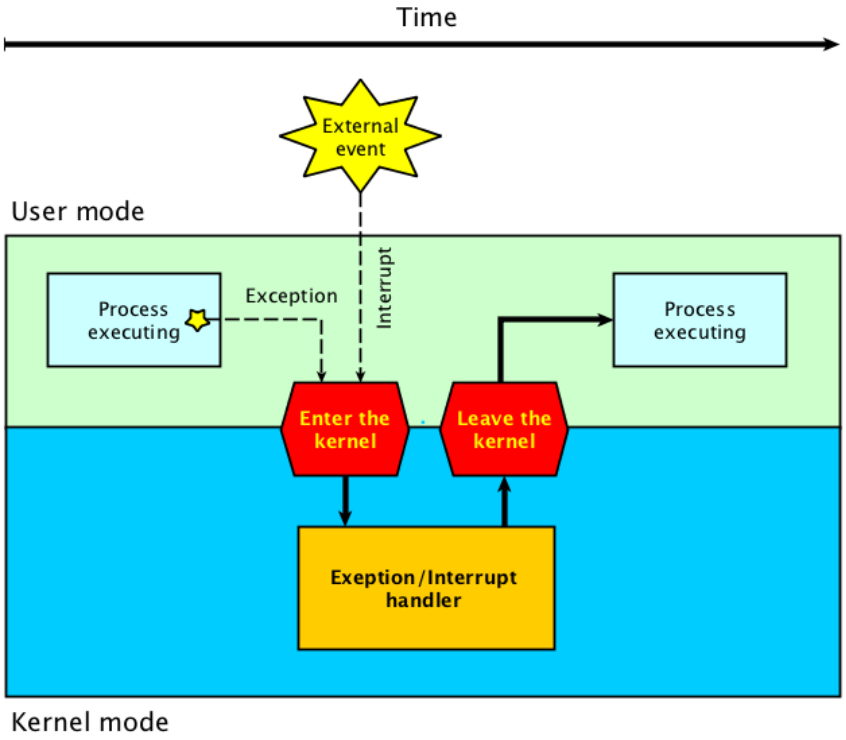
5.1 中断机制

西安邮电大学

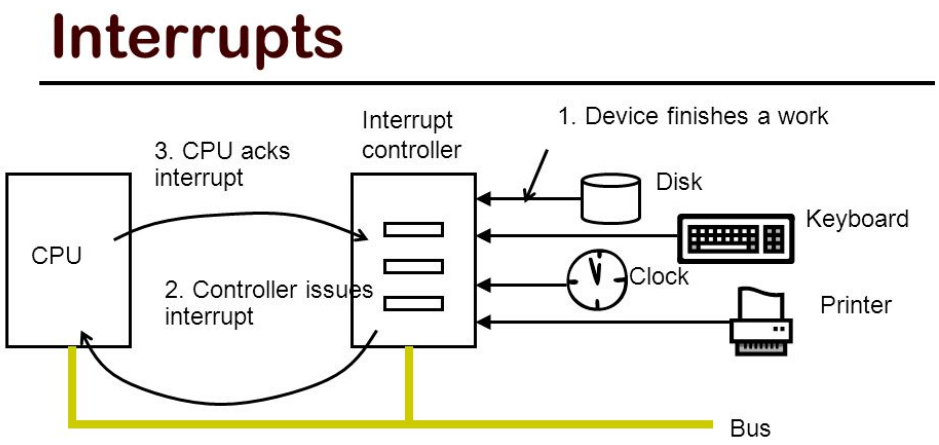
中断是什么？



中断是CPU对系统发生的某个事件作出的一种反应

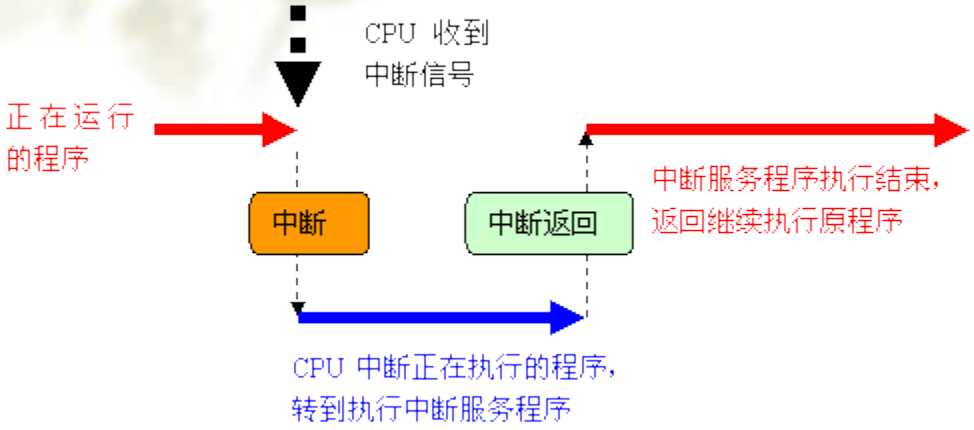
当中断发生时，CPU暂停正在执行的程序，保留现场后自动转去执行相应事件的处理程序，处理完成后返回断点，继续执行被打断的程序。听起来很简单，实际上中断是操作系统的脉搏，是并发处理的基础，远不像概念这么简单。

