回顾 — 操作系统的主要工作

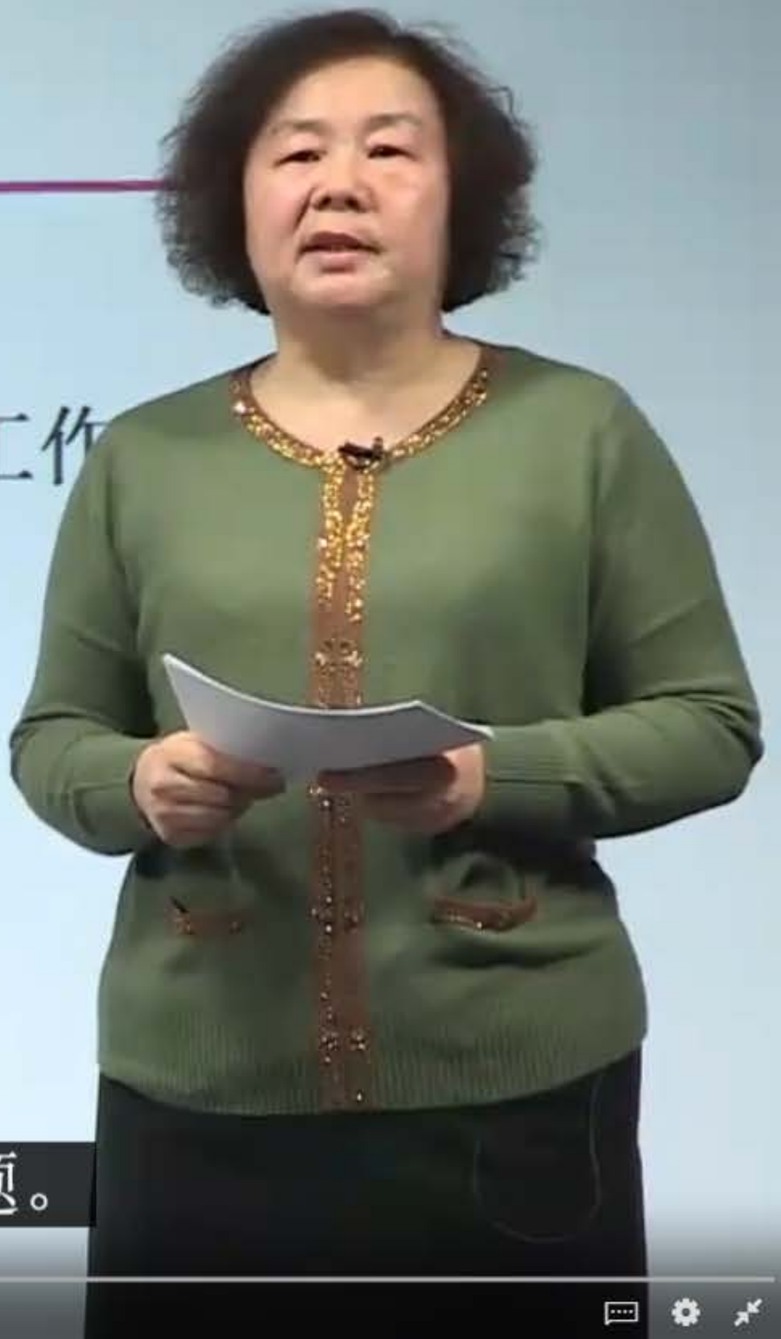
- 程序的执行

  启动程序、执行程序以及程序结束的工作

- 完成与体系结构相关的工作
- 完成应用程序所需的共性任务

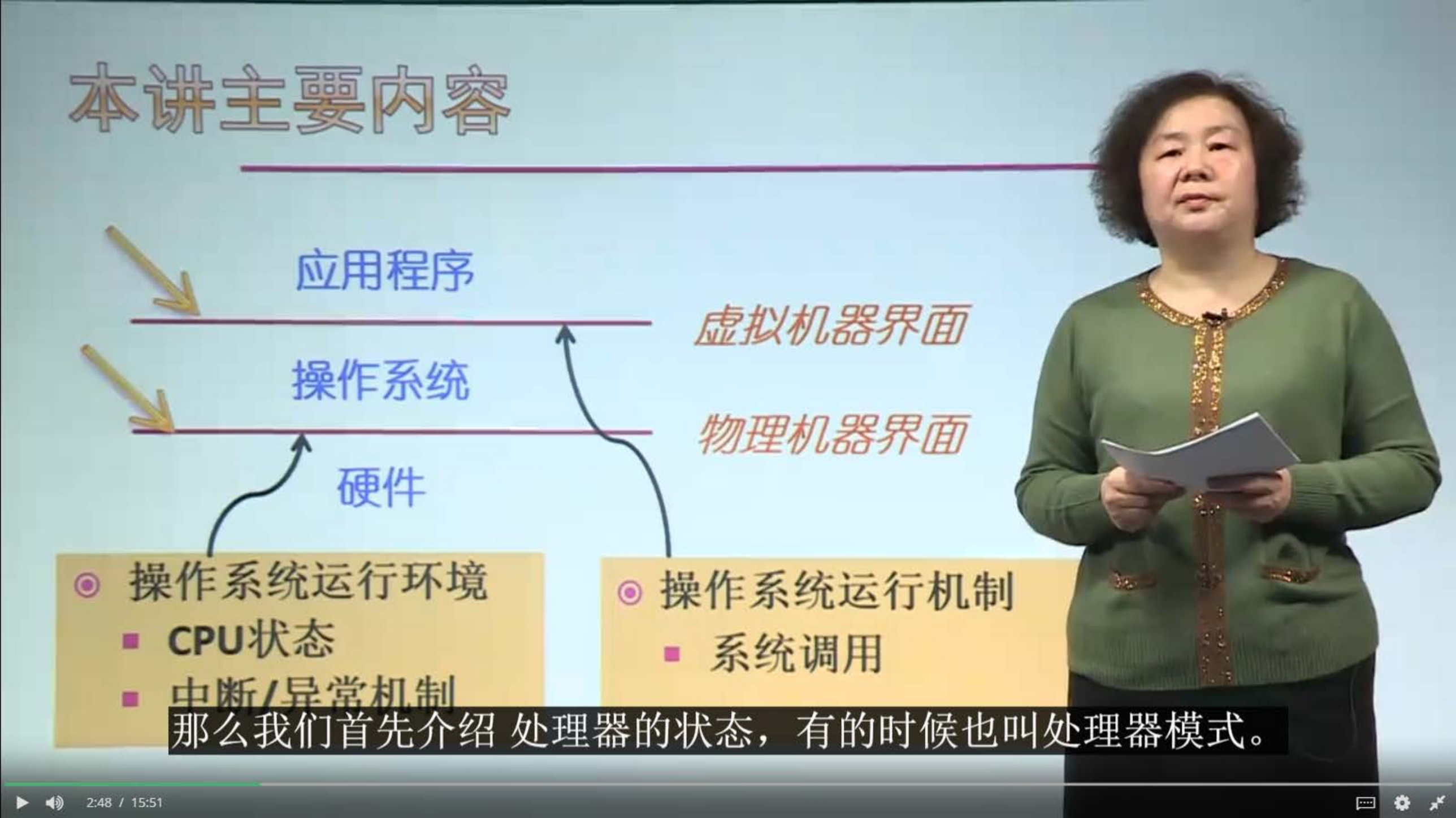
  提供各种基本服务

- 性能、安全、健壮等问题



还包括了性能、 安全、 健壮等问题。

本讲主要内容



- 操作系统运行环境
  - CPU状态
  - 中断/异常机制
- 操作系统运行机制
  - 系统调用

那么我们首先介绍 处理器的状态，有的时候也叫处理器模式。

处理器状态(模式)

