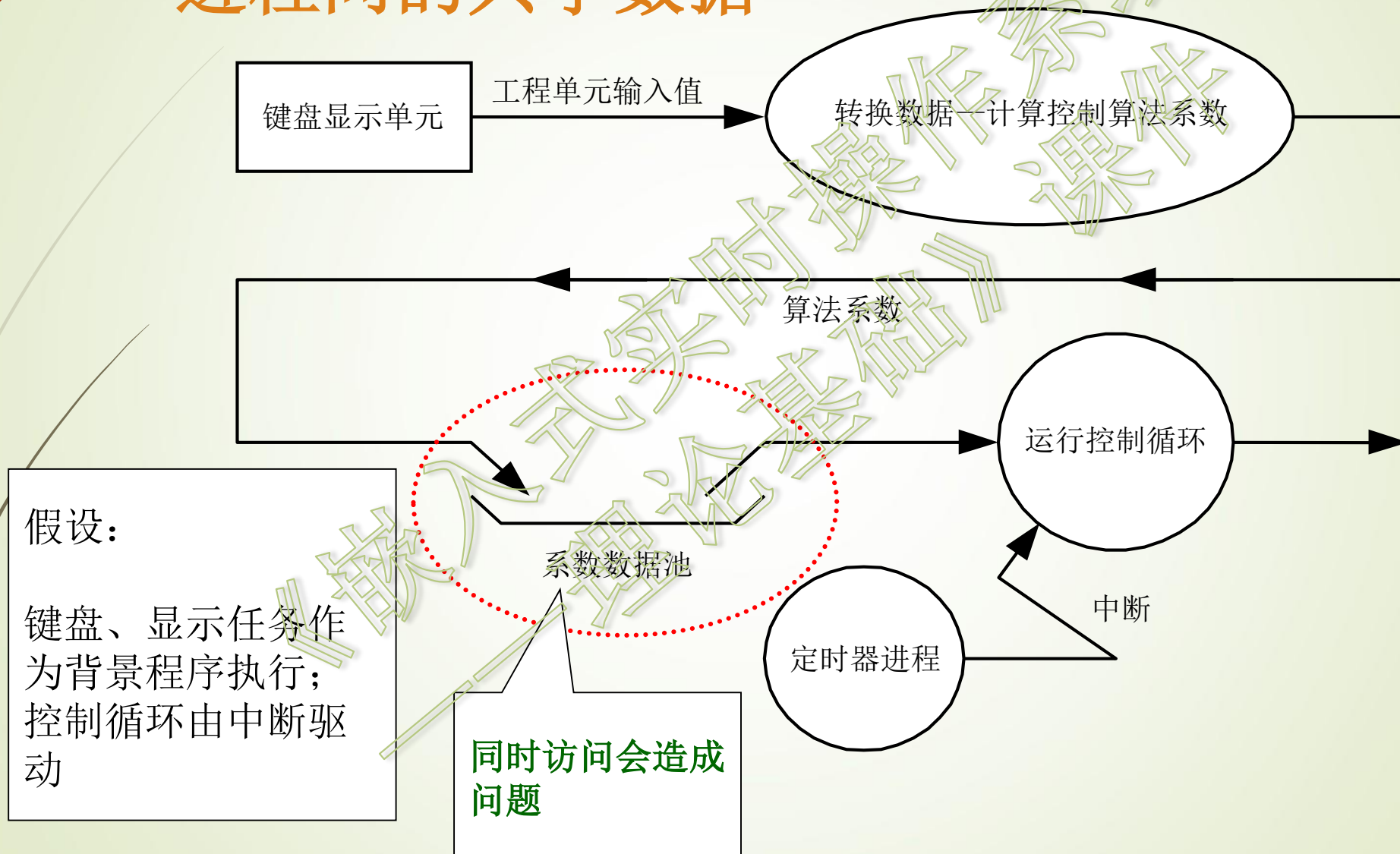


第3章 使用互斥机制控制资源共享

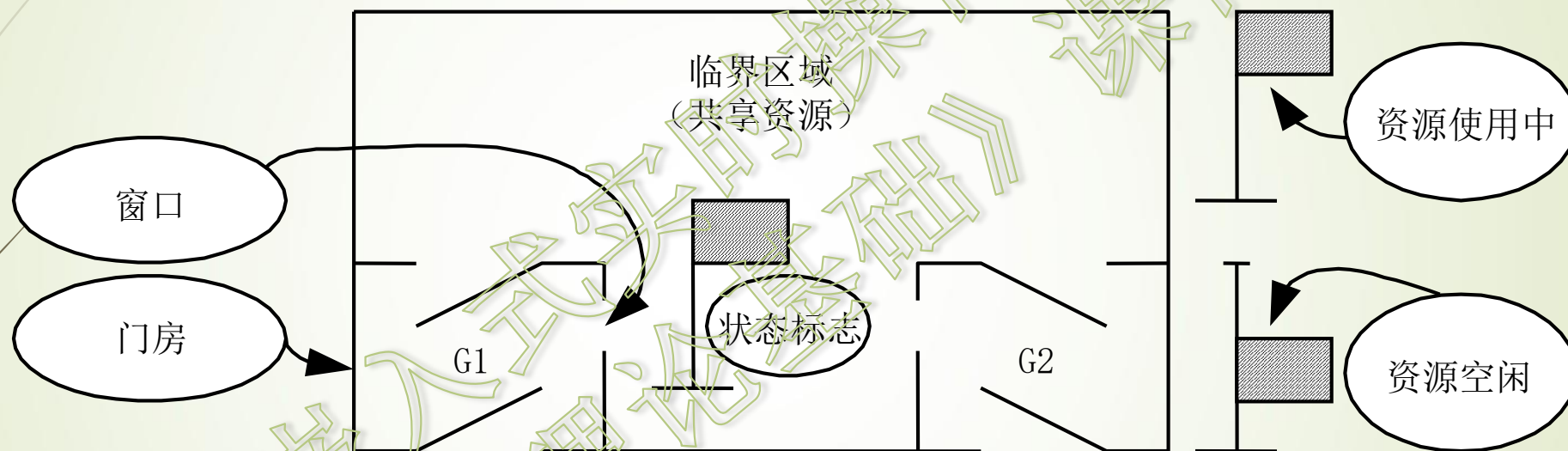
本章内容

- 多任务设计中使用共享资源的问题。
- 互斥的概念。
- 通过程序标志实现互斥。
- 二值/计数信号量的概念和使用。
- 互斥量的概念和相比信号量的优势。
- 信号量和互斥量的问题。
- 如何用简单监视器解决这些问题。