为什么引入中断



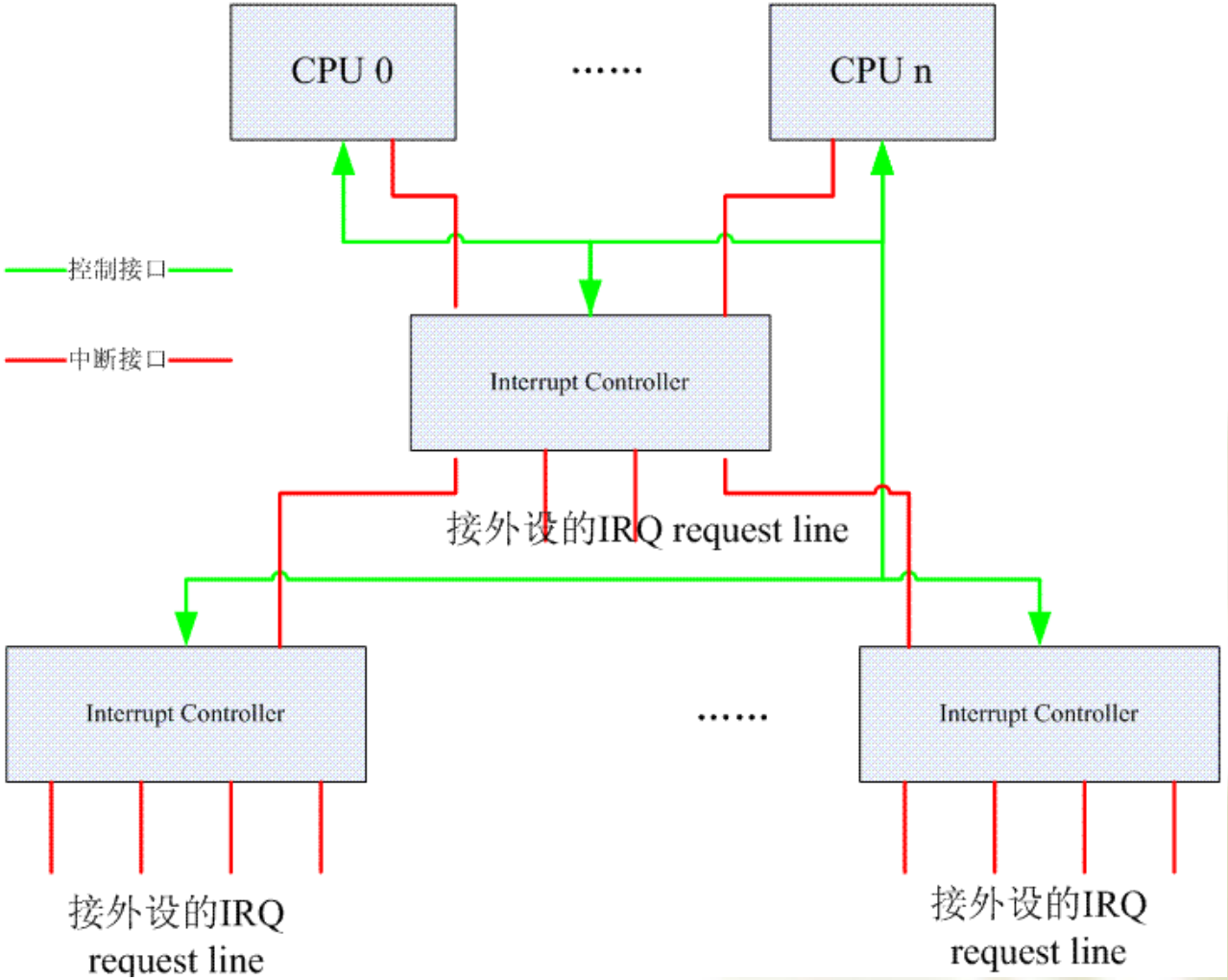
中断的引入，为了支持CPU和设备之间的并行操作。
（为什么可以并行？当CPU启动设备进行输入/输出后，设备便可以独立工作，CPU转去处理自己的事情；当设备完成输入/输出后，通过向CPU发中断报告此次输入/输出的结果，让CPU决定如何处理以后的事情），这些从概念上说起来也是比较简单的，工程实践起来也远没有这么简单