那么我们首先介绍 处理器的状态，有的时候也叫处理器模式。

中央处理器（CPU）

- 处理器由运算器、控制器、一系列的寄存器以及高速缓存构成

- 两类寄存器：
  - 用户可见寄存器：高级语言编译器通过优化算法分配并使用之，以减少程序访问内存次数
  - 控制和状态寄存器：用于控制处理器的操作，通常由操作系统代码使用

控制处理器的操作，那么通常呢是只能由操作系统代码来使用。

控制和状态寄存器

- 用于控制处理器的操作
- 在某种特权级别下可以访问、修改

- 常见的控制和状态寄存器
  - 程序计数器（PC：Program Counter），记录将要取出的指令的地址
  - 指令寄存器（IR：Instruction Register），记录最近取出的指令
  - 程序状态字（PSW：Program Status Word），记录处理器的运行状态如条件码、模式、控制位等信息

操作系统的需求——保护

- 从操作系统的特征考虑

  并发、共享

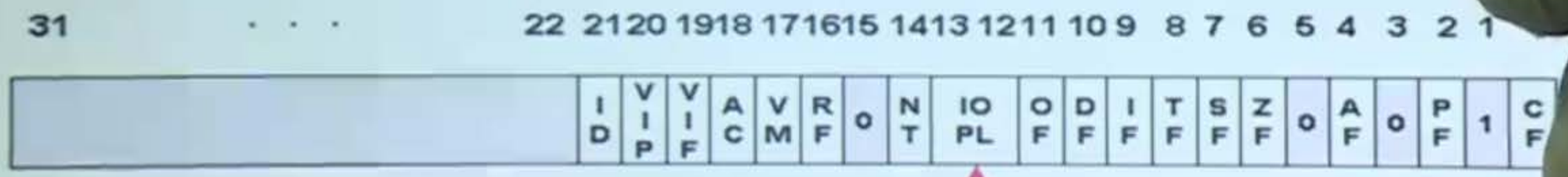
- 提出要求 → 实现保护与控制

需要硬件提供基本运行机制：

- 处理器具有特权级别，能在不同的特权级运行的不同指令集合
- 硬件机制可将OS与用户程序隔离

处理器的状态(模式MODE)

