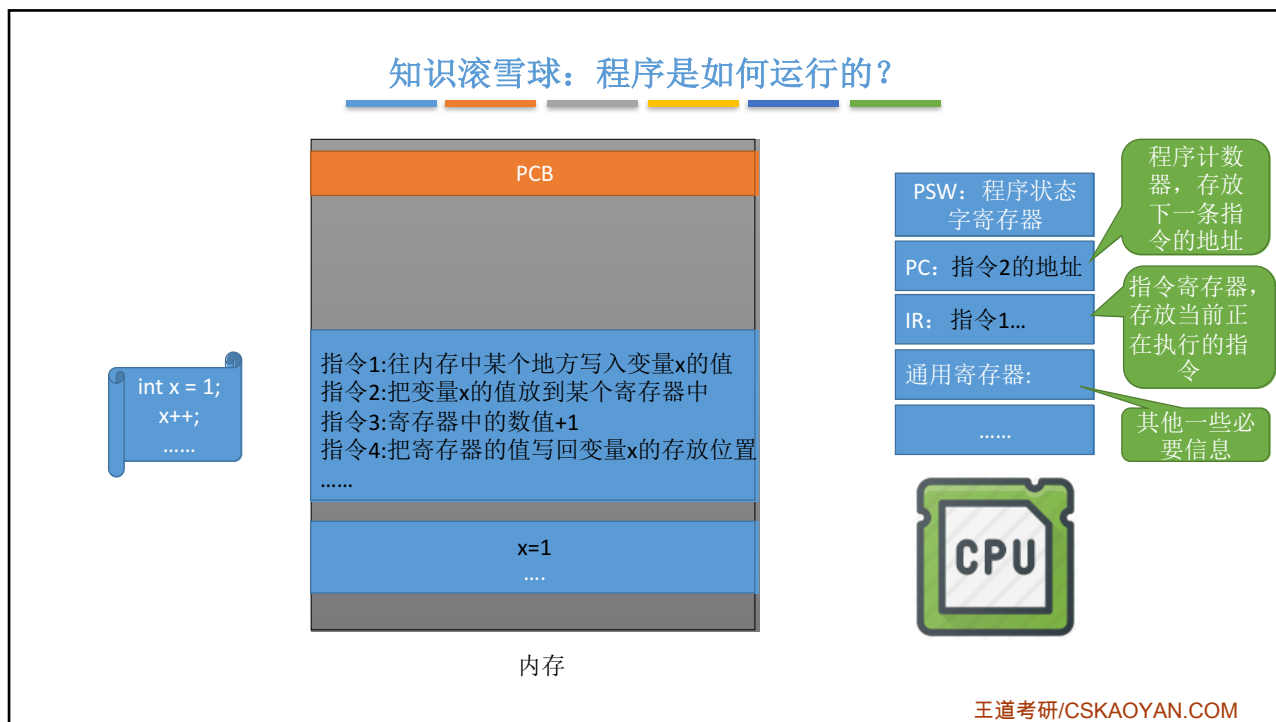
王道考研/CSKAOYAN.COM









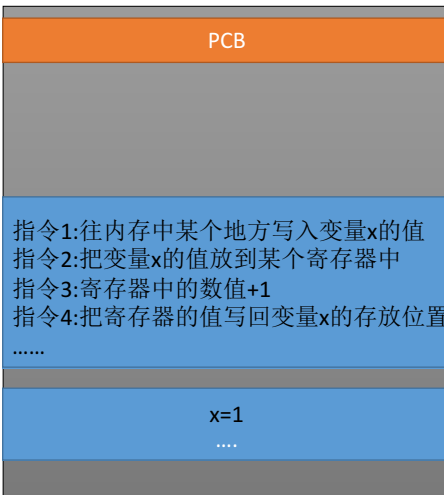


知识滚雪球：程序是如何运行的？

思考：执行完指令3后，另一个进程开始上CPU运行。

注意：另一个进程在运行过程中也会使用各个寄存器

```
int x = 1;
x++;
.....
```



内存

PSW: 程序状态
字寄存器

PC: 指令4的地址

IR: 指令3...

通用寄存器: 12

.....

程序计数器，存放下一条指令的地址

指令寄存器，存放当前正在执行的指令

其他一些必要信息



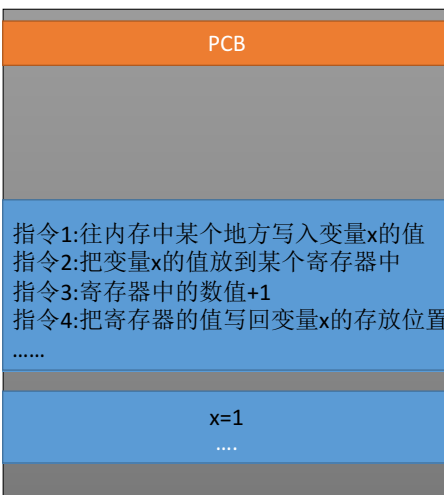
王道考研/CSKAOYAN.COM

知识滚雪球：程序是如何运行的？

思考：执行完指令3后，另一个进程开始上CPU运行。

注意：另一个进程在运行过程中也会使用各个寄存器

```
int x = 1;
x++;
.....
```



内存

PSW: 程序状态
字寄存器

PC: 指令y的地址

IR: 指令x...

通用寄存器: 250

.....

程序计数器，存放下一条指令的地址

指令寄存器，存放当前正在执行的指令

其他一些必要信息



王道考研/CSKAOYAN.COM

