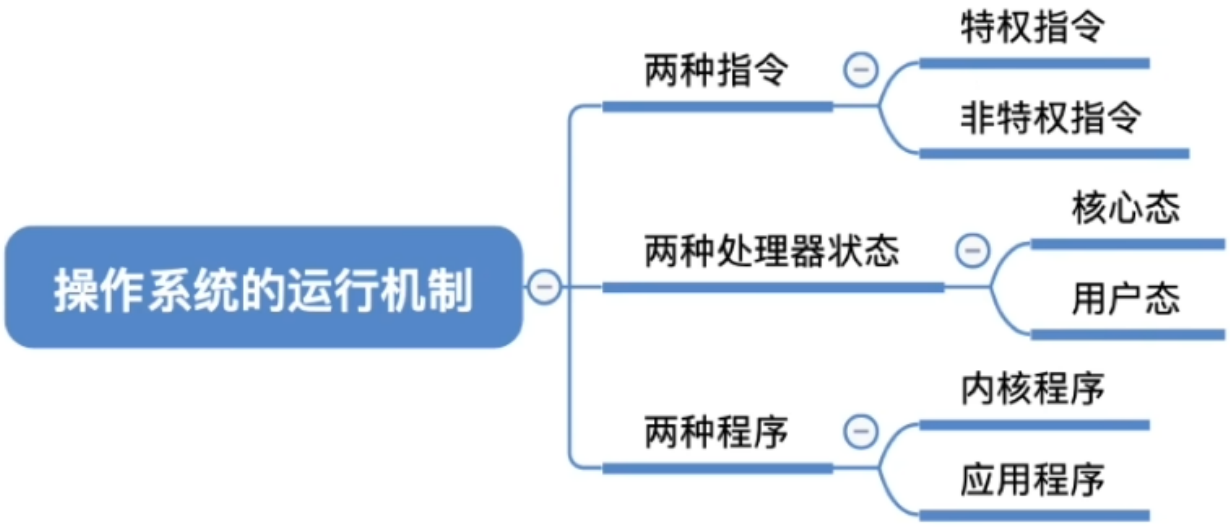
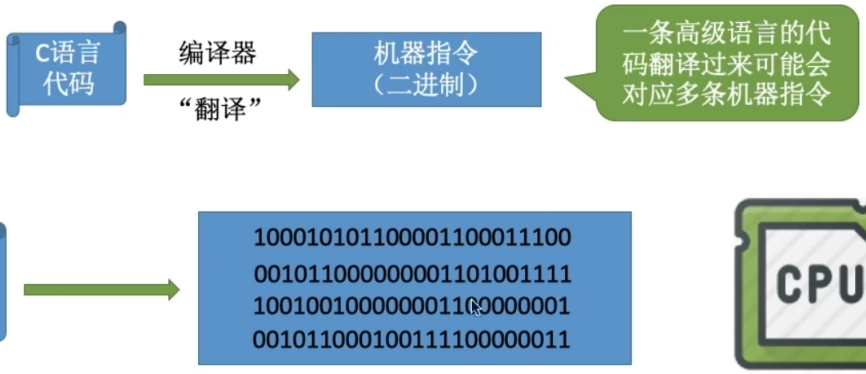
处理器运行模式



程序是如何运行的



“指令”就是处理器（CPU）能识别、执行的最基本命令

注：很多人习惯把 Linux、Windows、MacOS 的“小黑框”中使用的命令也称为“指令”，其实这是“交互式命令接口”，注意与本节的“指令”区别开。本节中的“指令”指二进制机器指令

内核程序v.s.应用程序

普通程序员写的程序就是”应用程序“

微软、苹果有一帮人负责实现操作系统，它们写的是”内核程序“

由很多内核程序组成了”操作系统内核“，或简称”内核(Kernel)“

内核是操作系统最重要最核心的部分，也是最接近硬件的部分

甚至可以说，一个操作系统只要有内核就够了（eg. Docker->仅需Linux内核）

操作系统的功能未必都在内核中，如图形化用户界面GUI

特权指令v.s.非特权指令

操作系统内核作为"管理者"，有时会让CPU执行一些"特权指令"，如：内存清零指令。这些指令影响重大，只允许"管理者"——即操作系统内核来使用

应用程序只能使用"非特权指令"，如：加法指令、减法指令等

在CPU设计和生产的时候就划分了特权指令和非特权指令，因此CPU执行一条指令前就能判断出其类型

内核态v.s.用户态

CPU能判断出指令类型，但是它怎么区分此时正在运行的是内核程序or应用程序?