- 现代处理器通常将CPU状态设计划分为两种、三种或四种
- 在程序状态字寄存器PSW中专门设置一位，根据运行程序对资源和指令的使用权限而设置不同的CPU状态



例:X86架构中的EFLAGS寄存器

描述符中也设置了权限级别

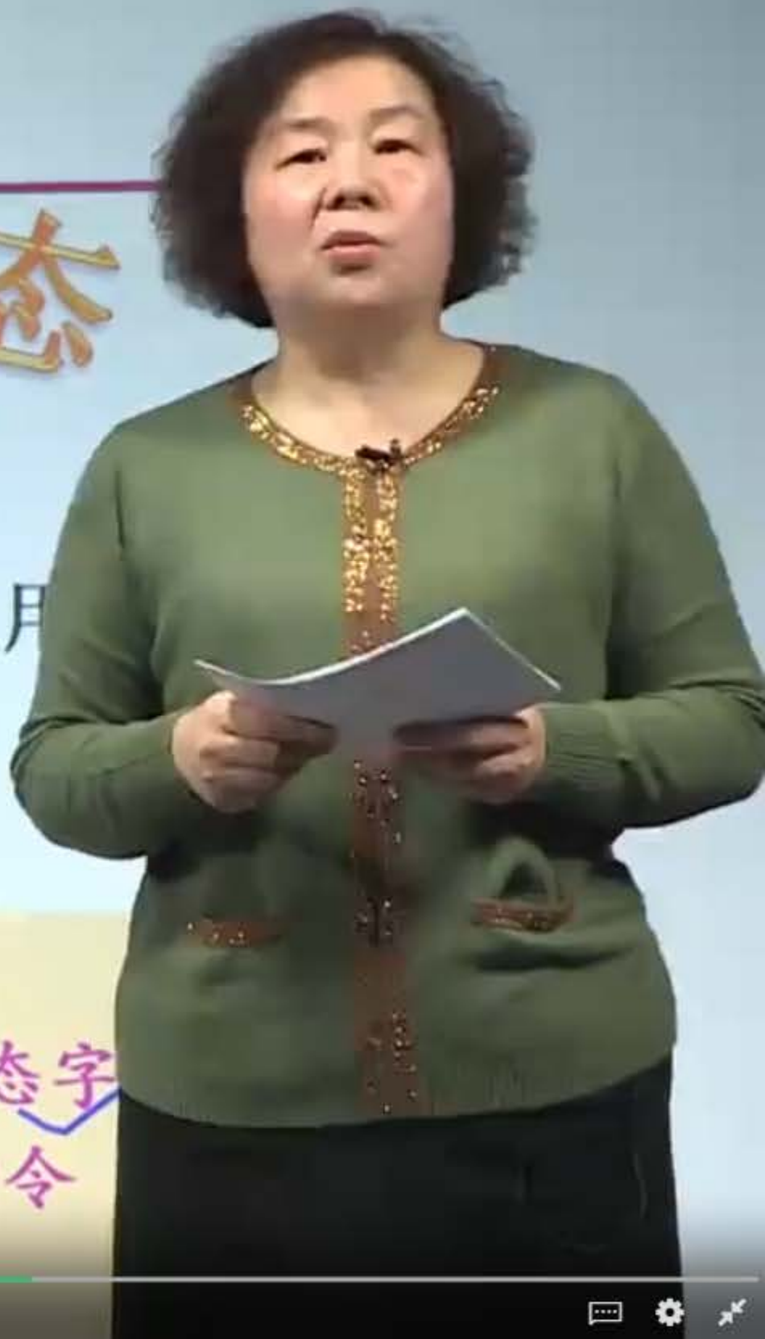
特权指令和非特权指令

2个状态

- 操作系统需要两种CPU状态
  - 内核态(Kernel Mode)：运行操作系统程序
  - 用户态(User Mode)：运行用户程序

- 特权(privilege)指令：只能由操作系统使用、用户程序不能使用的指令
- 非特权指令：用户程序可以使用的指令

Q
- 下列哪些是特权指令？哪些是非特权指令？
  - 启动I/O　控制转移　内存清零　修改程序状态字
    设置时钟　算术运算　允许/禁止中断　访管指令
    取数指令　停机