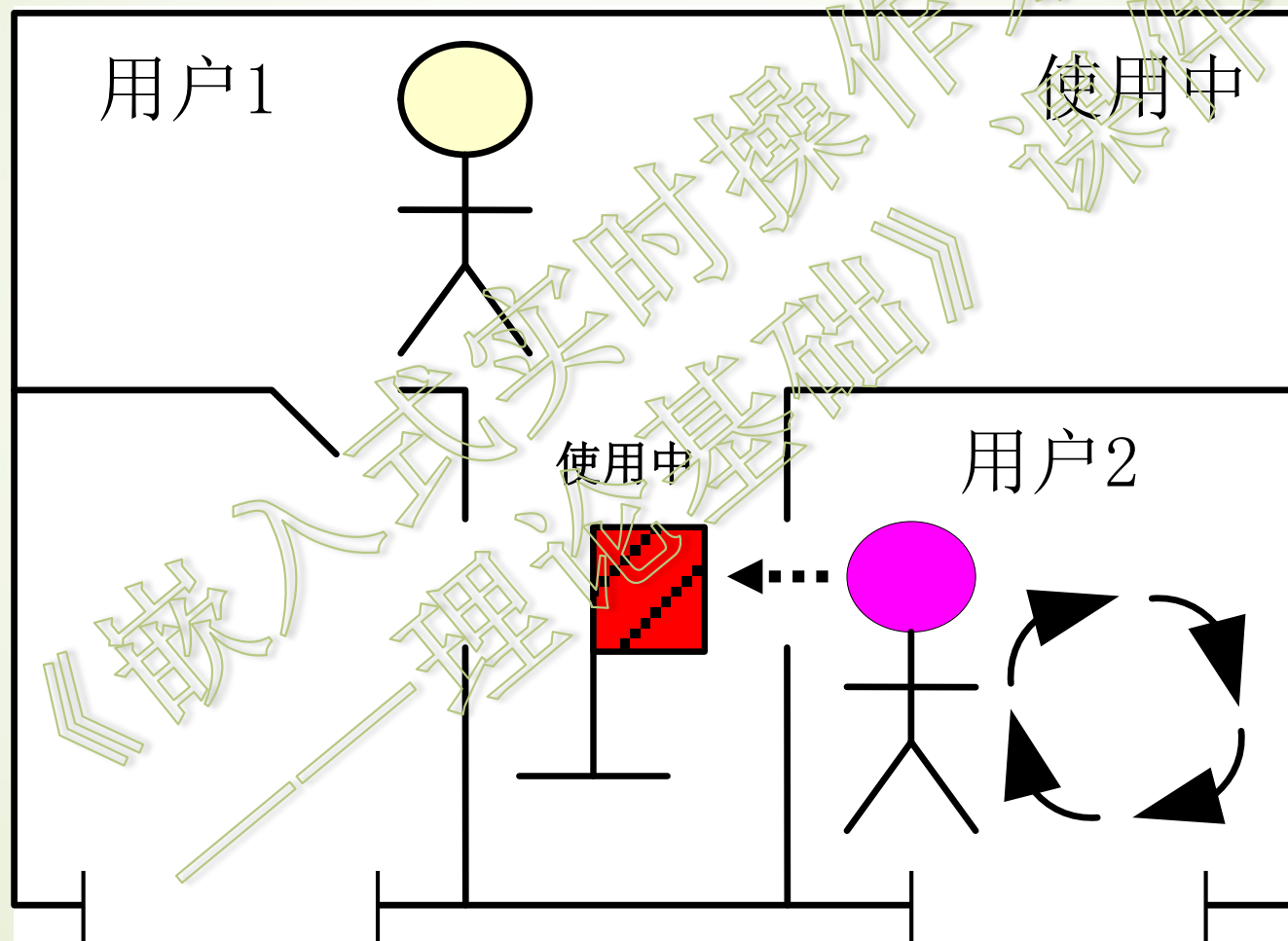
进程间的共享数据



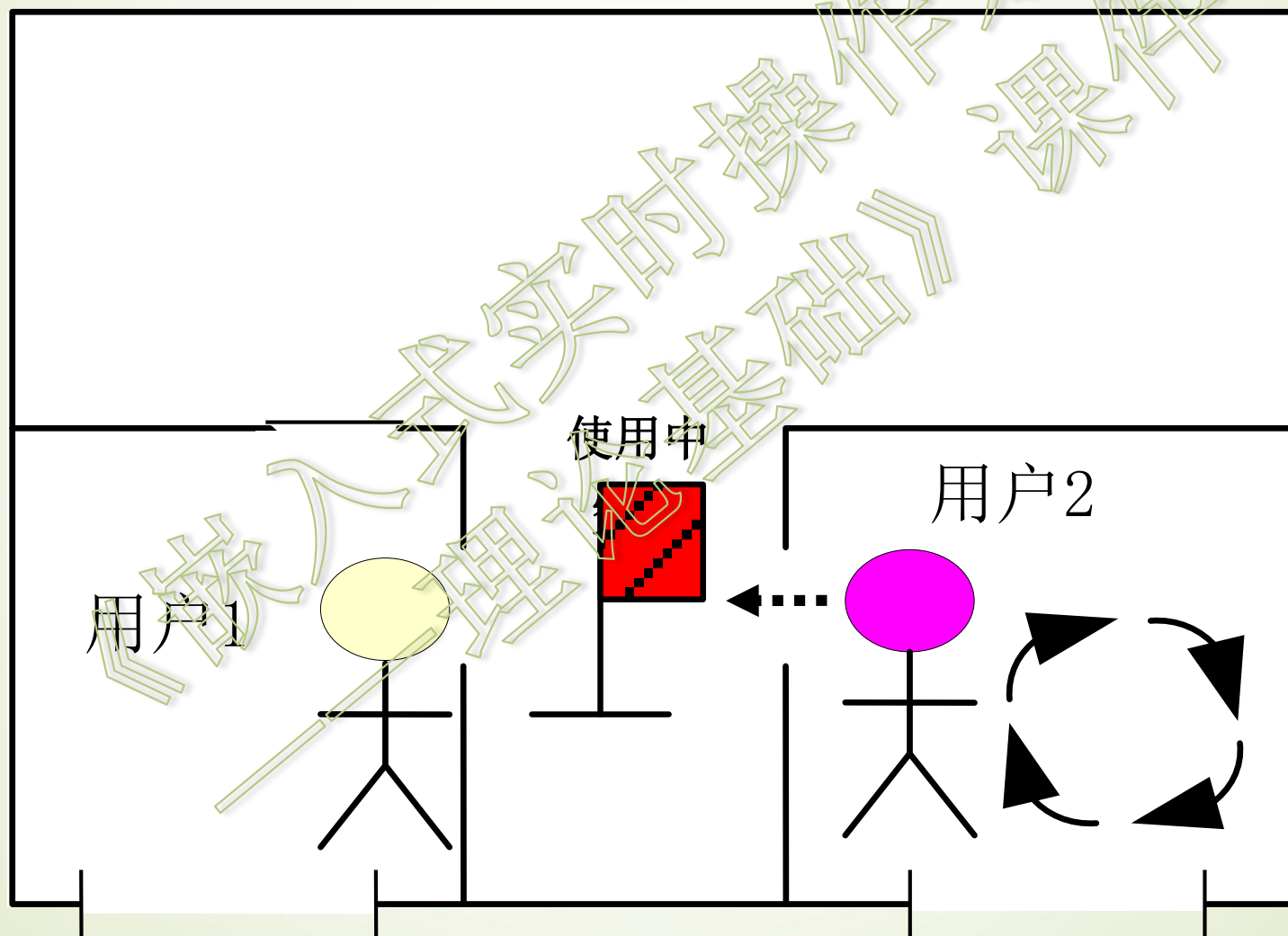
单个标志方法