CPU什么时候响应中断



CPU收到中断信号，并不立即响应，而是在执行每条指令周期的最后一个时钟周期，一旦检测到中断信号有效，并且中断允许标志置1时，cpu才在当前指令执行完后，转入中断响应周期。

中断模型-C/S结构

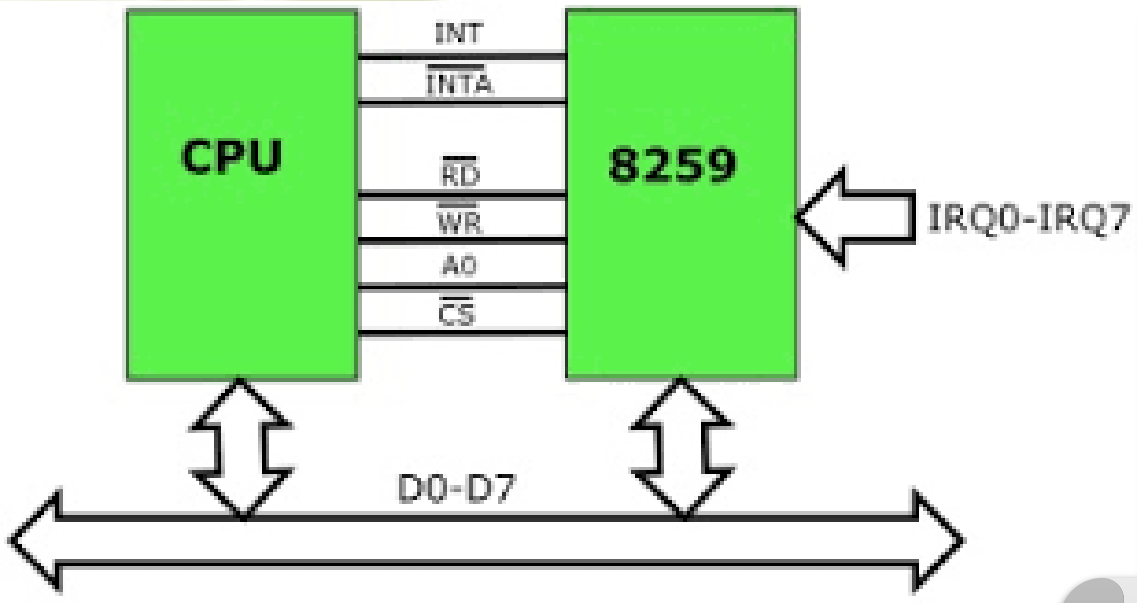

CPU 0
……
CPU n
控制接口
中断接口
Interrupt Controller
接外设的IRQ request line
Interrupt Controller
……
Interrupt Controller
接外设的IRQ
request line
接外设的IRQ
request line

中断模型解释

系统中有若干个CPU用来接收中断事件并进行处理，若干个中断控制器形成树状的结构，汇集系统中所有外设的中断请求线，并将中断事件分发给某一个CPU 进行处理。

如果对此模型进行简化，实际上是一种C/S模型，外设发出请求，这个请求并不是马上传给CPU，而是由中断控制器进行收集，中断控制器相当于中介，在外设与CPU之间架起桥梁。当CPU接受到请求后，给予应答。