CPU有两种状态，内核态和用户态

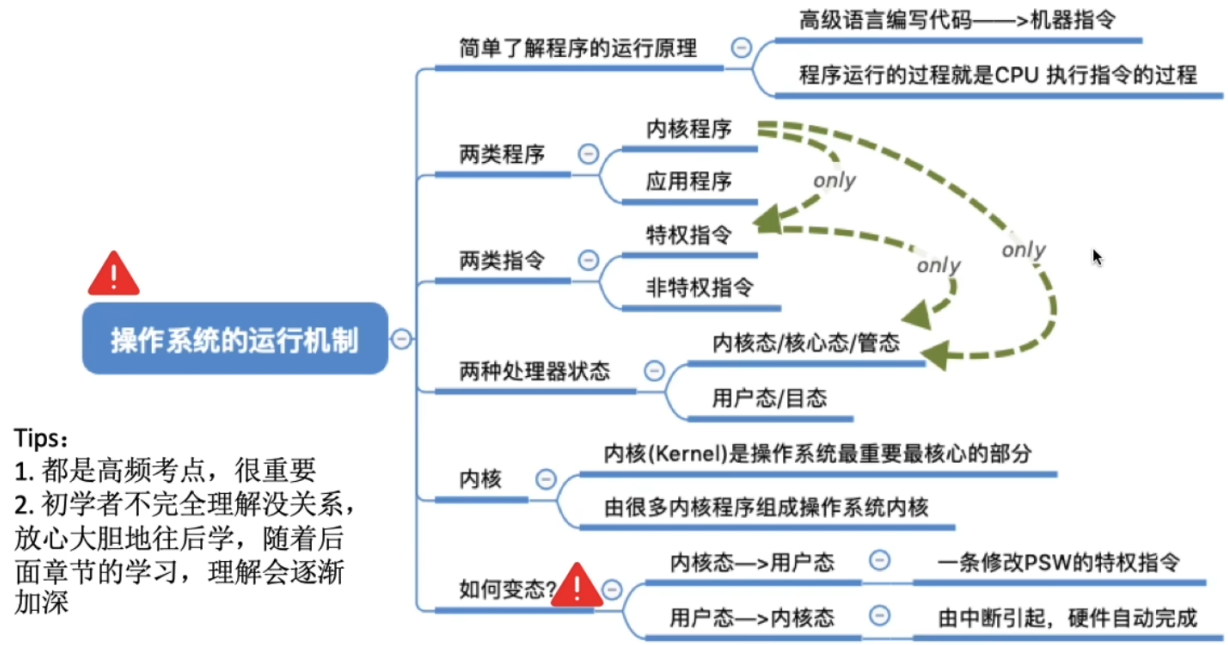
处于内核态时，说明此时正在运行的是内核程序，此时可以执行特权指令

处于用户态时，说明此时正在运行的是应用程序，此时只能执行非特权指令

拓展：CPU中有一个寄存器叫程序状态字寄存器（PSW），其中有个二进制位，1表示内核态，0表示用户态

别名：内核态=核心态=管态；用户态=目态

内核态、用户态的切换



内核态->用户态：执行一条特权指令——修改PSW的标志位为"用户态"，这个动作意味着操作系统将主动让出CPU使用权

用户态->内核态：由中断引发，硬件自动完成变态过程，触发中断信号意味着操作系统将强行夺回CPU的使用权

除了非法使用特权指令之外，还有很多事件会触发中断信号。一个共性是，但凡需要操作系统接入的地方，都会触发中断信号

一个故事：

① 刚开机时，CPU 为“内核态”，操作系统内核程序先上CPU运行

② 开机完成后，用户可以启动某个应用程序

③ 操作系统内核程序在合适的时候主动让出 CPU，让该应用程序上CPU运行

④ 应用程序运行在“用户态”

⑤ 此时，一位猥琐黑客在应用程序中植入了一条特权指令，企图破坏系统...

⑥ CPU发现接下来要执行的这条指令是特权指令，但是自己又处于“用户态”

⑦ 这个非法事件会引发一个中断信号

操作系统内核在让出CPU之前，会用一条特权指令把 PSW 的标志位设置为“用户态”

CPU检测到中断信号后，会立即变为“核心态”，并停止运行当前的应用程序，转而运行处理中断信号的内核程序

⑧ “中断”使操作系统再次夺回CPU的控制权

⑨ 操作系统会对引发中断的事件进行处理，处理完了再把CPU使用权交给别的应用程序