本节内容

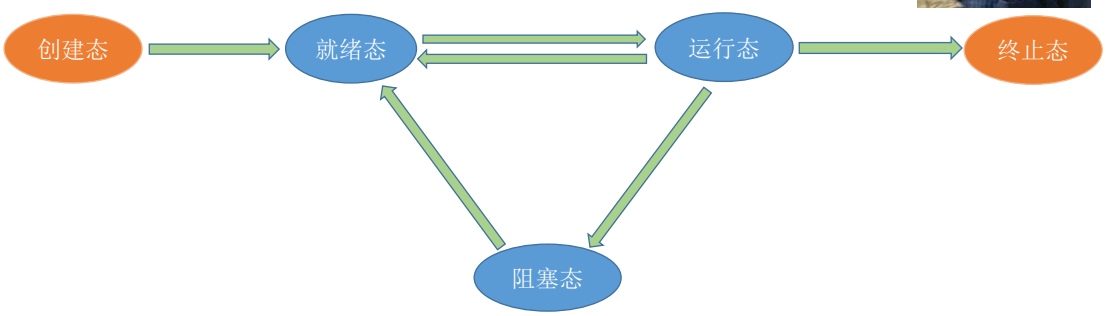
进程控制

1

什么是进程控制？

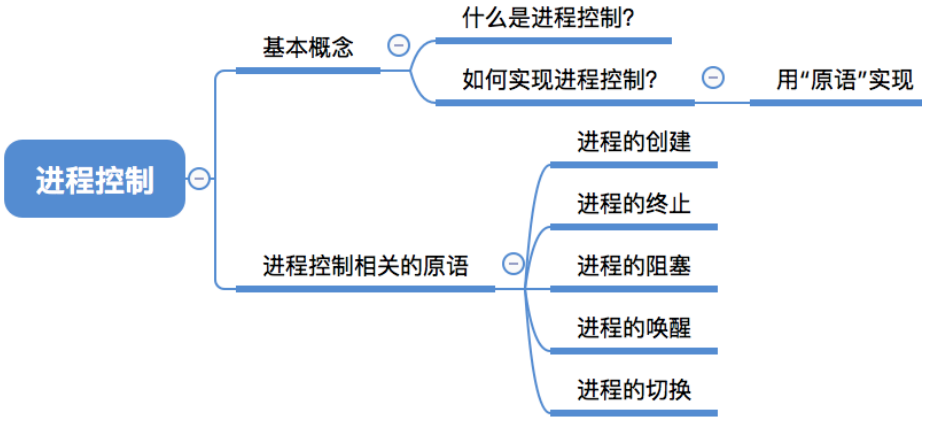
进程控制的主要功能是对系统中的所有进程实施有效的管理，它具有创建新进程、撤销已有进程、实现进程状态转换等功能。

简化理解：反正进程控制就是要实现进程状态转换



2

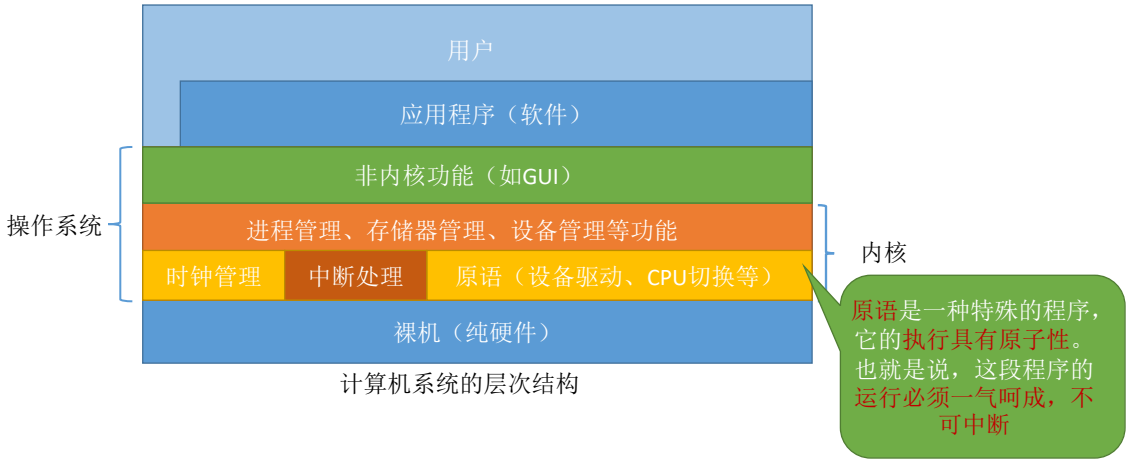
知识总览



3

用“原语”实现

如何实现进程控制？



计算机系统的层次结构

4

如何实现进程控制？

用“原语”实现

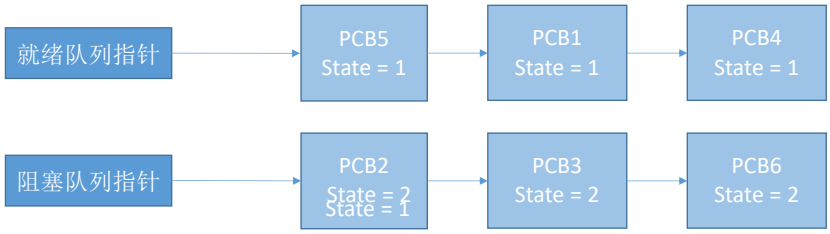
原语的执行具有“原子性”，一气呵成

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

可以用“原语”来实现“一气呵成”啊汪！

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...



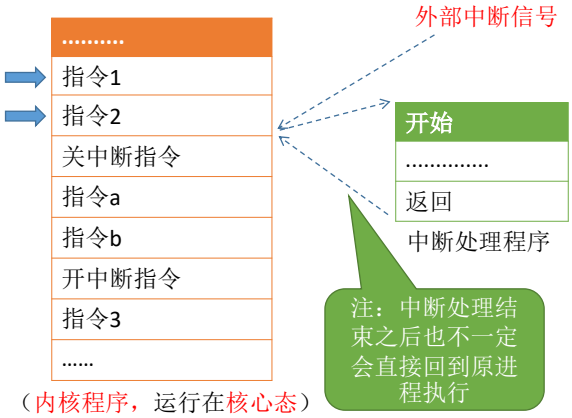
假设此时进程2等待的事件发生，则操作系统中，负责进程控制的内核程序至少需要做这样两件事：
①将PCB2的 state 设为 1
②将PCB2从阻塞队列放到就绪队列

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

5

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



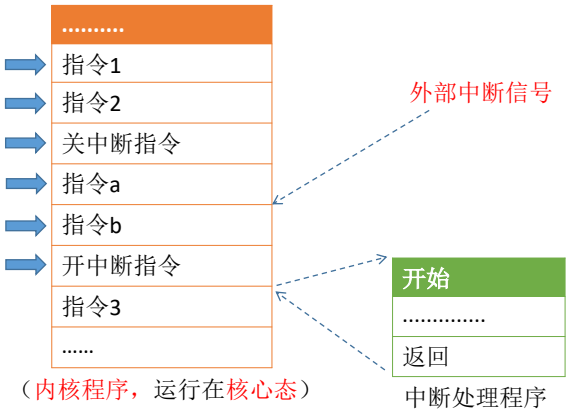
（内核程序，运行在核心态）

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。

6

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



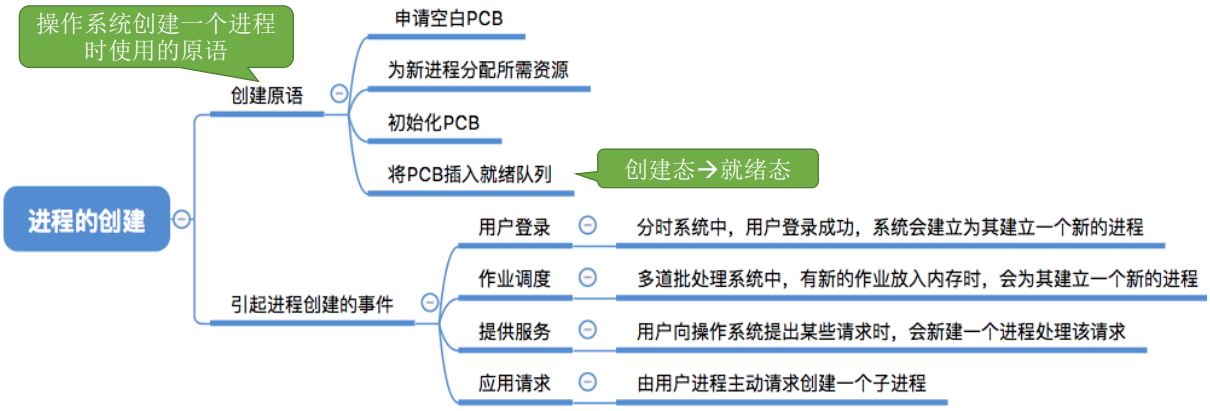
CPU执行了关中断指令之后，就不再例行检查中断信号，直到执行开中断指令之后才会恢复检查。