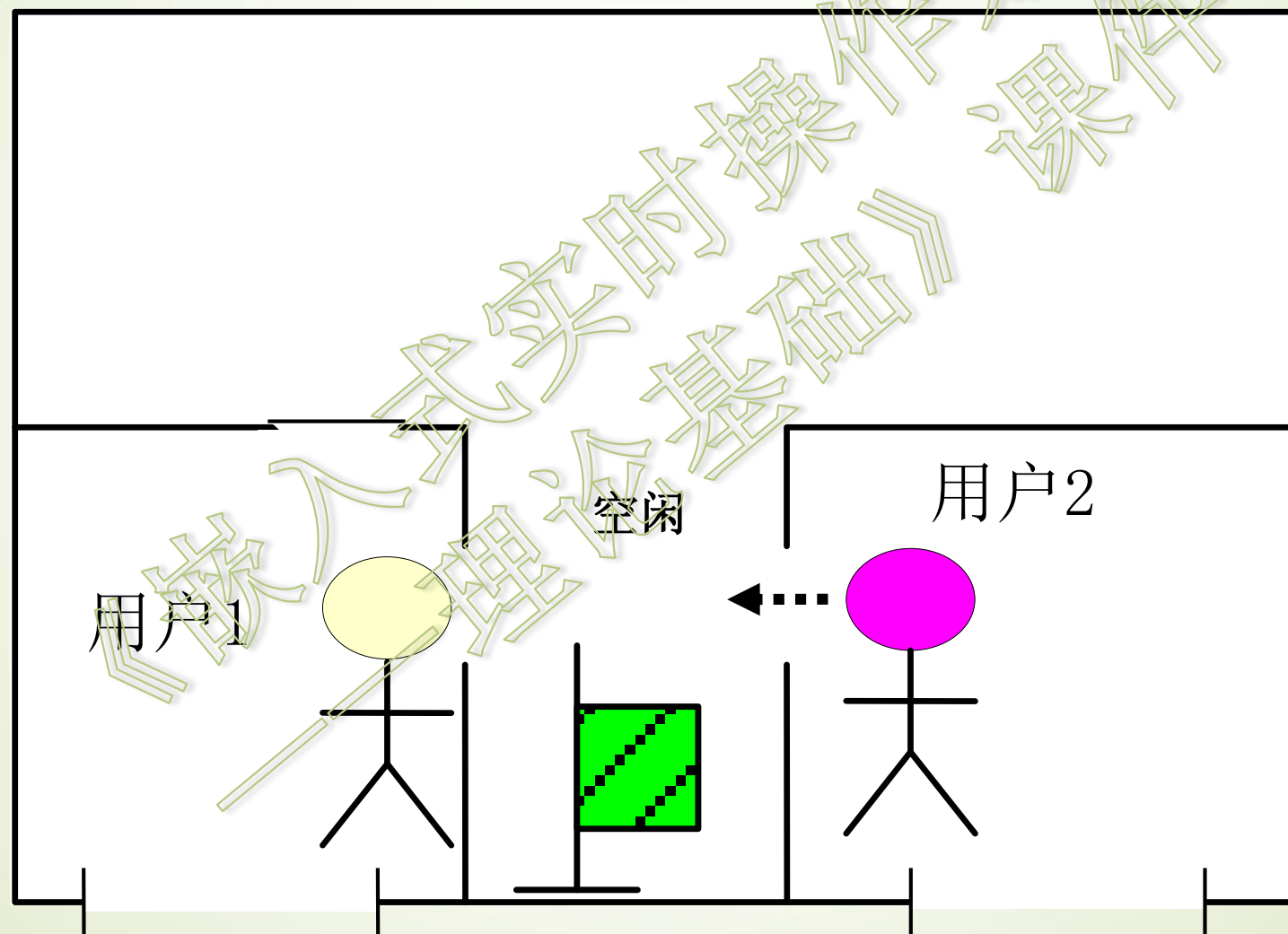
互斥的实现（单标志方法）



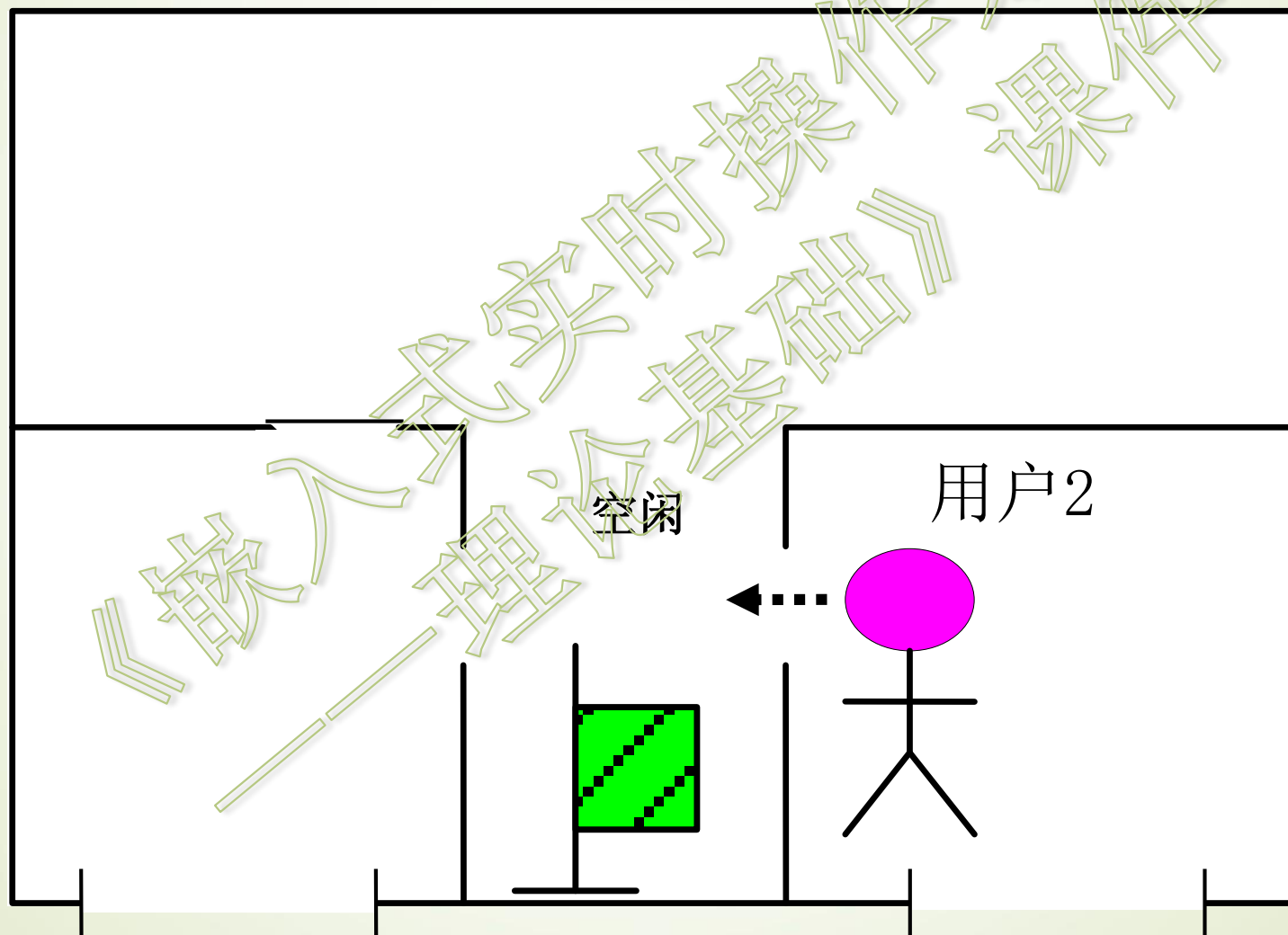
互斥的实现（单标志方法）