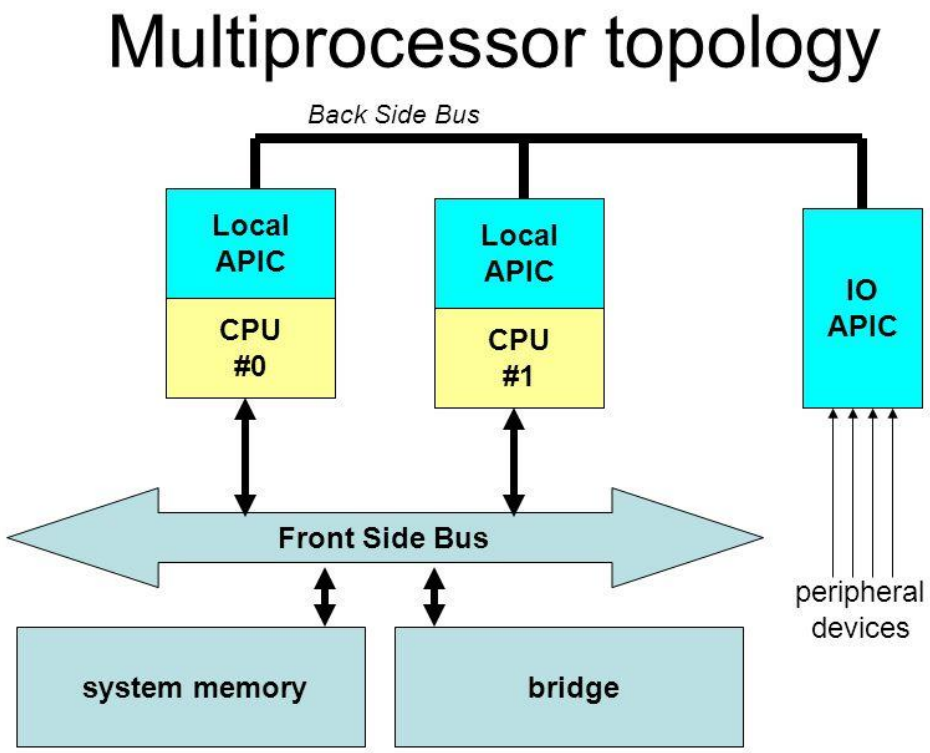
作为中介的中断控制器



在此，并不对中断控制器给予详细介绍，只是以最简单的8259中断控制器为例，说明中断请求IR0~IR7是从外设发来的，然后通过中断控制器的INT引脚向CP发出中断请求，然后CPU通过中断应答引脚INTA应答请求。目前x86采用的是APIC（高级可编程控制器）。

以X86的8259中断控制器为例

高级可编程中断控制器（APIC）



每个x86的核有一个本地APIC，这些本地 APIC 通过中断控制器通信（Interrupt Controller Communication）总线连接到IO APIC上。IO APIC收集各个外设的中断，并翻译成总线上的信息，传递给某个CPU上的本地APIC。