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”

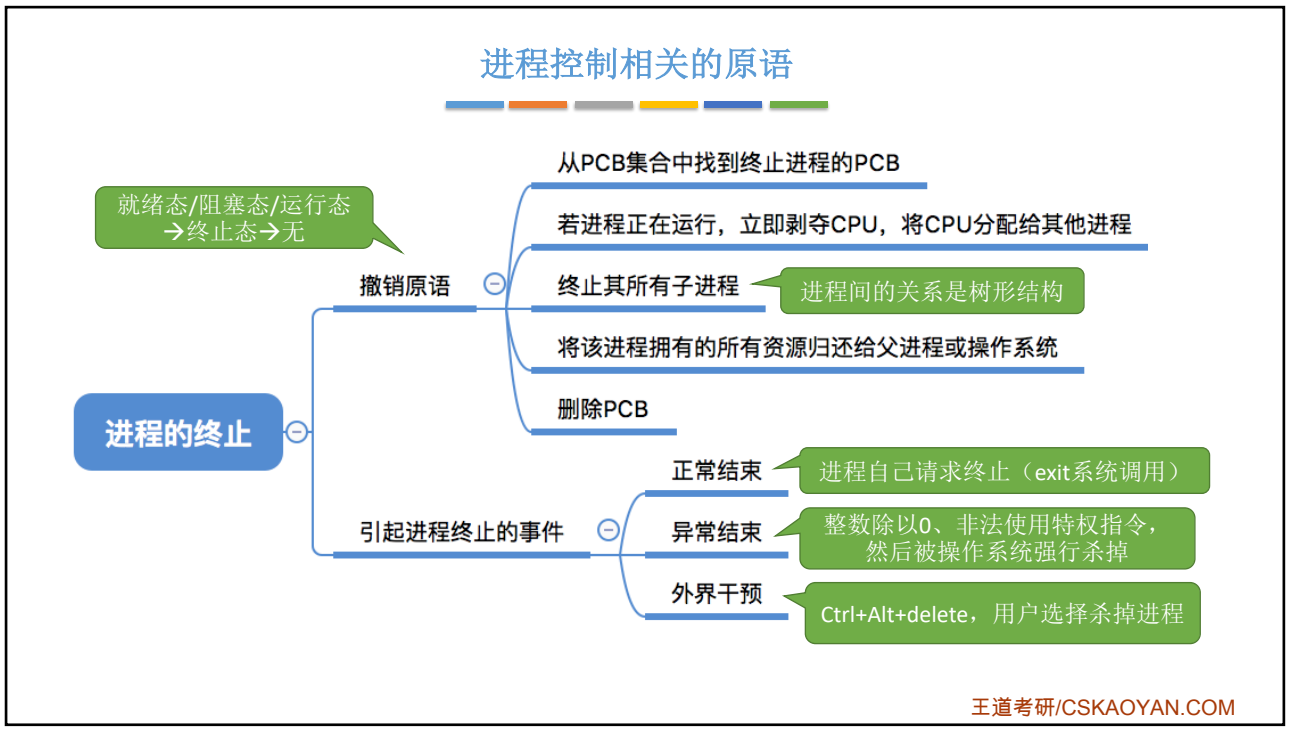
思考：如果这两个特权指令允许用户程序使用的话，会发生什么情况？

7

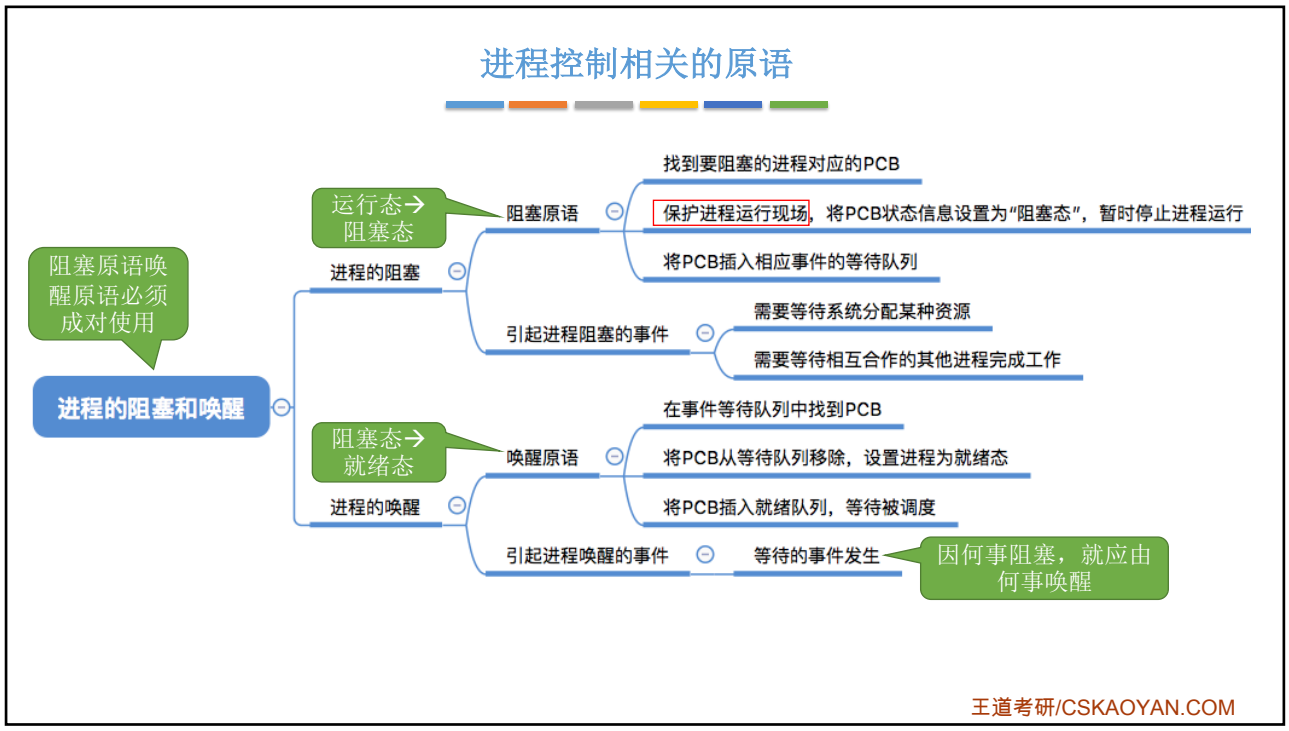
进程控制相关的原语



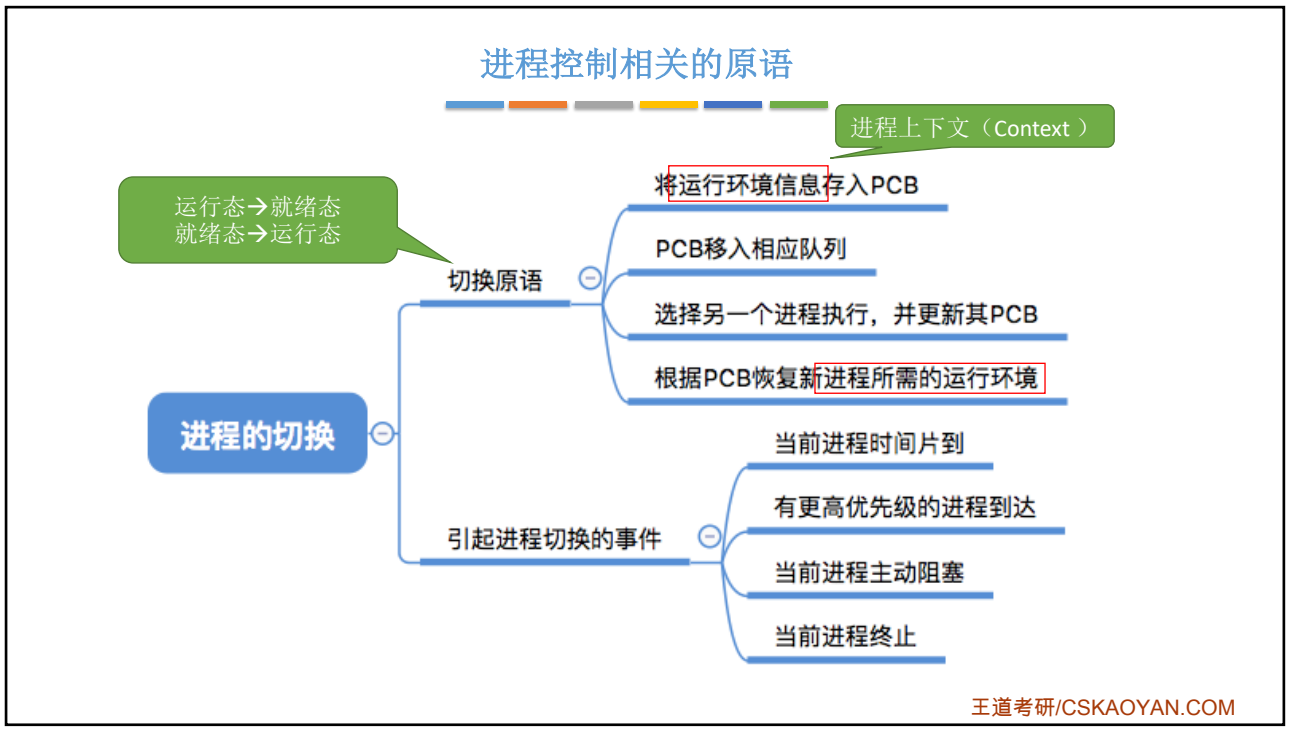
8


进程控制相关的原语
进程的终止
撤销原语
从PCB集合中找到终止进程的PCB
若进程正在运行，立即剥夺CPU，将CPU分配给其他进程
终止其所有子进程
进程间的关系是树形结构
将该进程拥有的所有资源归还给父进程或操作系统
删除PCB
就绪态/阻塞态/运行态→终止态→无
引起进程终止的事件
正常结束
进程自己请求终止（exit系统调用）
异常结束
整数除以0、非法使用特权指令，然后被操作系统强行杀掉
外界干预
Ctrl+Alt+delete，用户选择杀掉进程
王道考研/CSKAOYAN.COM

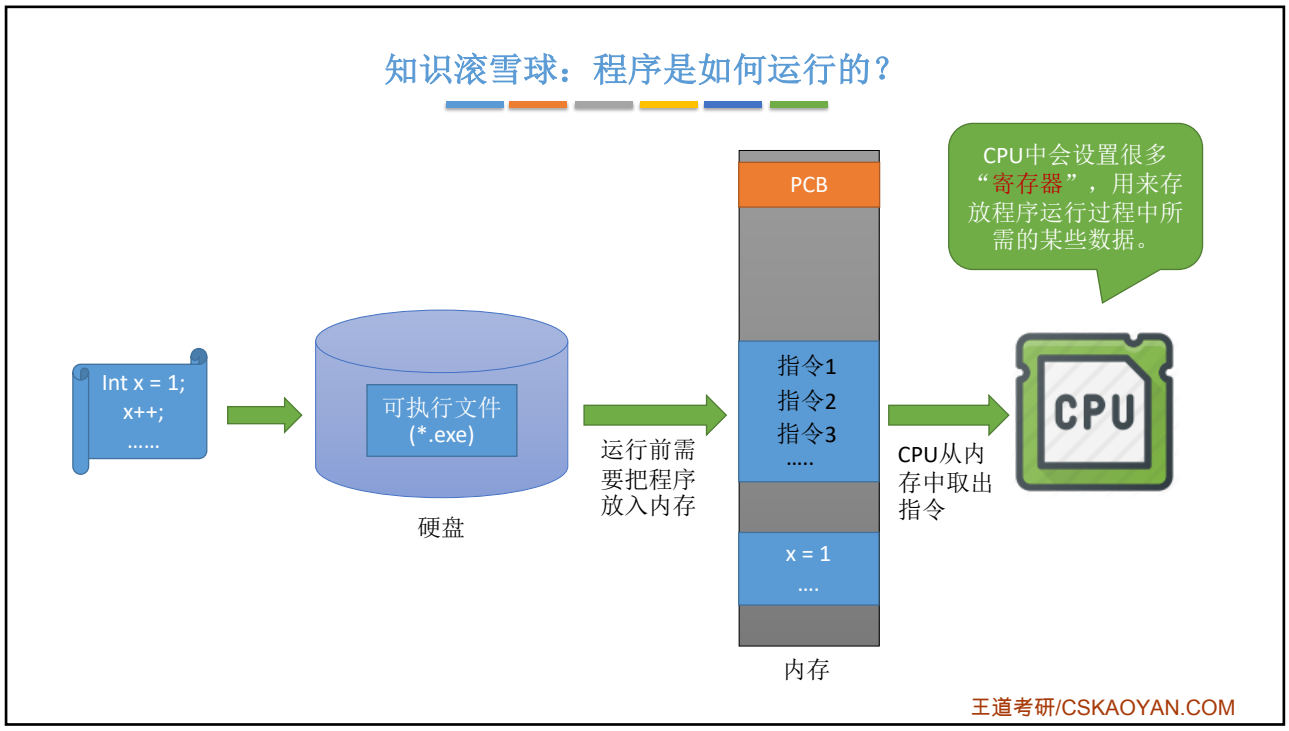