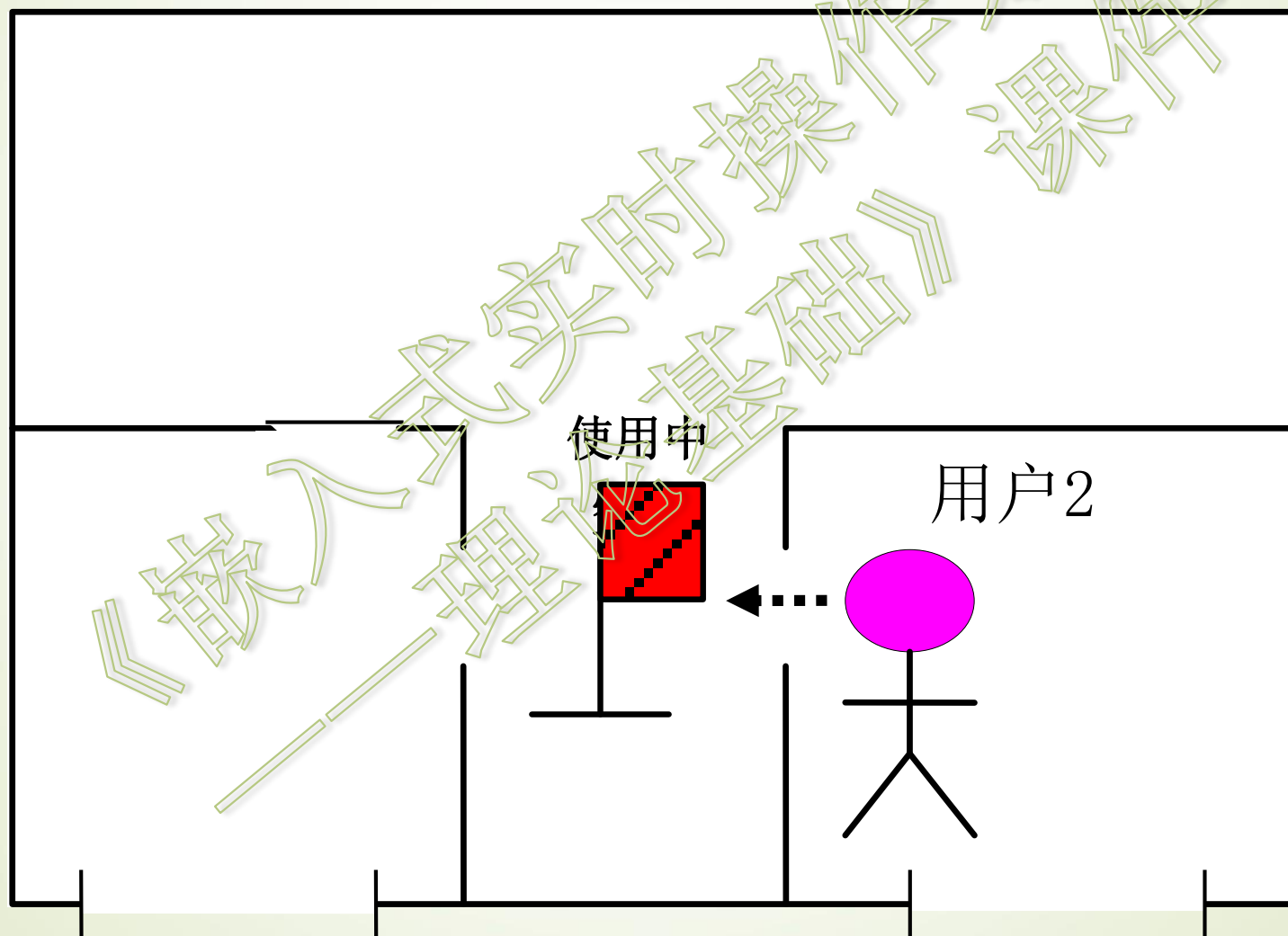
互斥的实现（单标志方法）



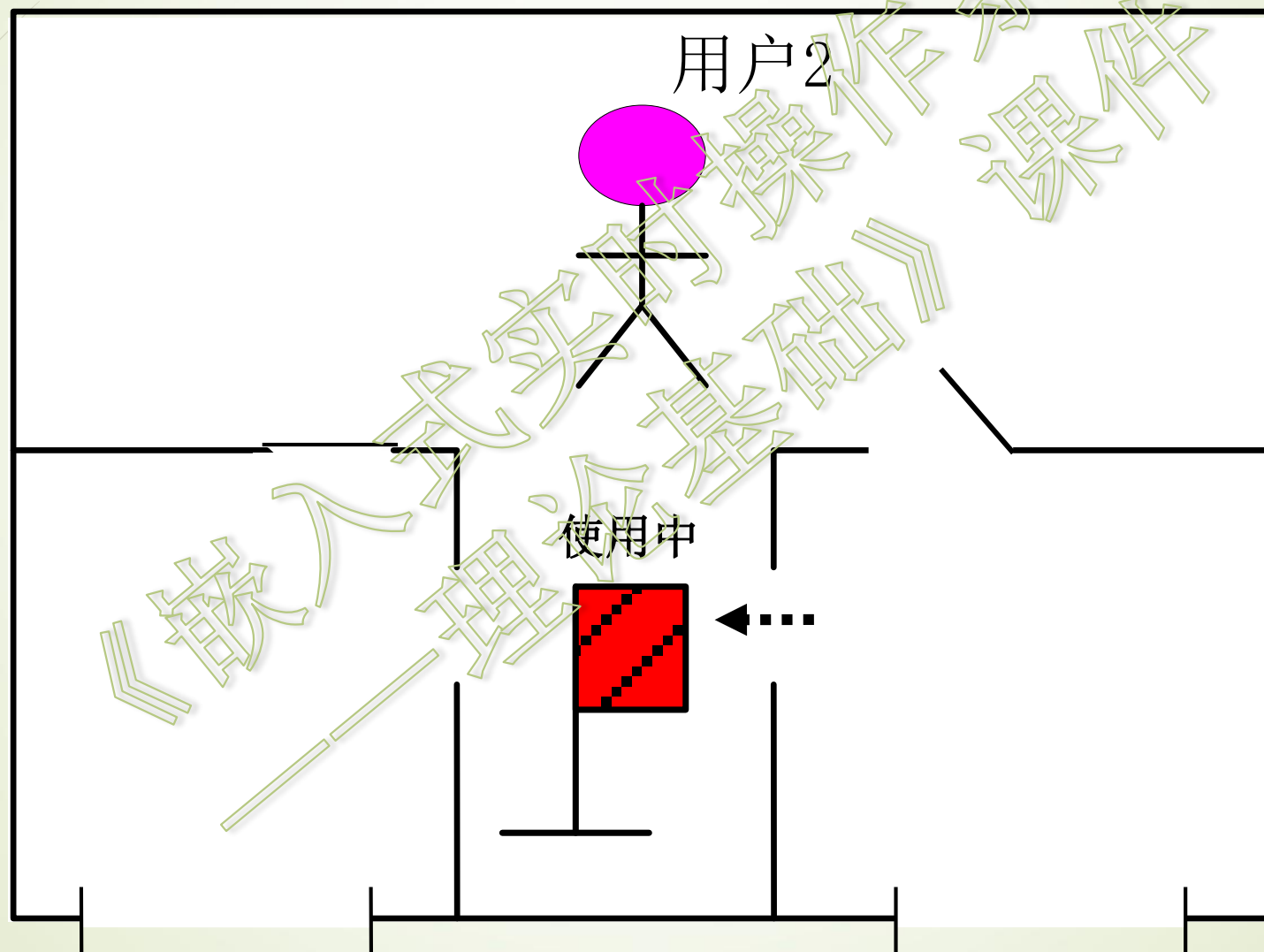
互斥的实现（单标志方法）



互斥的实现（单标志方法）



