X86处理器对中断/异常的支持

中断/异常机制实例

下面我们介绍中断异常机制的实例。

基本概念——X86处理器

- 中断
  - 由硬件信号引发的，分为可屏蔽和不可屏蔽中断
- 异常
  - 由指令执行引发的，比如除零异常
  - 80x86处理器发布了大约20种不同的异常
  - 对于某些异常，CPU会在执行异常处理程序之前产生硬件出错码，并压入内核态堆栈
- 系统调用
  - 异常的一种，用户态到内核态的唯一入口

1:08 / 9:48

X86处理器对中断的支持(1/6)

- 中断控制器（PIC或APIC）
  - 负责将硬件的中断信号转换为中断向量，并引发CPU中断

- 实模式：中断向量表 (Interrupt Vector)
  - 存放中断服务程序的入口地址
    - 入口地址＝段地址左移4位＋偏移地址
    - 不支持CPU运行状态切换
    - 中断处理与一般的过程调用相似
- 保护模式：中断描述符表 (Interrupt Descriptor Table)

  采用门(gate) 描述符数据结构表示中断向量

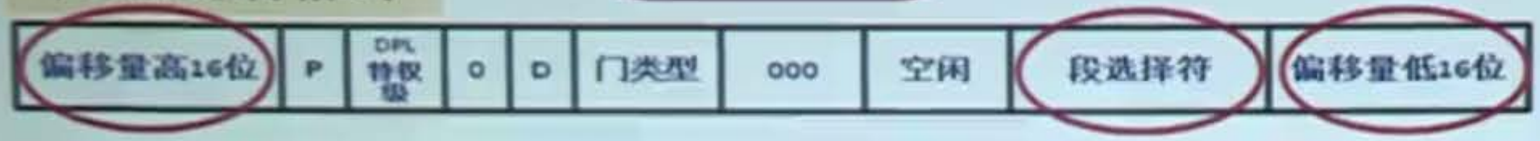
X86处理器对中断的支持(2/6)

- 中断向量表/中断描述符表
  - 四种类型门描述符
    - 任务门(Task Gate)
    - 中断门(Interrupt Gate)
      - 给出段选择符 (Segment Selector)、中断/异常序的段内偏移量 (Offset)
      - 通过中断门后系统会自动禁止中断
    - 陷阱门(Trap Gate)
      - 与中断门类似，但通过陷阱门后系统不会自动禁止中断
    - 调用门(Call Gate)

X86处理器对[illegible]的支持(3/6)