9


进程控制相关的原语
进程的阻塞和唤醒
阻塞原语唤醒原语必须成对使用
进程的阻塞
运行态→阻塞态
阻塞原语
找到要阻塞的进程对应的PCB
保护进程运行现场，将PCB状态信息设置为"阻塞态"，暂时停止进程运行
将PCB插入相应事件的等待队列
引起进程阻塞的事件
需要等待系统分配某种资源
需要等待相互合作的其他进程完成工作
进程的唤醒
阻塞态→就绪态
唤醒原语
在事件等待队列中找到PCB
将PCB从等待队列移除，设置进程为就绪态
将PCB插入就绪队列，等待被调度
引起进程唤醒的事件
等待的事件发生
因何事阻塞，就应由何事唤醒
王道考研/CSKAOYAN.COM

10

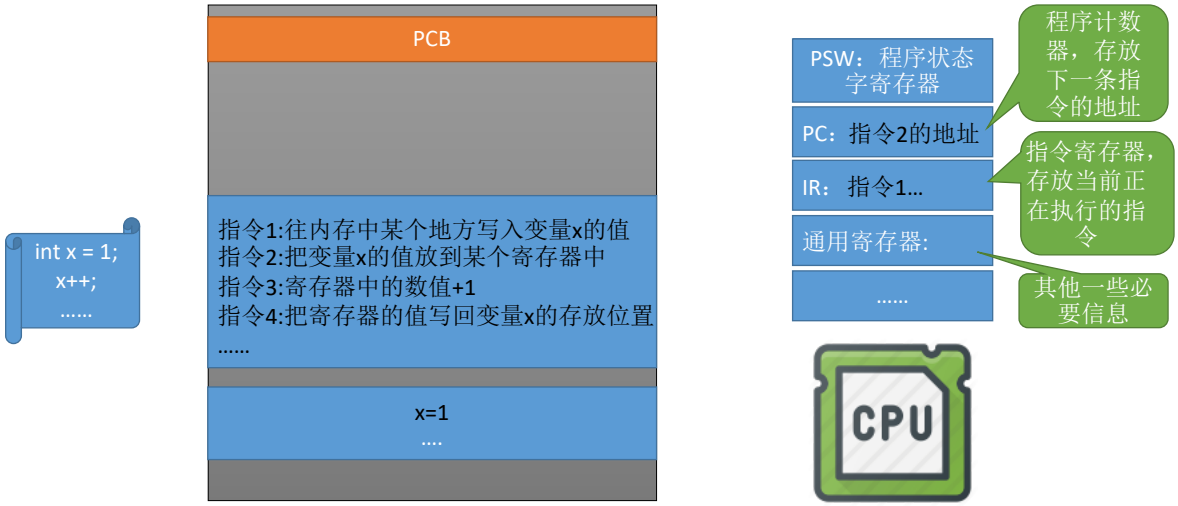

进程控制相关的原语
进程上下文（Context）
运行态→就绪态
就绪态→运行态
切换原语
将运行环境信息存入PCB
PCB移入相应队列
选择另一个进程执行，并更新其PCB
根据PCB恢复新进程所需的运行环境
进程的切换
当前进程时间片到
有更高优先级的进程到达
引起进程切换的事件
当前进程主动阻塞
当前进程终止
王道考研/CSKAOYAN.COM