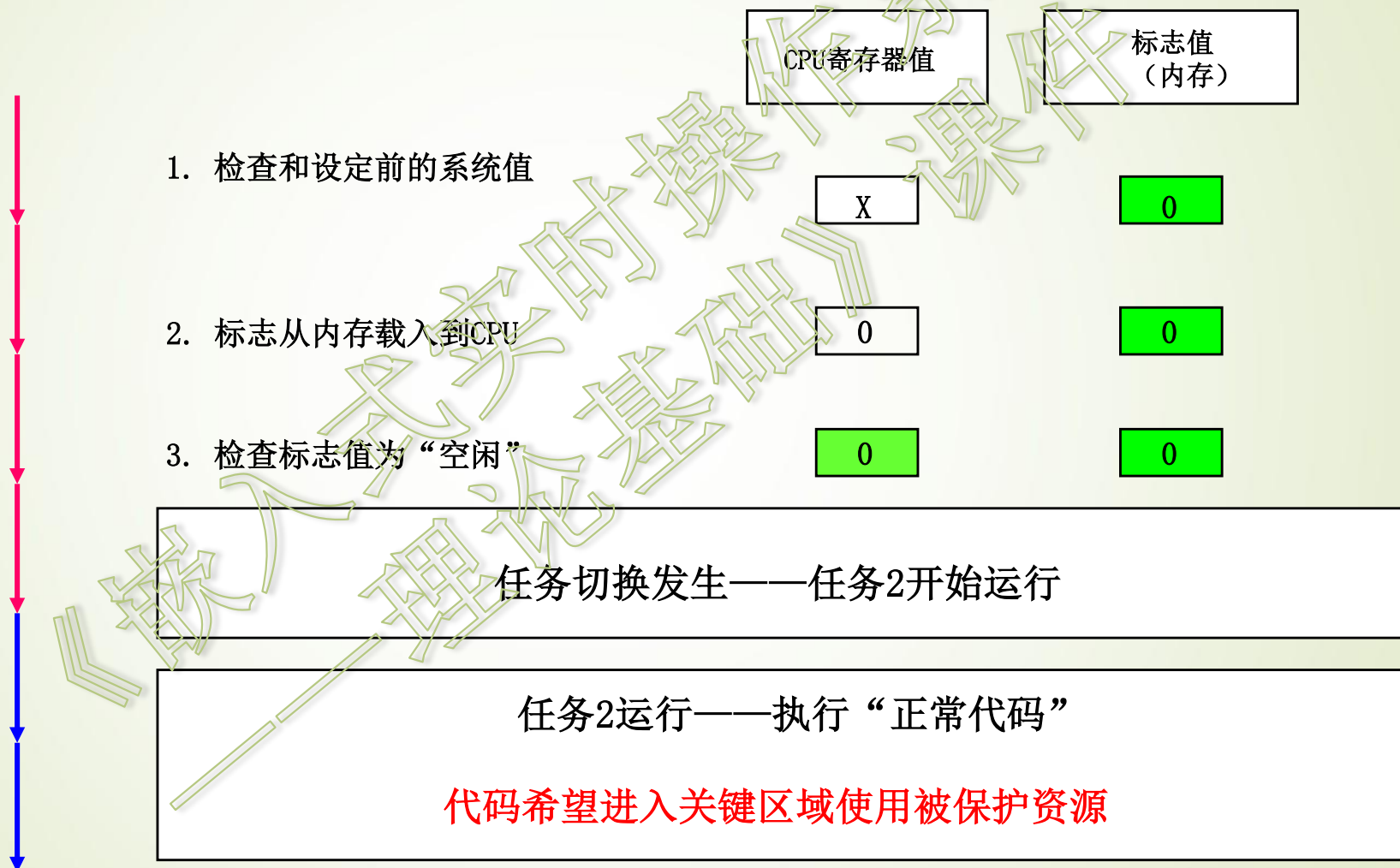
互斥的实现（单标志方法）



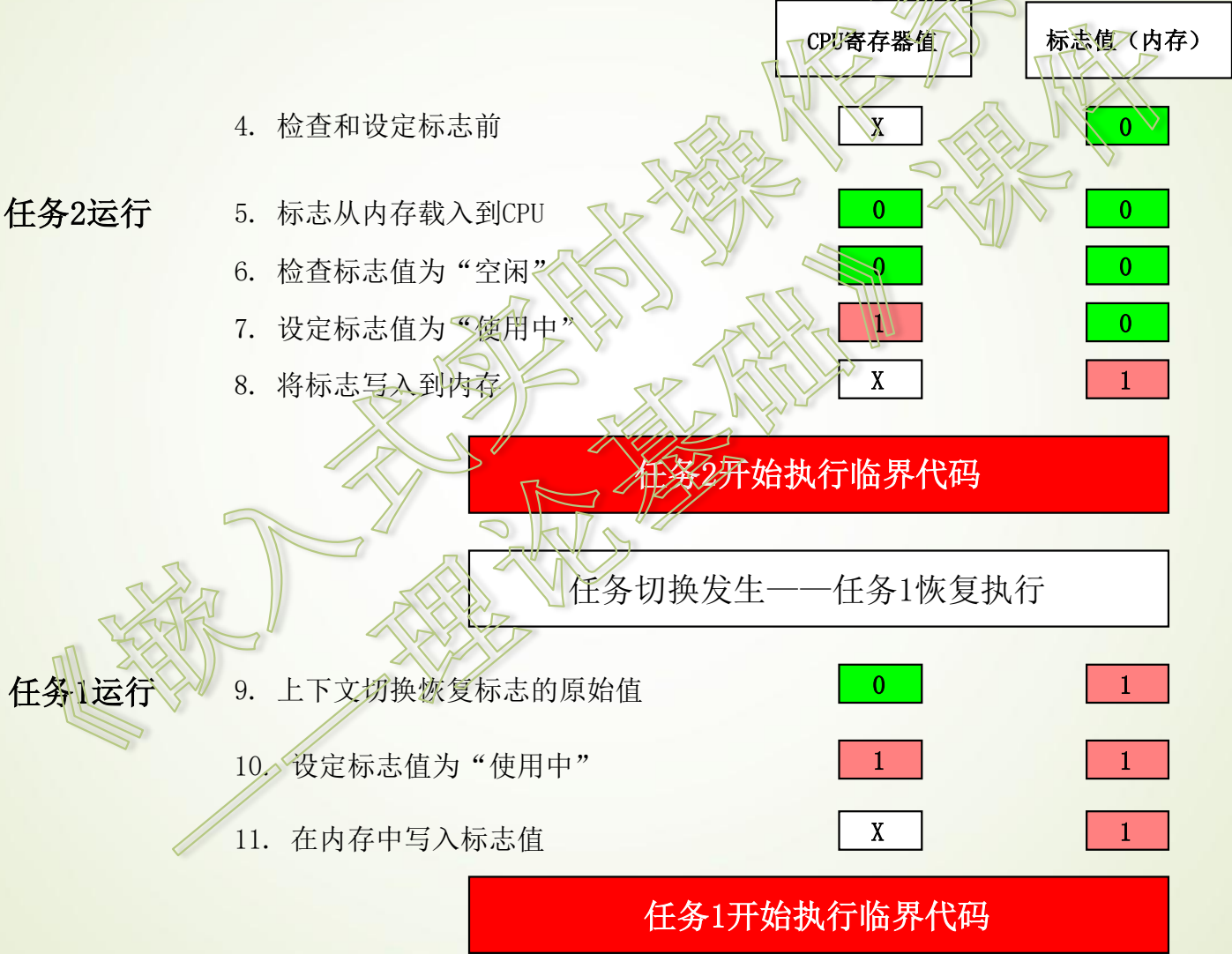
检查和设定标志——处理器级别细节

	CPU 寄存器值	标志值 (内存)
1. 检查和设定前	X	0
2. 从内存载入到CPU中	0	0
3. 标志检查——状态为“可用”	0	0
4. 标志状态设置为“使用中”	1	0
5. 将标志写入到内存	X	1