实例：X86系列处理器

- X86支持4个处理器特权级别

  特权环：R0、R1、R2和R3

  - 从R0到R3，特权能力由高到低
  - R0相当于内核态；R3相当于用户态；R1和R2则介于两者之间
  - 不同级别能够运行的指令集合不同

- 目前大多数基于x86处理器的操作系统只用了R0和R3两个特权级别

13:28 / 15:51

CPU状态之间的转换

- 用户态 → 内核态

唯一途径 → 中断/异常/陷入机制

- 内核态 → 用户态

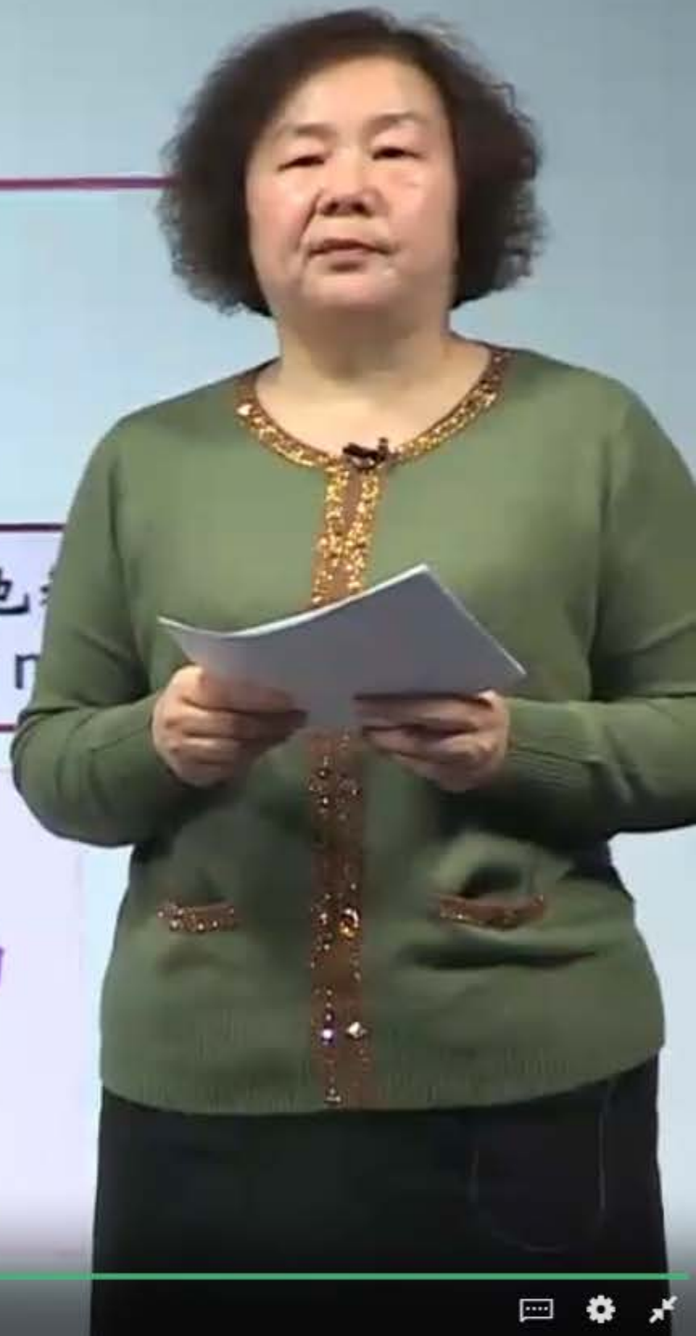
设置程序状态字PSW

因为内核态也
为supervisor m

一条特殊的指令：陷入指令（又称访管指令）

提供给用户程序的接口，用于调用操作系统的功能（服务）

例如：int，trap，syscall，sysenter/sysexit



0:00 大家好，今天我给大家带来的是操作系统原理的第二讲。 操作系统运行环境与运行机制。 那这一部分的内容是非常重要的。 为什么呢？我们先来回顾一下操作系统的主要工作。 操作系统支持程序的执行， 包括了启动程序、 执行程序以及程序结束后的一些工作。 操作系统还完成与体系结构相关的工作。 完成与应用程序所需的共性任务。 还包括了性能、 安全、 健壮等问题。 那么在这里头操作系统完成与体系结构相关的工作， 这是一个非常重要的点。 这是操作系统与其他软件所不同的地方。 设计操作系统的时候必须和硬件打交道， 必须了解硬件体系结构。 除此之外 操作系统还完成应用程序所需的共性任务。 在应用程序运行过程中需要做很多的工作。 比如说读盘、 申请一块内存存放数据、 用打印机来打印文件。 那么这些工作呢， 是操作系统完成的，并且呢向用户提供这些基本服务。 所以我们通过这张图的回顾，来明确一下操作系统的地位。 操作系统 是在硬件基础上的第一层扩展。 那么它底下是硬件， 那么这个硬件都做了什么工作呢？ 我们怎么样去了解底下的硬件呢？那么这就是我们 本讲的一个内容之一：操作系统运行环境。 那么和硬件相关的很多的 工作呢实际上是和操作系统的各个功能 相结合的，那么我们这里头呢只介绍最基本的 CPU 状态： 中断与异常机制。 其他的像虚拟页式存储管理的机制， 设备等等的，那么我们放到后面跟操作系统功能结合起来讲。 操作系统完成应用程序当中的一些共性的工作， 向应用程序提供一些基本服务，那么这是我们本讲的另外一方面的内容 操作系统运行机制，我们重点介绍系统调用机制。 那么我们首先介绍 处理器的状态，有的时候也叫处理器模式。 CPU 是由运算器 控制器、 一系列的寄存器以及高速缓存 构成。 那我们来看一看有两类寄存器。 一类是用户可见寄存器，那么高级语言 编译器通过优化算法分配这些寄存器，并且使用这些寄存器 主要的目的是为了减少访问内存的次数， 来提高程序的运行效率。 作为操作系统设计者，我们更加关注的是 控制和状态寄存器。 那么这些寄存器呢是用于 控制处理器的操作，那么通常呢是只能由操作系统代码来使用。 我们来看一下控制和状态寄存器。 它用于控制处理器的操作， 在某些特权级别下可以访问来修改。 典型的控制和状态寄存器呢包括了程序计数器 PC，那么这里头记录了要取出的指令的地址。 指令寄存器记录了最近取出的指