11


知识滚雪球：程序是如何运行的？
CPU中会设置很多“寄存器”，用来存放程序运行过程中所需的某些数据。
PCB
Int x = 1;
x++;
……
可执行文件
(*.exe)
硬盘
运行前需要把程序放入内存
指令1
指令2
指令3
……
x = 1
……
内存
CPU从内存中取出指令
CPU
王道考研/CSKAOYAN.COM

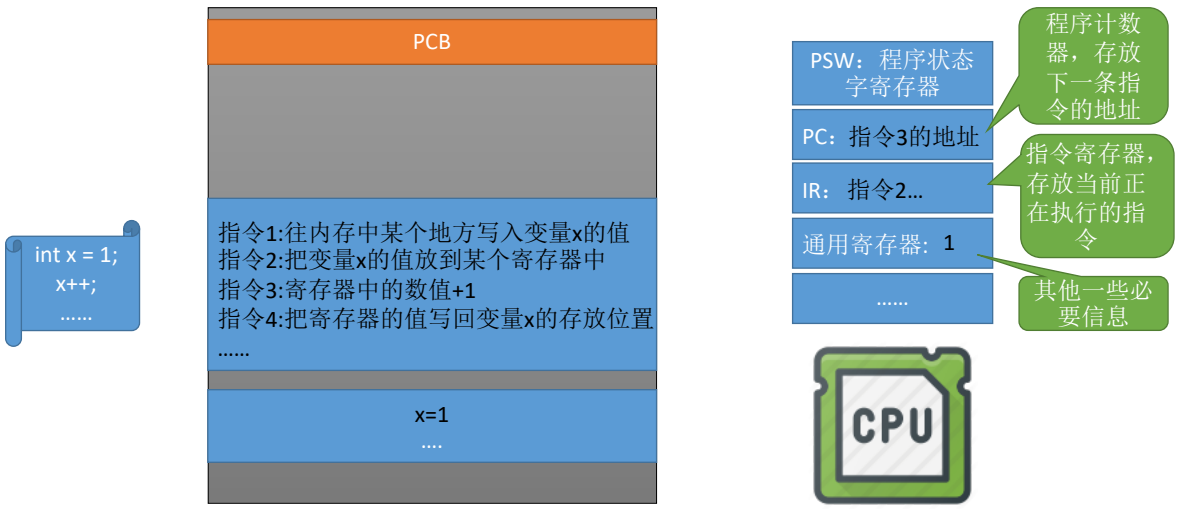
12

知识滚雪球：程序是如何运行的？

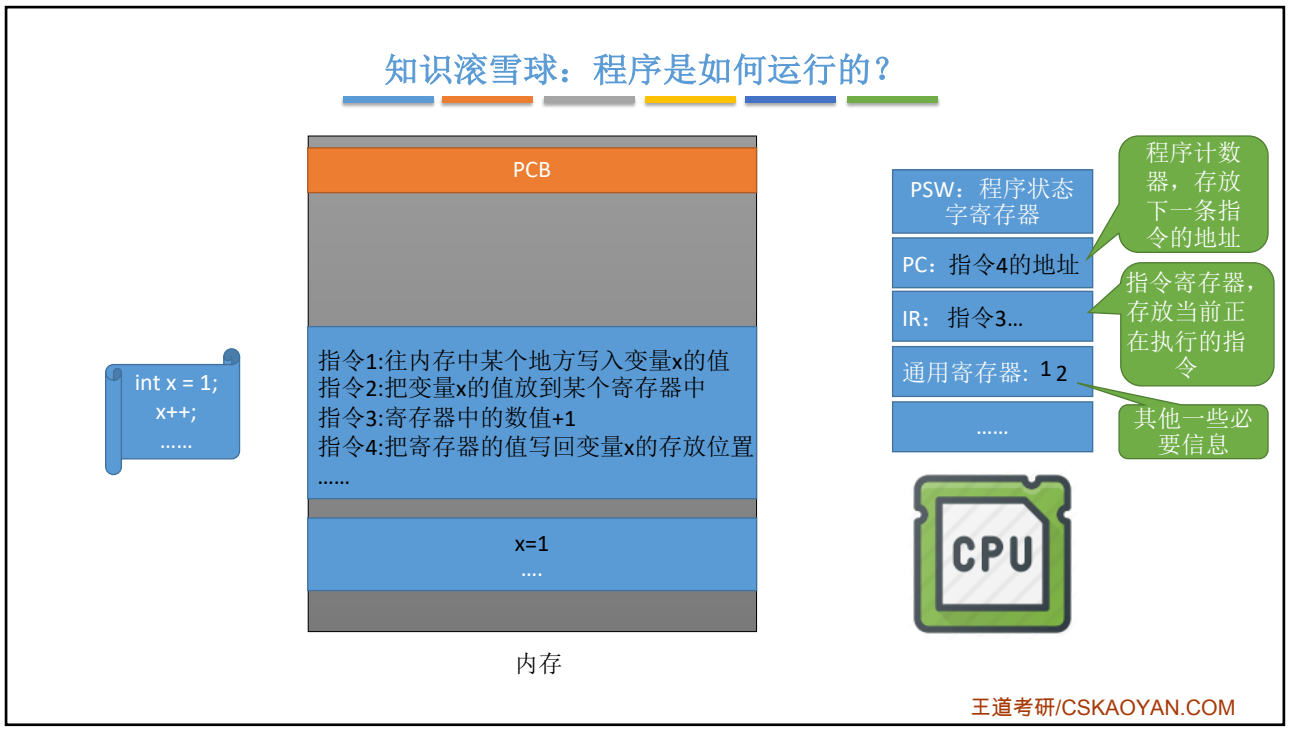


13

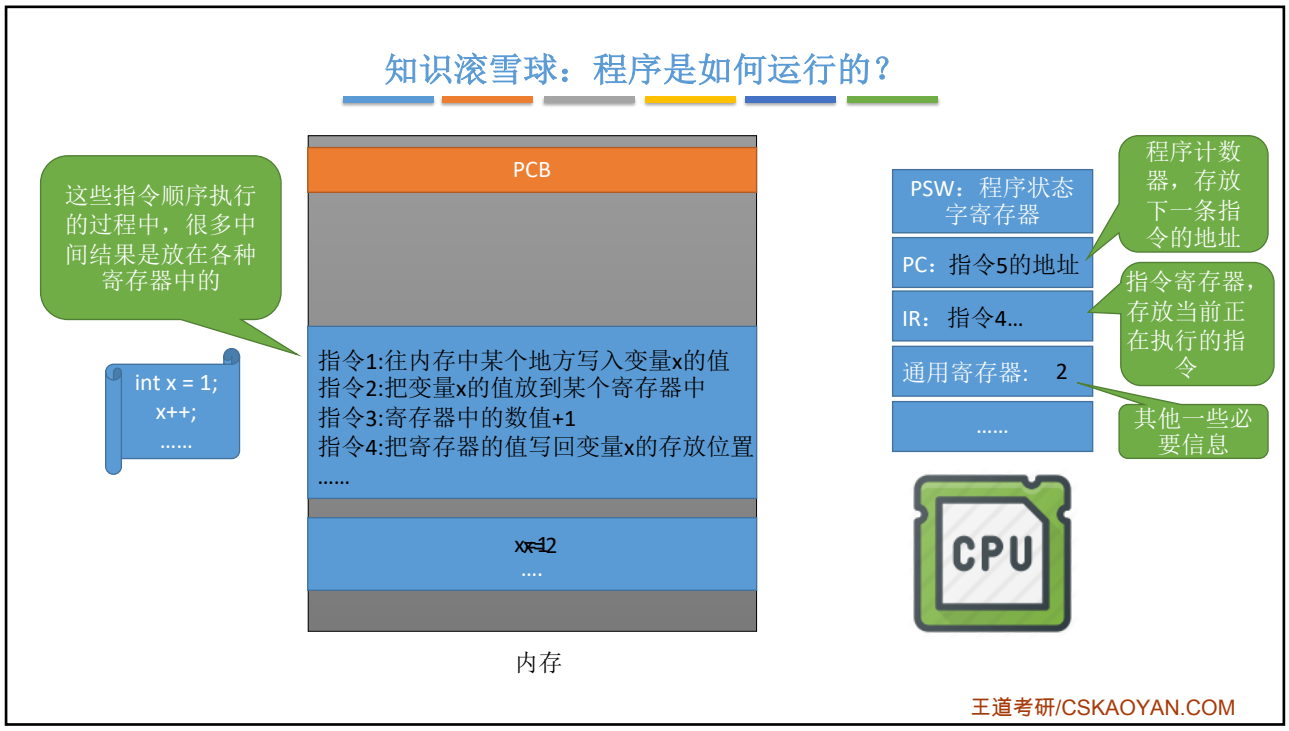
知识滚雪球：程序是如何运行的？



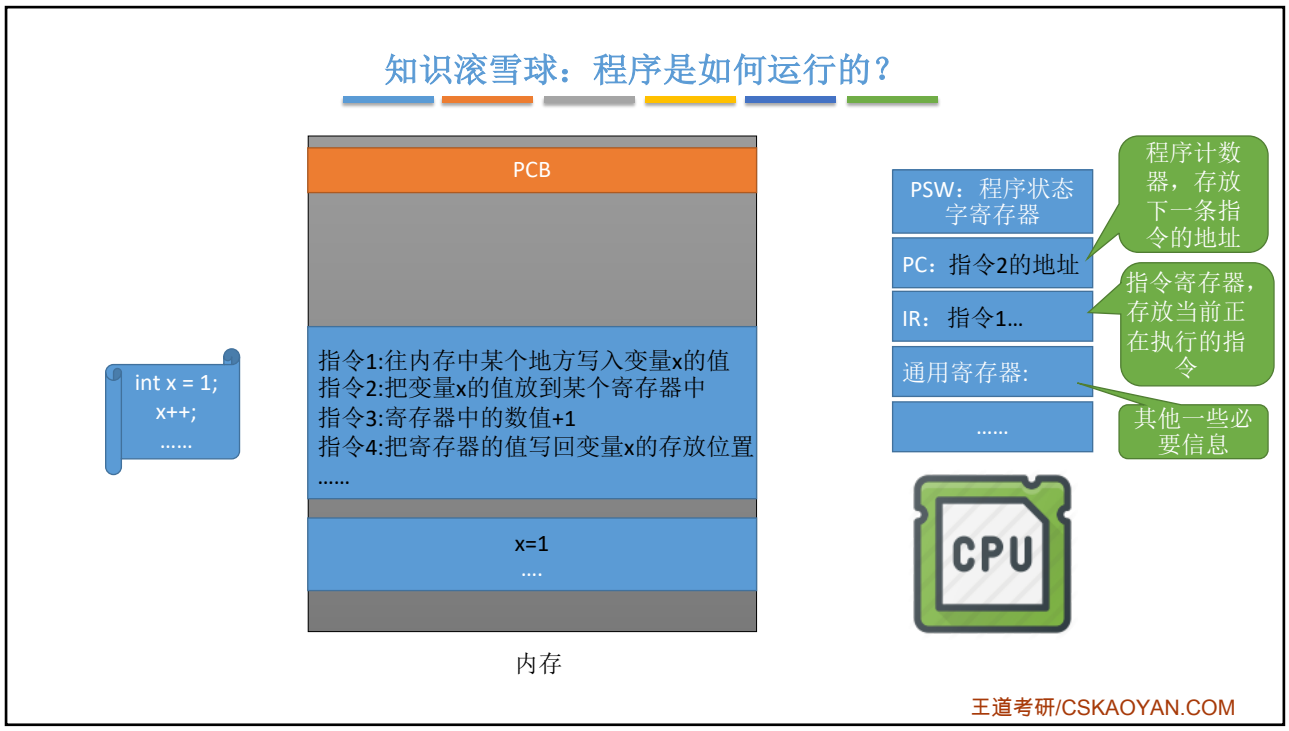
14


知识滚雪球：程序是如何运行的？
PCB
指令1:往内存中某个地方写入变量x的值
指令2:把变量x的值放到某个寄存器中
指令3:寄存器中的数值+1
指令4:把寄存器的值写回变量x的存放位置
.....
x=1
....
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：1 2
.....
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
CPU