机制与策略分离的中断机制

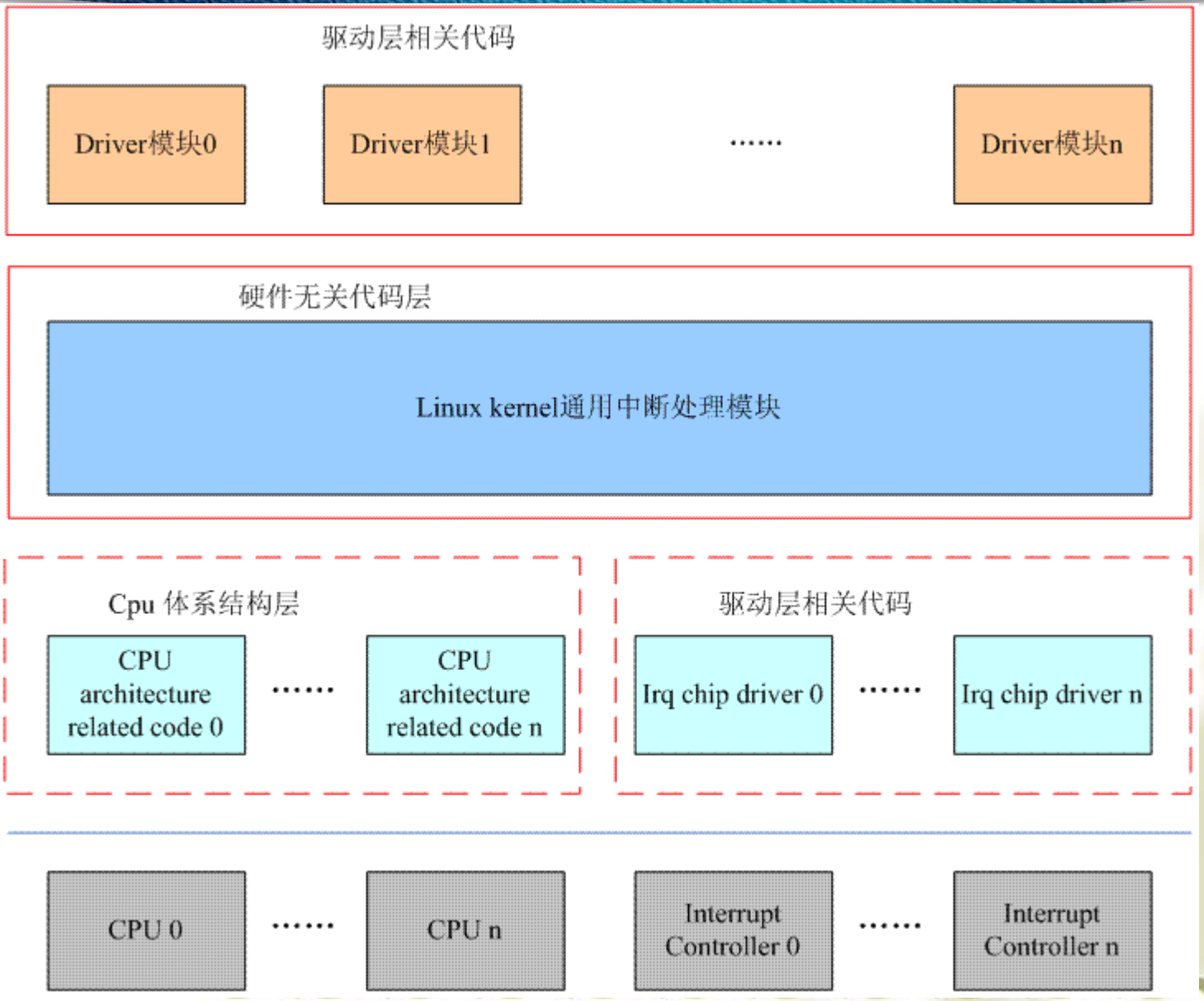
1. 尽管中断与CPU密切相关，但是CPU的设计独立于中断控制器的设计。

2. 尽管中断是操作系统非常重要的组成部分，但是操作系统的设计者只负责提供接口，通过该接口可以调用针对具体设备的中断服务程序。

3. 中断和对中断的处理被解除了耦合。无论是你在需要加入新的中断时，还是在你需要改变现有中断的服务程序时、又或是取消对某个中断支持的时候，CPU架构和操作系统都无需作改变，这其中的功臣就是中断控制器。

参见Linux内核之旅上中断趣味谈
(http://www.kerneltravel.net/journal/viii/index.htm)

Linux内核中断子系统的相关软件框架



中断子系统分成4个部分

（1）硬件无关的代码，我们称之为Linux 内核通用中断处理模块。无论是哪种CPU，哪种中断控制器，其中断处理的过程都有一些相同的内容，这些相同的内容被抽象出来。此外，各个外设的驱动代码中，也希望能用一个统一的接口实现中断相关的管理。