中断服务程序
入口地址 =
段基址+偏移

中断描述符格式



段描述符格式

段选择符格式

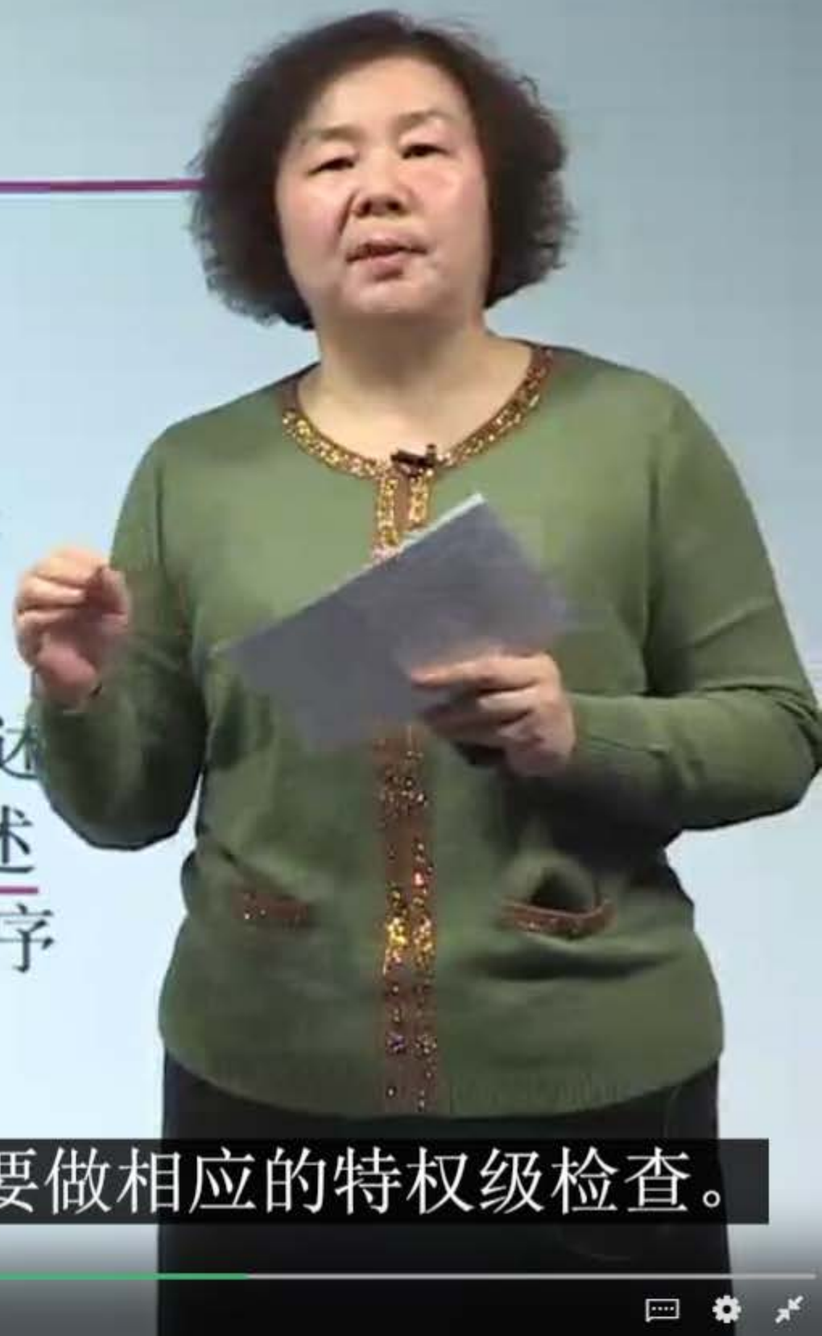
是这样一个大致的过程。

6:21 / 9:48

X86处理器对中断的支持(4/6)

中断/异常的硬件处理过程：

- 确定与中断或异常关联的向量i
- 通过IDTR寄存器找到IDT表，获得中断描述符（表中的第i项）
- 从GDTR寄存器获得GDT的地址；结合中断描述符中的段选择符，在GDT表获取对应的段描述符；从该段描述符中得到中断或异常处理程序所在的段基址
- 特权级检查