王道考研/CSKAOYAN.COM

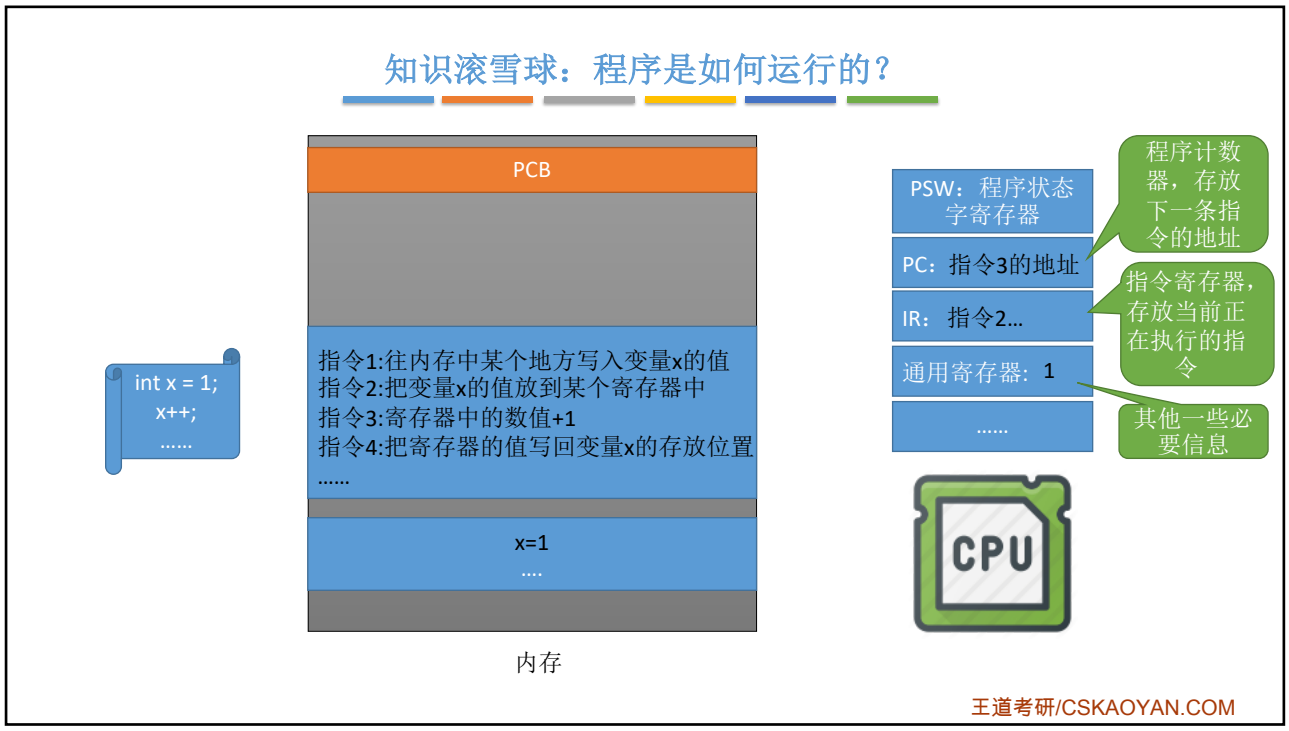
15


知识滚雪球：程序是如何运行的？
这些指令顺序执行的过程中，很多中间结果是放在各种寄存器中的
PCB
指令1:往内存中某个地方写入变量x的值
指令2:把变量x的值放到某个寄存器中
指令3:寄存器中的数值+1
指令4:把寄存器的值写回变量x的存放位置
.....
x=2
....
int x = 1;
x++;
.....
内存
PSW：程序状态字寄存器
PC：指令5的地址
IR： 指令4...
通用寄存器： 2
.....
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
CPU
王道考研/CSKAOYAN.COM

16


知识滚雪球：程序是如何运行的？
PCB
指令1:往内存中某个地方写入变量x的值
指令2:把变量x的值放到某个寄存器中
指令3:寄存器中的数值+1
指令4:把寄存器的值写回变量x的存放位置
.....
x=1
....
int x = 1;
x++;
.....
内存
PSW：程序状态字寄存器
PC：指令2的地址
IR： 指令1...
通用寄存器:
.....
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
CPU
王道考研/CSKAOYAN.COM