（2）CPU 体系结构相关的中断处理。 和系统使用的具体的CPU 体系结构相关。

（3）中断控制器的驱动代码。和系统使用的中断控制器相关。

（4）普通外设的驱动。这些驱动将使用Linux 内核通用中断处理模块的API来实现自己的驱动逻辑。

中断向量—中断源的类型

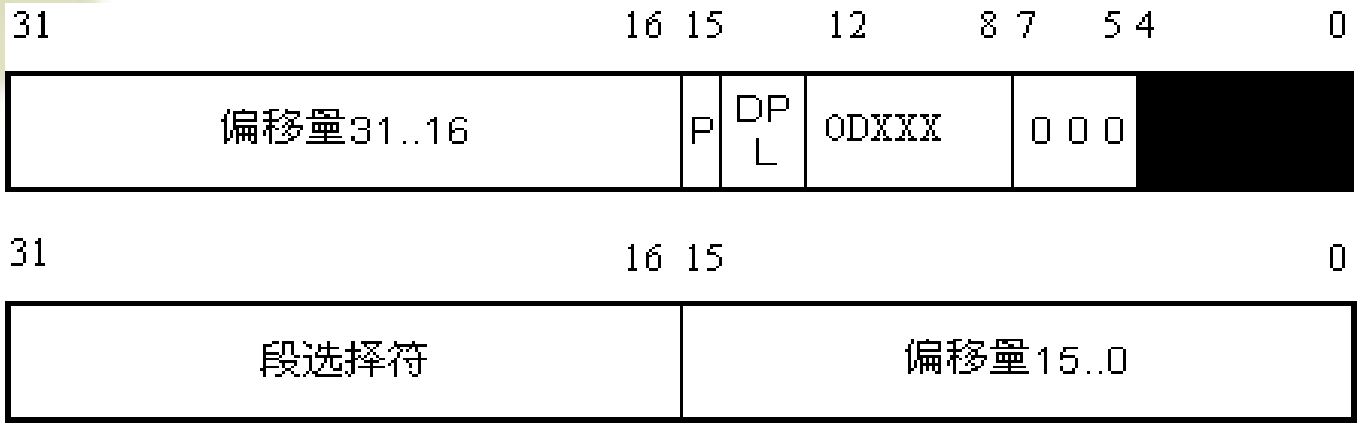
- **中断向量**—每个中断源都被分配一个8位无符号整数作为类型码，即中断向量

中断向量I=f(irq) （x86中 ：I=32+irq）

向量范围	用途
0～19	不可屏蔽中断和异常
20～31	Intel保留
32～127	外部中断（IRQ）
128（0x80）	用于系统调用的可编程异常
129～238	外部中断
239	本地APIC时钟中断
240	本地APIC高温中断
241～250	Linux保留
251～253	处理器间中断
254	本地APIC错误中断
255	本地APIC伪中断

中断描述符表

- 中断描述符表（IDT）：即中断向量表，每个中断占据一个表项



DPL	段描述符的特权级
偏移量	入口函数地址的偏移量
P	段是否在内存中的标志
段选择符	入口函数所处代码段的选择符
D	标志位，1=32位，0=16位
XXX	3位门类型码

中断描述符表

在实地址模式中，CPU把内存中从0开始的1K字节作为一个中断向量表。表中的每个表项占四个字节，由两个字节的段地址和两个字节的偏移量组成，这样构成的地址便是相应中断处理程序的入口地址。但是，在保护模式下，由四字节的表项构成的中断向量表显然满足不了要求。这是因为，①除了两个字节的段描述符，偏移量必用四字节来表示；②要有反映模式切换的信息。因此，在保护模式下，中断向量表中的表项由8个字节组成，如图所示，中断向量表也改叫做中断描述符表IDT（Interrupt Descriptor Table）。其中的每个表项叫做一个门描述符（gate descriptor）。