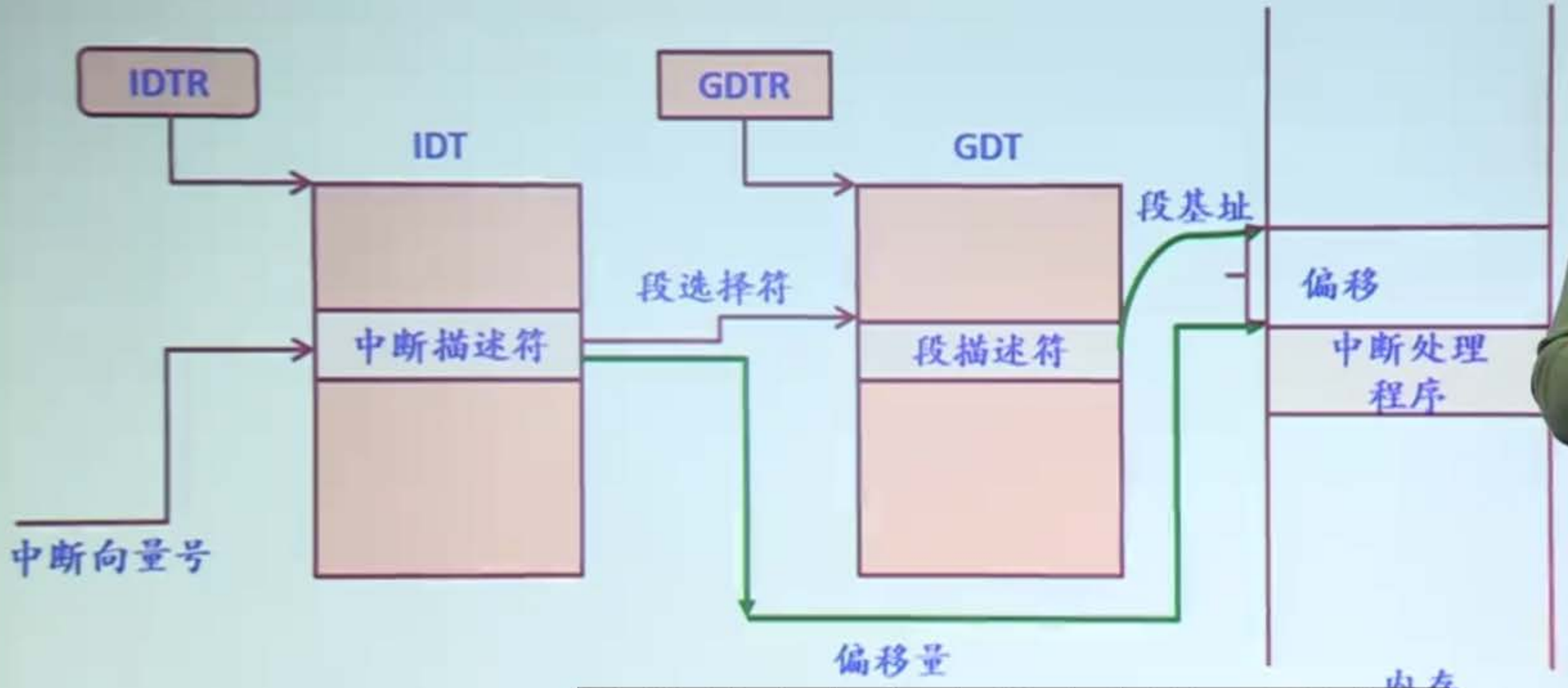
刚才我们看到了不同的描述符里头都有特权级，所以要做相应的特权级检查。

X86处理器对中断的支持(5/6)

- 检查是否发生了特权级的变化，如果是，则进行堆栈切换(必须使用与新的特权级相关的栈)
- 硬件压栈，保存上下文环境；如果异常产生了硬件出错码，也将它保存在栈中
- 如果是中断，清IF位
- 通过中断描述符中的段内偏移量和段描述符中的基地址，找到中断/异常处理程序的入口地址，执行其第一条指令

它的第一条指令。

X86处理器对中断的支持(6/6)



这就是在内存中的中断处理程序。

0:00 下面我们介绍中断异常机制的实例。 那么主要以 X86 处理器对中断／异常的这个支持为例。 在 X86 处理器当中 中断是指的由硬件信号引发的， 那么通常分成了可屏蔽中断和不可屏蔽中断。 异常呢是由正在执行指令而引发的， 前面我们说的像除零啊、 算数溢出等等。 那么 X86 这样一个处理器呢，大约呢发布了大概 20 多种 不同的异常。 不同的处理器发布的异常个数是不一样的。 而且对于某些异常呢，CPU 会在处理异常处理 程序之前产生一个硬件出错码， 把这个硬件出错码压入到内核栈里头。 另外一种就是特殊的一种异常，叫系统调用。 这个我们前面反复强调，它是用户态到内核态的一个唯一入口。 在 X86 处理器当中呢我们有中断控制器。 中断控制器呢是负责将硬件的中断信号转换为中断向量，并引发 CPU 的中断。 那么这里头我们来介绍中断向量表。 在 X86 处理器当中呢我们有实模式和保护模式之分。 在实模式下呢，中断向量表就叫中断向量表，存放了中断服务程序的入口地址。 那么拿到这个地址之后呢，把这个段地址，这个 入口地址等于这个段地址，段寄存器的段地址， 左移 4 位，然后加上偏移地址就得到了这个中断处理程序的入口地址。 由于在实模式下呢没有 CPU 状态的这种转换， 所以中断的处理与一般过程的这个调用呢是很相似的。 但是我们现在更多地呢是在保护模式下工作。 因此呢在保护模式下呢，中断向量表呢就改了个名字，叫中断描述符表。 那么中断描述符表当中每一个中断向量呢 我们把它现在的名字呢就叫门描述符，用一个门描述符 来表示我们前面说的中断向量，这是一个数据结构。 好下面我们来看一下具体的。 那么中断向量表或者叫中断描述符表 这里头有几种类型的门描述符呢？一般情况下有四种。 任务门、 中断门、陷阱门还有调用门。 那么我们呢通常只用了两种，一个是 中断门，一个是陷阱门。 所谓中断门呢是在这个门描述符当中给出了段选择符，以及中断／异常程序的一个段内的偏移量。 中断门通过之后， 系统会自动禁止中断，就不再接收新的中断了。 而陷阱门呢其实它的整体的处理是和中断门相似的，只是通过了陷阱门之后 不会自动禁止中断，所以还可以接收新的中断。 我们来看看门描述符的 一些具体的数据结构和整个的中断响应过程。 这里头呢我们来首先看一下 这张表，中断描述符表。 有一个寄存器叫 IDTR 寄存器， 那么通过这个寄存器我可以得到中断描述符表的 地址，我可以找到中断描述符表。 中断描述符表的每一行是个门描述符，或者叫做中断描述符。 那么中断描述符的格式呢是这样一个格式： 它呢是 64 位，分成不同的内容。 我们现在看到这有一个门的类型，