单一标志的问题



单一标志的问题



挂起等待和二值信号量

- 忙-等待方法很简单，但是会浪费处理器时间，并降低性能。
- 用挂起-等待替代，具体方式包括：信号量、互斥量和监控器。



信号量：

- 是一种流控制机制，用于控制任务执行。
- 类比：铁路信号，列车要么可以通过，要么必须停止。
- 程序中每个信号量都相当于一个信号。

二值信号量基础

- 二值信号量只有两种可能值：0或1。
- 0表示资源正在被使用，1表示资源空闲。
- 信号量的操作：

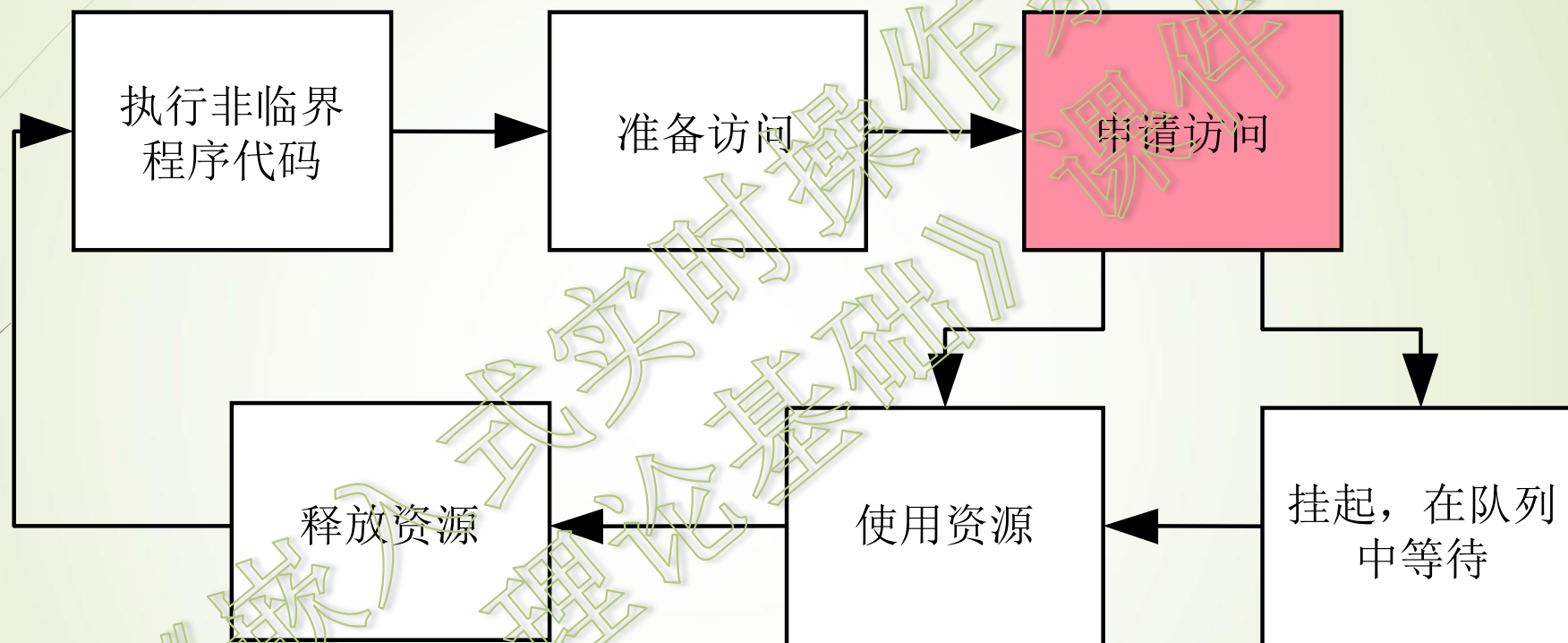
```
(*等待Coefficients信号量)  
IF (Coefficients=1) THEN  
    Coefficients:=0;  
ELSE 挂起任务;  
END
```

RTOS软件的一部分

```
(*释放Coefficients信号量)  
IF (任务等待中) THEN  
    唤醒任务;  
ELSE Coefficients:=1;  
END
```

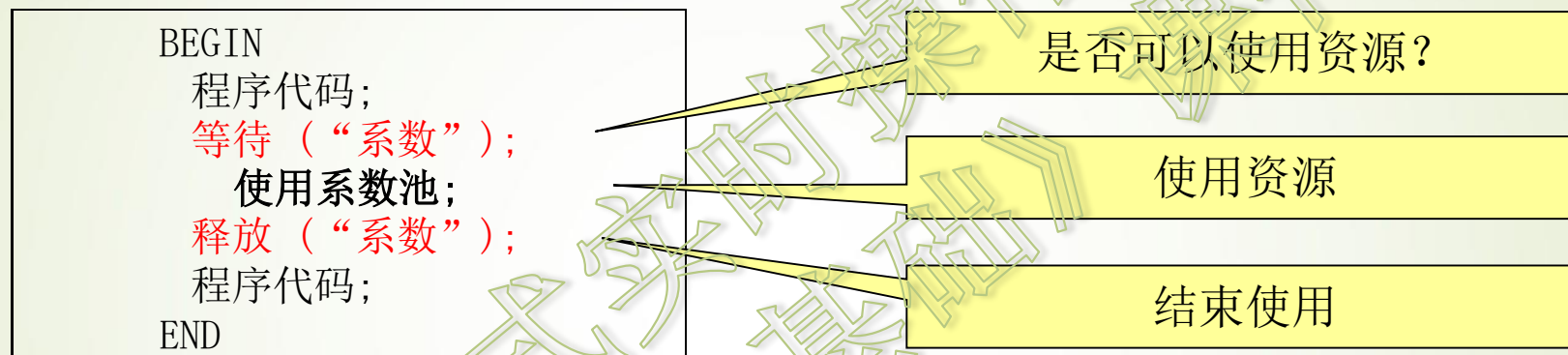
RTOS软件的一部分

使用信号量时的任务状态



二值信号量基础知识

- 在程序中按需使用二值信号量：



- 等待和操作为“原语”，不能再分割。
- 一旦等待/释放过程开始，机器指令序列不会被中断。

计数信号量基础知识

- 一个资源有数个单元。
- 每个单元都是相同的，例如CAN网络队列。
- 为了支持这一用例，对信号量做出如下修改：
 - 支持数值范围（比如0到5），初始值为最大值（5）。
 - 数值和资源的数目相关，0意味着所有设备都在使用中。
 - 用户访问资源前首先要检查资源是否空闲（非零数值）。如果可以访问，将信号量的值减一，然后开始使用资源。
 - 用户结束使用资源时，将信号量的值加一。

计数信号量基础知识

下面样例中的“队列”/Queue信号量:

- 控制对资源的访问
- 定义可以使用的资源数目
- 值不会为负

(*等待Queue信号量*)

IF (Queue)>0) THEN

Queue:= (Queue-1);

ELSE 挂起任务;

END;

(*释放Queue信号量*)

IF (任务等待中) THEN

唤醒任务;

ELSE Queue:=(Queue+1);

END;