17


知识滚雪球：程序是如何运行的？
PCB
指令1:往内存中某个地方写入变量x的值
指令2:把变量x的值放到某个寄存器中
指令3:寄存器中的数值+1
指令4:把寄存器的值写回变量x的存放位置
.....
x=1
....
int x = 1;
x++;
.....
内存
PSW：程序状态字寄存器
PC：指令3的地址
IR： 指令2...
通用寄存器: 1
.....
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
CPU
王道考研/CSKAOYAN.COM

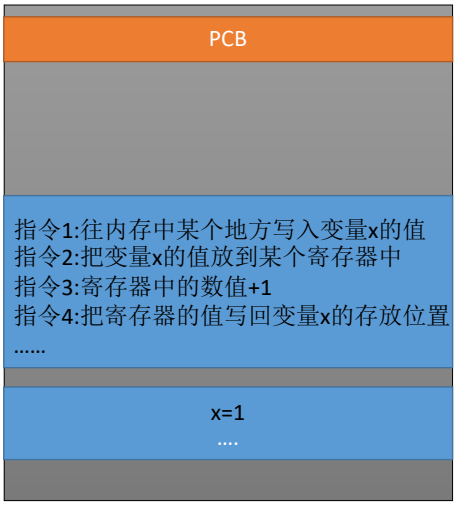
18

知识滚雪球：程序是如何运行的？

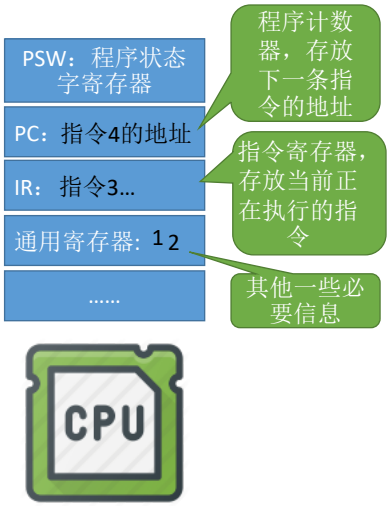
思考：执行完指令3后，另一个进程开始上CPU运行。

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



19

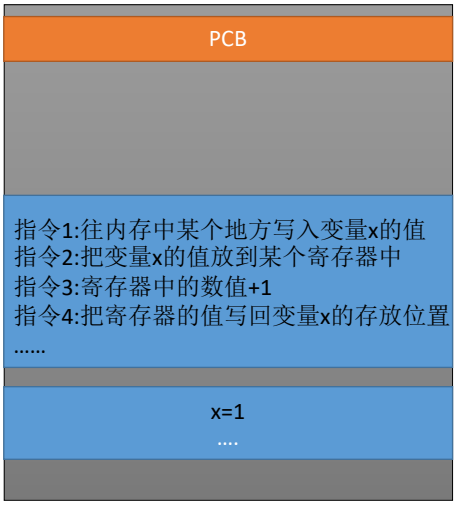
知识滚雪球：程序是如何运行的？

思考：执行完指令3后，另一个进程开始上CPU运行。

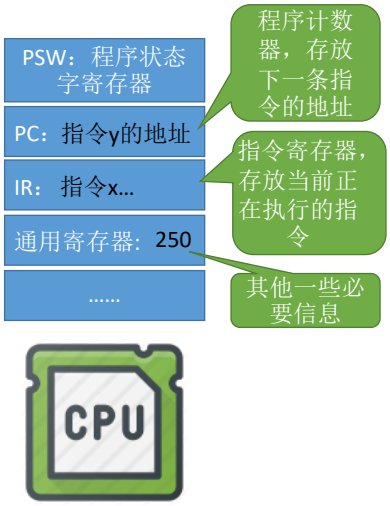
注意：另一个进程在运行过程中也会使用各个寄存器

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

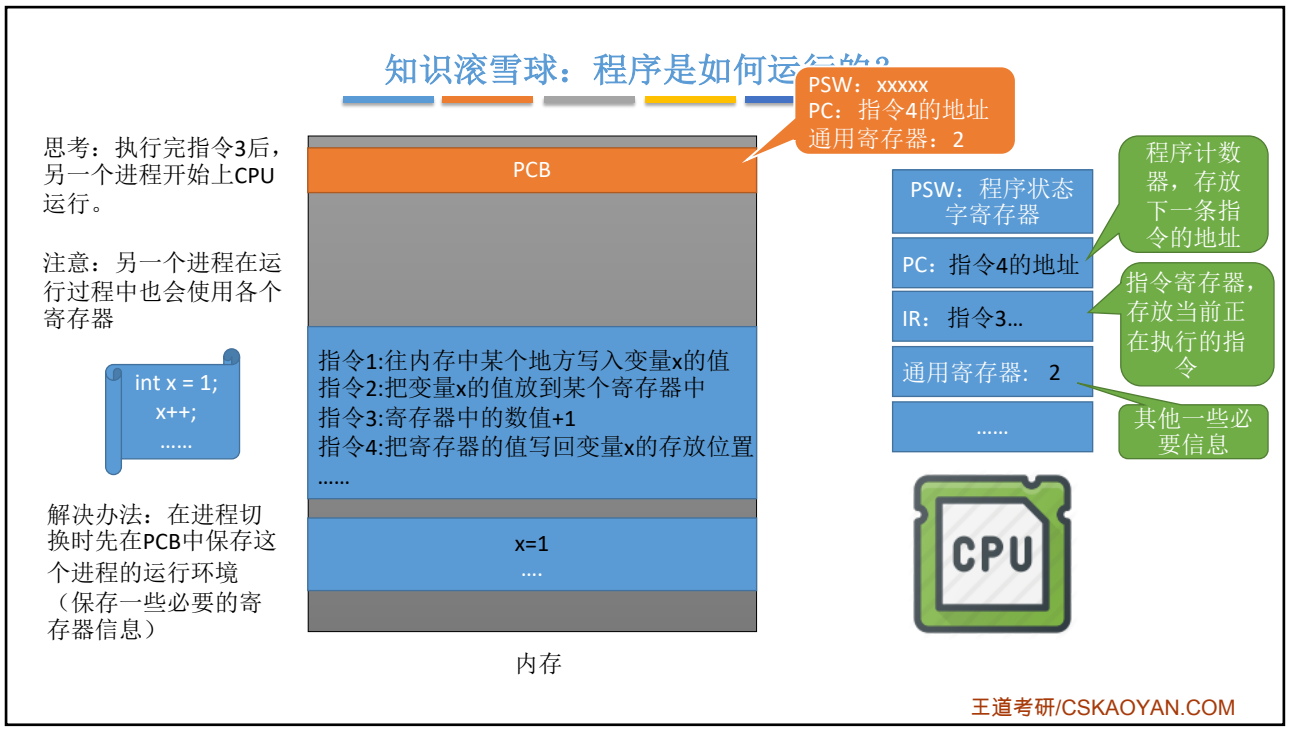
灵魂拷问：之后还怎么切换回之前的进程？？？？



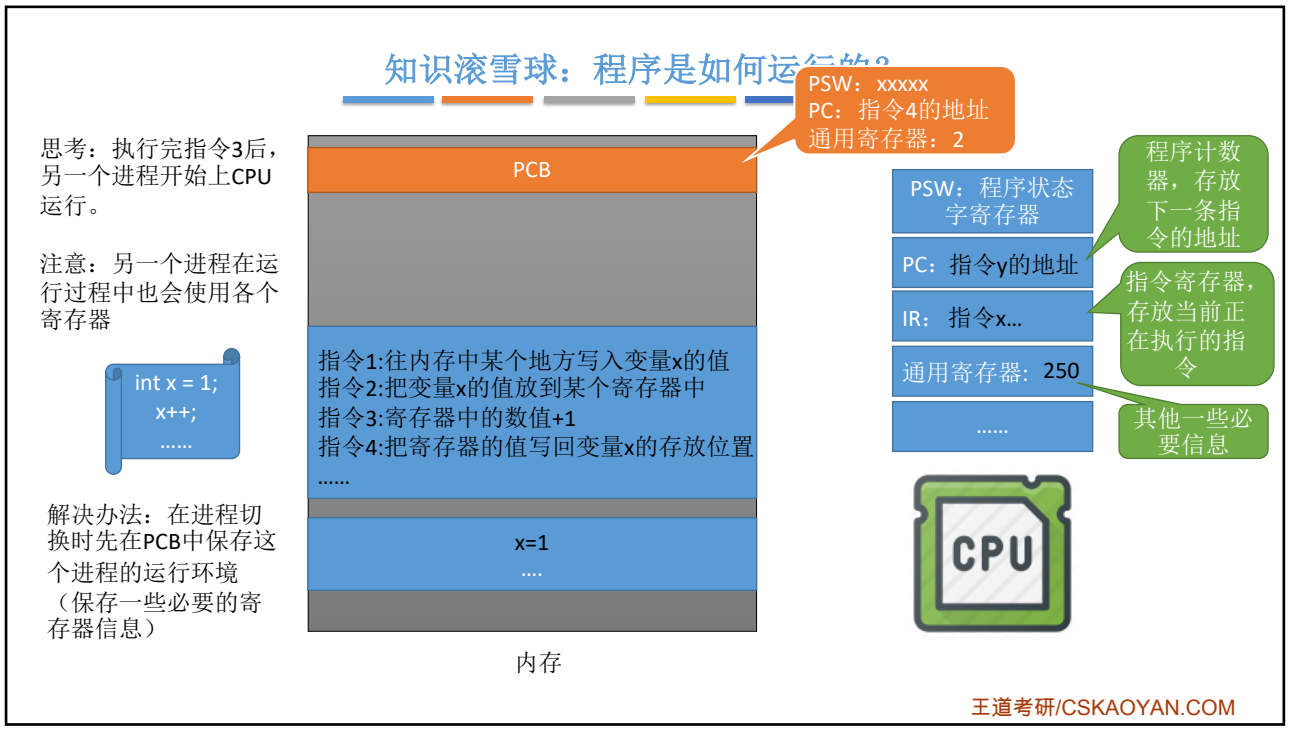