令。 程序状态字寄存器 PSW，它记录了 CPU 的运行状态， 一些条件码、 模式、 控制位等信息。 下面我们来探讨一下操作系统 对硬件的需求，其中一个非常重要的需求就是保护。 因为操作系统运行在一个多进程的这样一个环境下，支持这些进程的运行。 因此我们从操作系统的特征来考虑: 并发、 共享。 操作系统为多个程序的执行提供了这样一个 并发的环境，那么多个进程之间呢又共享操作系统所管理的各种资源。 那么这样一个并发、 共享的计算环境就要求保护 保护是保护用户程序与用户程序之间 互不干扰，保护是保护用户程序不对操作系统干扰。 那么这就从操作系统的角度给硬件提出了一个需求。 要实现保护、 实现保护控制。 通常，我们对硬件希望提供这样的一个 基本运行机制。 也就是 CPU 呢它具有一个特权级别， 在不同的特权级下可以运行不同的指令集合。 这样把指令分成不同的集合，那么供 操作系统和用户程序分别使用。 那么通过保护呢又使得 操作系统与用户能够相隔离，能够相隔离。 比如说当要访问操作系统空间的时候，那么用户程序是不能够访问的。 那么操作系统可以访问用户程序空间， 因此达到了，通过一个保护机制能够达到操作系统与用户的隔离。 有了这样一个需求之后，现代处理器通常把 CPU 的状态 设计为两种、 三种、 或者是四种。 那么在 CPU 上 时而运行操作系统、 时而运行用户程序， 那么 CPU 如何知道是运行哪一种状态呢？ 这样就有赖于一些寄存器的某些位的设置。 通常呢是在程序状态字寄存器 PSW 当中呢来专门设置一位，有的时候是两位 根据运行程序对资源和指令的权限不同， 来设置不同的 CPU 状态。 好，那么 不同的程序对资源和指令的使用要求是不同的， 我们来看一下 X86 处理器当中的 典型的一个标志寄存器 EFLAGS，EFLAGS 寄存器。 那么这里头呢就有一位 IO PL 是 IO 的权限位，IO 的权限级别。 那么这个级别呢我们可以看到用两位来表示 两位可以表示四个状态，四个状态。 除了这个寄存器以外，还有一些 描述符也规定了，设置了权限级别。 这是硬件提供的各种不同的 CPU 状态 那么操作系统呢其实需要两种 CPU 状态 一种是用于内核态：运行操作系统的代码 另一种呢是用户态：运行用户程序。 所以呢，操作系统只需要两个状态 那么两个状态呢可以指向不同的指令集合 因此我们把指令的集合划分成 两类，一类呢我们称之为特权指令， 所谓特权指令呢是只能够由操作系统来使用 那么用户呢是不能使用的指令 那么另外一类呢就叫非特权指令 是用户可以使用的指令，其实整个指令系统呢，那么操作系统是都可以使用的 它既可以使用特权指令，也可以使用非特权指令 而用户呢