信号量机制的缺陷——互斥量的概念

- 任何任务可以在任何时候进行信号量等待或者释放。
- 进行调用的任务并不“拥有”信号量。
- 解决方法——互斥量。
- 互斥量和二值信号量类似。
- 可以“锁定”或者“解锁”互斥量。
- 只有锁定互斥量的任务才可以解锁。
- 任务在互斥量锁定期间拥有互斥量的所有权。

使用互斥量的方法

使用pthread互斥量的例子

```
/* 声明 */  
pthread_mutex_t  mutex_ADC; /* 互斥量变量的声明 */  
  
/*初始化互斥量 */  
pthread_mutex_init (&mutex_ADC, NULL);
```

```
/* 锁定互斥量前的准备 */  
pthread_mutex_lock (&mutex_ADC);  
/* 临界代码 */  
GetAnalogueInput (RotorSpeed);  
pthread_mutex_unlock (&mutex_ADC);  
/* 互斥量已解锁 */
```

信号量机制的缺陷——简单监视器的概念

共享资源的概念并不可靠

- 资源在一些情形下受保护，在其他情形下只是普通的变量。
- 一个任务在临界区域时，另一任务可能直接访问资源。

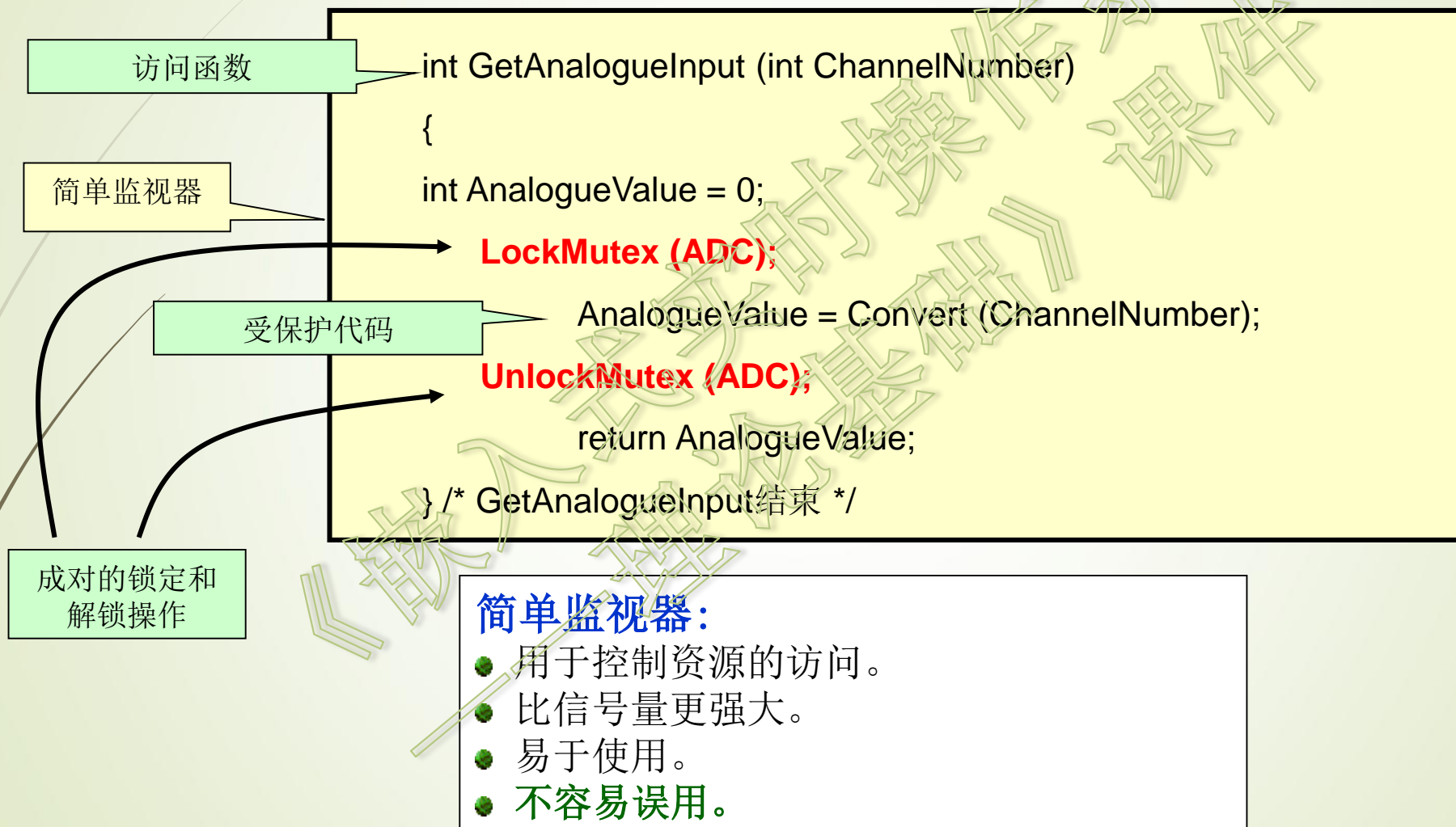
资源和保护资源的信号量互不相联

- 资源不能和信号量自动联系起来。

解决方法：

- 将资源和信号量封装在一起。
- 阻止对受保护代码的一般访问，即将它们变为封装单元中的私有代码。
- 这些就是简单监视器的基础。

简单监视器的要素



信号量机制的缺陷——优先级反转

和信号量相关的调度机制有其限制：

- 当一个任务调用锁定的互斥量时会被挂起。
- 挂起的任务以先入先出方式入队。
- 使用基于优先级的调度时这会导致问题：高优先级的任务也许会排在低优先级任务的后面。
- 这样就产生了优先级反转。
- 下一章会讨论解决方案。

本章结束

版权说明和联系方式

- 课件由《嵌入式实时操作系统——理论基础》一书翻译团队成员何灵渊、何小庆、张爱华和付元斌编写，图书原作者提供原始素材，清华大学出版社提供部分图片。课件可用于非商业场合，比如教学和研究课题，商业使用需联系作者。
- 如果你是高校老师，希望以本书内容开设课程，需要全套的课件（PPT格式）可以联系：
 - ① 本书译者：何小庆老师 xiaoqinghe@live.com 或者添加何老师微信 allanhexq（注明你的姓名+学校+专业）
 - ② 本书责编：刘星 liux@tup.tsinghua.edu.cn 电话：83470219 QQ：1468682976

注意： 请提供教材订购证明，获取全书的PPT课件。