内存



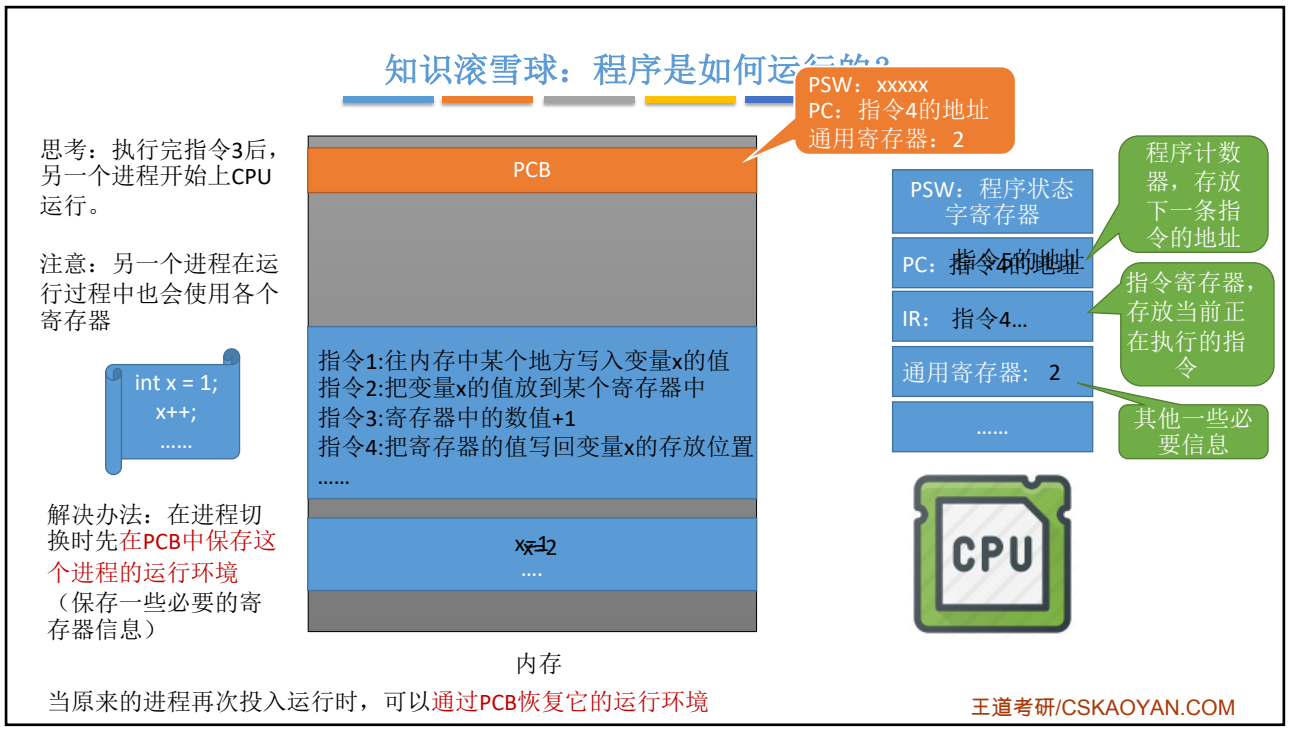
20


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++;
.....
解决办法：在进程切换时先在PCB中保存这个进程的运行环境（保存一些必要的寄存器信息）
PCB
指令1:往内存中某个地方写入变量x的值
指令2:把变量x的值放到某个寄存器中
指令3:寄存器中的数值+1
指令4:把寄存器的值写回变量x的存放位置
.....
x=1
....
内存
PSW：xxxxx
PC：指令4的地址
通用寄存器：2
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：2
.....
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
CPU
王道考研/CSKAOYAN.COM

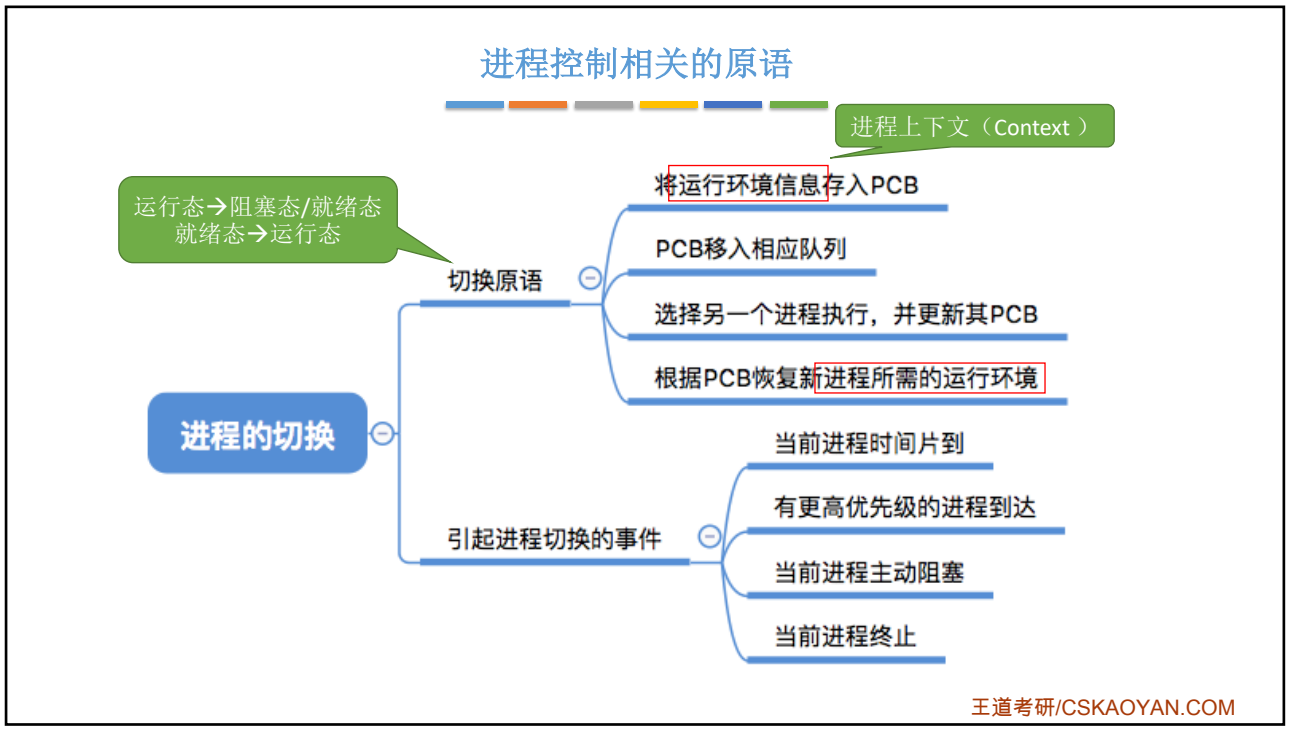
21


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++;
.....
解决办法：在进程切换时先在PCB中保存这个进程的运行环境（保存一些必要的寄存器信息）
PCB
指令1:往内存中某个地方写入变量x的值
指令2:把变量x的值放到某个寄存器中
指令3:寄存器中的数值+1
指令4:把寄存器的值写回变量x的存放位置
.....
x=1
....
内存
PSW：xxxxx
PC：指令4的地址
通用寄存器：2
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
CPU
王道考研/CSKAOYAN.COM