是中断门还是陷阱门。 还有一个呢是一个特权级 DPL，这就是我们说的那种 特权级别，描述了特权级别。 通过这个中断描述符呢 我们得到了一个段选择符。 段选择符的格式呢我们来看一下，段选择符其实就是个索引， 它有一个索引，当然了它还有一个在哪个表索引，是 GDT 表还是 LDT 表？那么还有一个特权级，所以 段选择符也有特权级。 有了这个索引之后呢，我们用它来什么呢？查 这张表。 这张表叫做全局描述符表。 我们就中断处理程序的相关信息放在这张表的某一行。 那么查到这张表之后呢， 我们就得到了一个另外一个 描述符，我们叫做段描述符。 这个段描述符当中也有，大家可以看也有这个 相应的权限，相应的权限，在这个 属性里头也有相应的权限。 我们看到每一个段都有相应的权限，表明它的特权级别。 但是在段描述符当中我们主要关注的呢是段的基地址。 有了这个段的基地址，有了在中断描述符当中的 偏移，我们就得到了 中断服务程序的一个入口地址。 所以中断服务程序的入口地址呢是 段基址再加上偏移。 那么段基址呢 是通过查 GDT 表得到的。 怎么查 GDT 表呢？是通过了段选择符做索引去查的。 是这样一个大致的过程。 我们把刚才那个过程呢 再用文字介绍一下。 首先，我们要确定 与中断和异常相关的中断向量， 某一个中断向量。 然后呢 去查中断描述符表，中断描述符表的 起始位置呢可以通过一个 IDTR 寄存器得到， 得到了这个中断描述符呢是在这个表当中的第 i 项，第 i 项。 然后我们再去查 GDT 表，首先先从 GDTR 寄存器得到了 GDT 的地址， 结合中断描述符表当中的 中断描述符当中的段选择符， 在 GDT 表当中呢查到对应的 段描述符。 从这个段描述符当中呢得到了我们中断 或者异常处理程序的段的基地址。 这个过程当中呢其实是要做很多的特权级检查。 刚才我们看到了不同的描述符里头都有特权级，所以要做相应的特权级检查。 下面要检查是否发生了特权级的变化。 如果是有特权级的改变，则要进行堆栈的切换。 因为我们要用的堆栈必须是与新的特权级相关的堆栈。 比如说你从用

户态进到了内核态， 原来的堆栈指针指向的是用户态，用户栈， 现在我要把堆栈的指针指向内核栈， 然后才能够做下面的工作。 那么下面的工作呢实际上就是硬件先压栈， 把一些信息保存在堆栈里头。 如果异常产生了，刚才讲的异常产生了一些出错码，也要把这个出错码 把它保存在堆栈中，当然这是某些异常会有这个出错码。 我们如果是中断门进来的，那么就要把 IF 位清掉，那么禁止中断，禁止下面的中断了。 当然如果是陷阱门进来就不需要做这件事。 那么通过中断描述符当中的段内偏移量 还有段描述符当中的基地址，我们就可以 算出来中断／异常处理程序的一个入口地址，然后呢去执行 它的第一条指令。 再用这张图 把刚才的过程走一遍。 得到了中断描述符，通过中断描述符当中的段选择址 和 GDTR 当中的地址去查 GDT 表，得到了段的描述符。 段的描述符当中我们就得到了段的基地址， 再从中断描述符当中得到的偏移量 相结合，就得到了中断处理程序的入口地址。 这就是在内存中的中断处理程序。