中断描述符表中门

“门”的含义是当中断发生时必须先通过这些门，然后才能进入相应的处理程序。

其中类型占3位，表示门描述符的类型，主要门描述符为：

(1) 中断门（Interrupt gate）

其类型码为110，中断门中的请求特权级（DPL）为0，因此，用户态的进程不能访问Intel的中断门。所有的中断处理程序都由中断门激活，并全部限制在内核态。

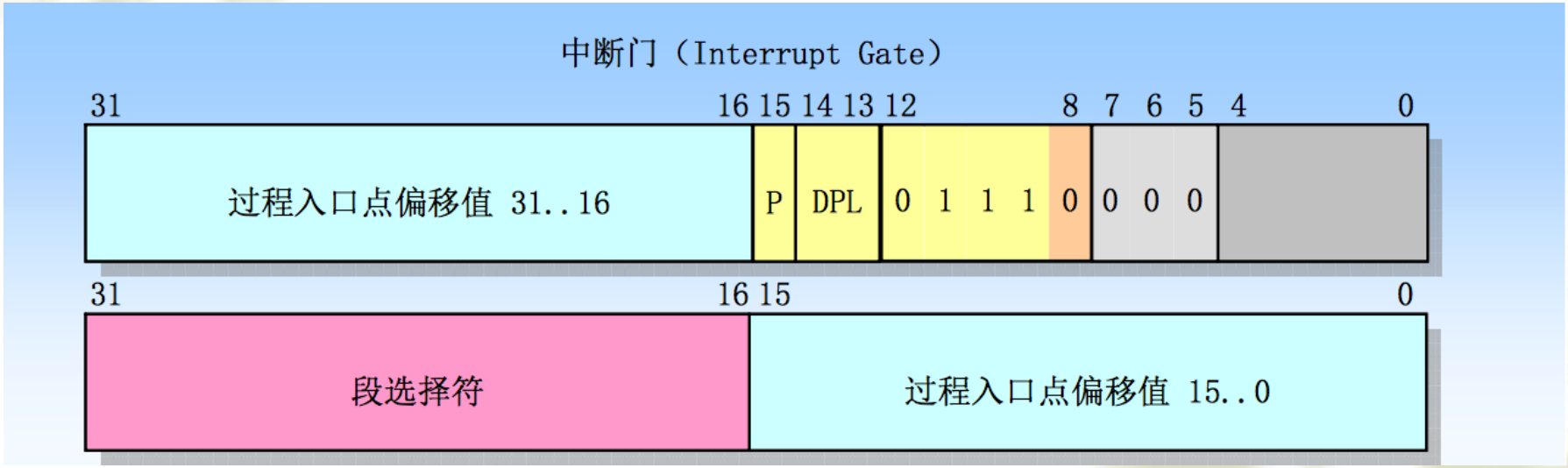
(2) 陷阱门（Trap gate）

其类型码为111，与中断门类似，其唯一的区别是，控制权通过陷阱门进入处理程序时维持中断标志位（IF）不变，也就是说，不关中断。

(3) 系统门（System gate）

这是Linux内核特别设置的，用来让用户态的进程访问Intel的陷阱门，因此，门描述符的DPL为3。系统调用就是通过系统门进入内核的。

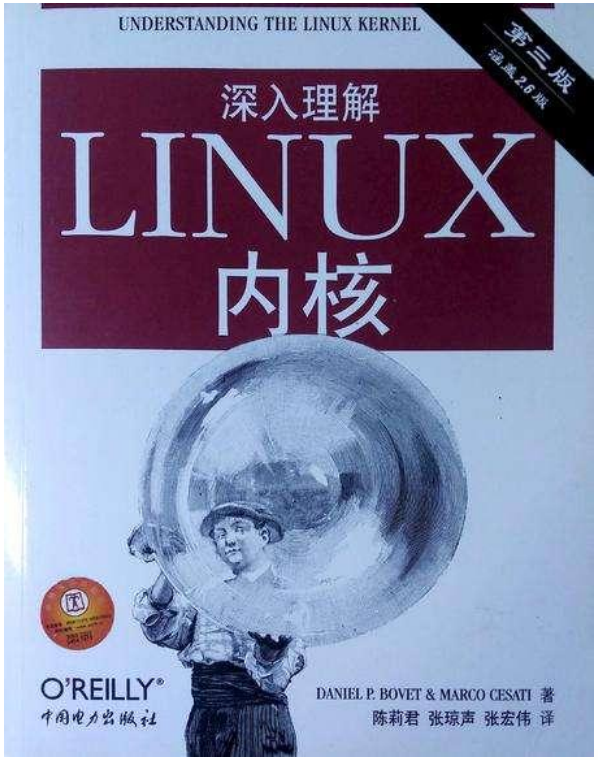
中断描述符表中的门


中断门（Interrupt Gate）
31
16 15 14 13 12
8 7 6 5 4
0
过程入口点偏移值 31..16
P
DPL
0 1 1 1 0
0 0 0
31
16 15
0
段选择符
过程入口点偏移值 15..0

动手实践

- Linux内核之旅网站 http://www.kerneltravel.net/
- 电子杂志栏目第八期“中断”，将向读者依次解释中断概念，解析Linux中的中断实现机理以及Linux下中断如何被使用。

参考资料



深入理解Linux内核 第三版第四章

Linux内核设计与实现 第三版第七章

http://www.wowotech.net/ ，蜗窝科技网站关于中断有一系列的文章，非常详细的介绍了中断的方方面面。中断是一个看似简单，但工程性非常强的部分，因此，希望大家务必阅读大量资料，并动手实践。

谢谢大家！

THANK YOU