22

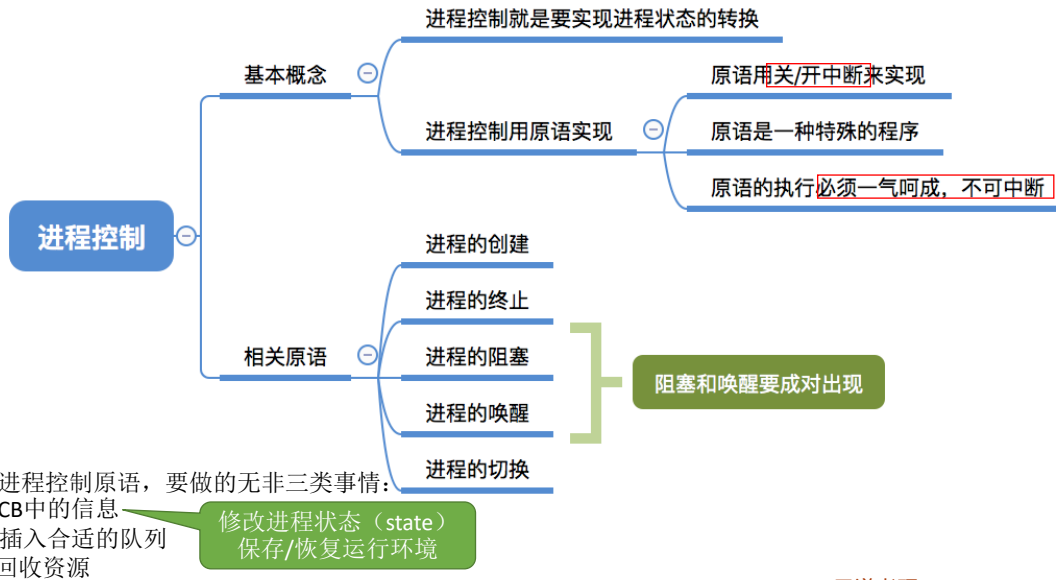

知识滚雪球：程序是如何运行的？
思考：执行完指令3后，另一个进程开始上CPU运行。
注意：另一个进程在运行过程中也会使用各个寄存器
int x = 1;
x++;
.....
解决办法：在进程切换时先在PCB中保存这个进程的运行环境（保存一些必要的寄存器信息）
PCB
PSW：xxxxx
PC：指令4的地址
通用寄存器：2
指令1:往内存中某个地方写入变量x的值
指令2:把变量x的值放到某个寄存器中
指令3:寄存器中的数值+1
指令4:把寄存器的值写回变量x的存放位置
.....
x=2
....
内存
PSW：程序状态字寄存器
PC：指令4的地址
IR： 指令4...
通用寄存器： 2
.....
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
CPU
当原来的进程再次投入运行时，可以通过PCB恢复它的运行环境
王道考研/CSKAOYAN.COM

23


进程控制相关的原语
进程的切换
切换原语
进程上下文（Context）
将运行环境信息存入PCB
PCB移入相应队列
选择另一个进程执行，并更新其PCB
根据PCB恢复新进程所需的运行环境
运行态→阻塞态/就绪态
就绪态→运行态
引起进程切换的事件
当前进程时间片到
有更高优先级的进程到达
当前进程主动阻塞
当前进程终止
王道考研/CSKAOYAN.COM

24

知识回顾与重要考点



无论哪个进程控制原语，要做的无非三类事情：

1. 更新PCB中的信息
2. 将PCB插入合适的队列
3. 分配/回收资源

修改进程状态（state）
保存/恢复运行环境

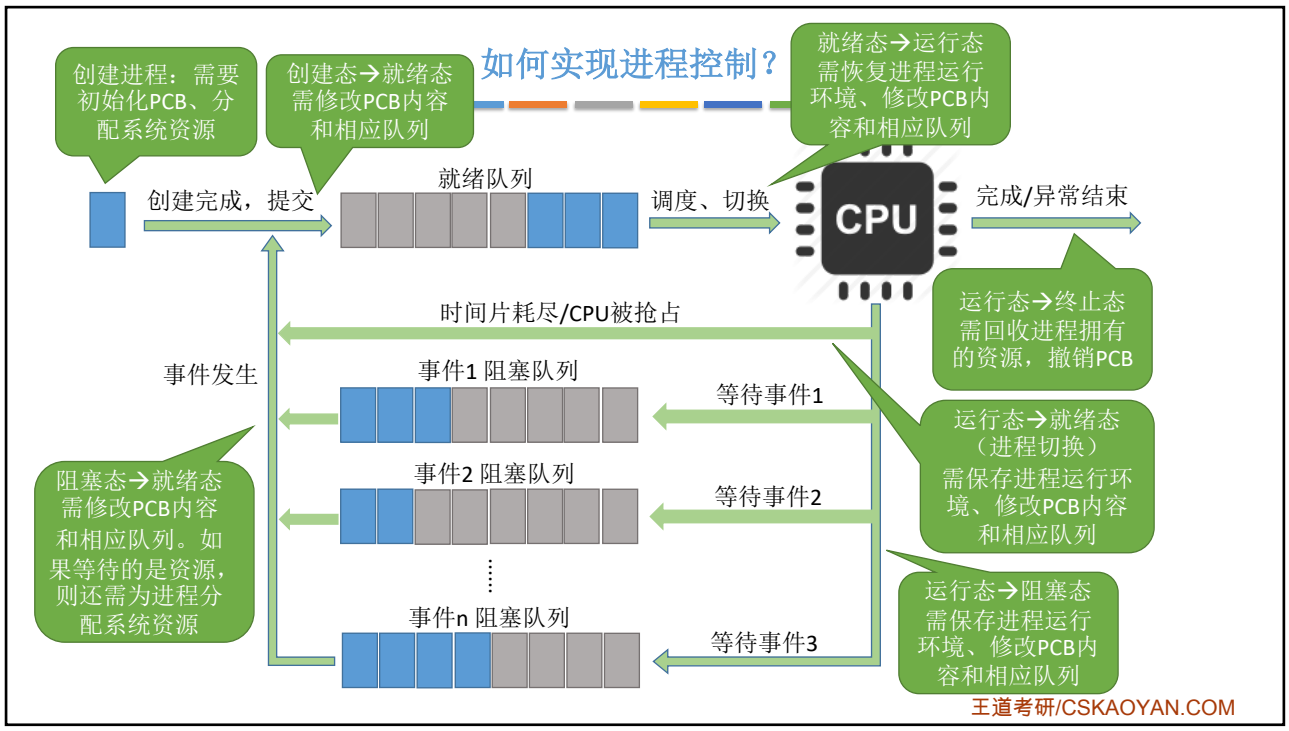
25

进程控制相关的原语

学习技巧：进程控制会导致进程状态的转换。无论哪个进程控制原语，要做的无非三类事情：

1. 更新PCB中的信息
   a. 所有的进程控制原语一定都会修改进程状态标志
   b. 剥夺当前运行进程的CPU使用权必然需要保存其运行环境
   c. 某进程开始运行前必然要恢复期运行环境
2. 将PCB插入合适的队列
3. 分配/回收资源

26


如何实现进程控制？
创建进程：需要初始化PCB、分配系统资源
创建态→就绪态需修改PCB内容和相应队列
就绪态→运行态需恢复进程运行环境、修改PCB内容和相应队列
创建完成，提交
就绪队列
调度、切换
CPU
完成/异常结束
运行态→终止态需回收进程拥有的资源，撤销PCB
时间片耗尽/CPU被抢占
事件发生
事件1 阻塞队列
等待事件1
运行态→就绪态（进程切换）需保存进程运行环境、修改PCB内容和相应队列
事件2 阻塞队列
等待事件2
阻塞态→就绪态需修改PCB内容和相应队列。如果等待的是资源，则还需为进程分配系统资源
事件n 阻塞队列
等待事件3
运行态→阻塞态需保存进程运行环境、修改PCB内容和相应队列
王道考研/CSKAOYAN.COM

27


@王道论坛
@王道计算机考研备考
@王道咸鱼老师-计算机考研
@王道楼楼老师-计算机考研
@王道计算机考研
等撩
等撩
@王道计算机考研
@王道计算机考研
@王道在线

28