只能使用这个指令 系统当中的一个子集，那么这个子集就是非特权指令 那么下面呢提个问题，下列哪些指令 是特权指令，哪些指令呢是非特权指令？比如说启动 I/O 那么各种各样的输入/输出设备由操作系统管理 用户程序如果需要打印，如果要扫描，那么它自己不能完成这些工作 它必须把这个请求提交给操作系统，由操作系统代它来完成这些任务 所以启动 I/O 这个指令是特权指令 那么程序执行过程中有很多的控制转移，那么这就是非特权指令 那如果对内存 设置清零，那么这个呢只能够由操作系统来做 那么用户呢只能在操作系统的授权下来完成这项事情，所以 内存清零是由，是特权指令 修改程序状态字，那显然是一个特权指令 只能操作系统来设置 CPU 的状态，做一些其他方面的设置 所以这是一个典型的特权指令。 设置时钟 是特权指令，算术运算当然是非特权指令 如果我不想接收中断 或者是我想接收中断，做这样的设置，那么也是特权指令 停机是特权指令 取数指令呢是非特权指令。 那么这里有一个非常特殊的指令 我们叫做访管指令。 那么这条指令呢 我们后面会专门来介绍。 这条指令呢是非特权指令 但是它的作用呢，是使得用户程序从用户态陷入操作系统内核态 那我们来举一个例子 X86 系列处理器，它提供了四个 特权级别。 我们把它称之为特权环 R0， R1， R2 和 R3 那么从 R0 到 R3 它的特权的能力是由高到底的。 那么硬件设计者 设计了这样不同的特权级别 主要的目的呢，是希望在不同的级别呢能够运行不同的 程序。 比如说 R0 是希望能运行操作系统的一些关键代码 所以 R0 相当于内核态。 R1 呢 是运行设备驱动程序和一些 I/O 处理的历程 R2 呢，是运行一些受保护 共享的代码，比如说一些语言编译环境 而 R0 呢，R3 呢是给 用户程序使用的。 那么 R3 呢就相当于用户态 而 R1 和 R2 呢实际上是介乎于两者之间 那么不同的这种特权级别 其实就是运行指令的集合是不一样的 我们来看一下 R0 实际上是运行了所有的指令 而 R3 是运行的一个最小的子集 那么当然了，我们刚才说了，操作系统需要两个状态 所以呢通常情况下，大部分的 操作系统我们所熟知的，Linux 啊，Windows 啊，Unix 啊 都是只选择了 R0 和 R3 这两个特权级别 来使用。 那么有了不同的 特权级别，那么就需要 让用户程序和操作系统之间能有转换 那么实际上呢用户程序在执行的过程中 如果需要操作系统的服务，它就要从用户态能够陷入到，进入内核态 而从用户态进入内核态的一个唯一的途径 唯一的通路实际上就是中断/异常/陷入机制 这也是我们下一个非常重要的主题 那么从内核态返回到用户态呢？ 则比较简单啊，只是通过设置程序状态字寄存器就可

以了 那么我们刚才说的一条特殊的指令 陷入指令，或者呢叫访管指令 那么这条指令呢，它的作用呢是提供给用户程序的一个接口 用这个接口使用户程序可以向操作系统提出各种服务请求 那么为什么叫访管指令呢？ 因为有的时候内核态也被称为一个 supervisor mode 管理态。 所以在这种情况下呢，那么从用户态进入了管理态，相当于访问管理态 所以叫访管指令，有这样一个说法。 所以大家记忆的时候呢说访管指令是因为有这样一个 对内核态的另外一种称呼，所以有这一个访管指令 那么这条特殊的指令在不同的计算机系统当中呢 实际上是用不同的指令的。比如说我们看到了有 int 指令 有 trap 指令，syscall 还有 我们就叫 sysenter/sysexit。 那么这些指令都是 不同的计算机体系结构提供的这种 特殊的指令，用于陷入，用于访管 所以这是关于 CPU 状态之间的转换需要的不同的条件