# 中断和异常处理

## 中断和异常处理概述

本章是保护模式下的处理器处理中断和异常的机制

中断和异常是强制性的执行流的转移，从当前正在执行的程序或任务转移到一个特殊的称作句柄的例程或任务

**中断**：当硬件发出信号时，便产生中断，中断的产生同正在执行的程序是异步的，即中断的产生是随机的。其用于处理处理器的外部事件，比如为外设服务的请求。使用INT n 指令，软件也可以产生中断。

**异常**：异常是在处理器执行指令的过程中发现错误而产生的，比如除数为0，处理器可以检测出多种不同的错误，包括保护异常，页错误，内部机器错误。

**处理机制**：当处理器收到中断信号或监测到异常时，便挂起当前正在运行的进程或任务，而转去执行中断或异常处理例程。中断或异常处理例程执行完之后，处理器继续被中断的进程或任务。被中断的进程或任务继续执行，就像从未被打断一样，但有两种情况例外：无法从发生的异常恢复，中断使当前的程序终止。

处理器的中断和异常处理机制使中断和异常的处理对于应用程序和操作系统或可执行程序来说是透明的。

两种模式下，中断向量表不同：

**实模式下**：称作中断向量表，每个条目是16位的段偏移地址对，只能寻址1MB的物理内存空间

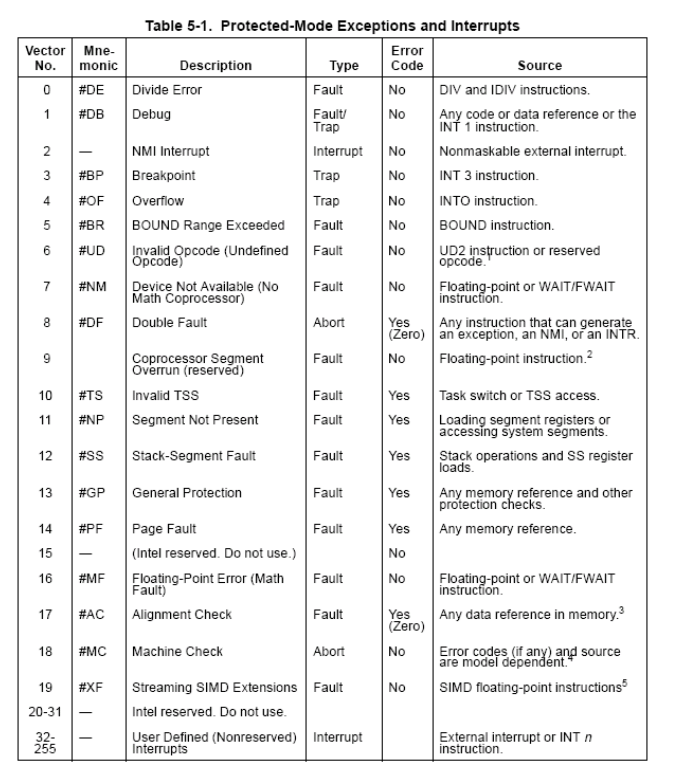
**保护模式下**：称作中断描述符表（IDT），每个IDT描述符包括一个选择子和一个偏移地址。保护模式下的中断向量表更加灵活和强大。

## 有关中断和异常了解性的内容

**中断和异常向量**：

处理器为每个异常和中断分配了一个识别码，称作**向量**；

向量号从0到31被分配给异常和NMI中断使用，从32到255之间的向量号提供给用户使用，这些中断不在Intel的保留部分之列，一般被分配给外部I/O设备，允许他们通过某个外部硬件中断机制向处理器传递信号



**中断源和异常源：**

**中断**有两个来源：

1. **硬件产生的中断**：通过处理器的引脚接收，也可以通过局部APIC串行总线接收；局部APIC的LINT有两个引脚，当局部APIC被关闭时，这两个引脚分别被配置为INTR和NMI引脚；当局部APIC打开时，可通过APIC向量表对LINT引脚编程，使其和处理器的任何异常和中断向量绑定；信号由INTR引脚或局部APIC传递到处理器的外部中断称作可屏蔽硬件中断。通过NMI引脚传递的则被称作不可屏蔽中断。
2. **软件产生的中断**：将中断向量号作为INT指令的操作数即可通过INT指令在程序中产生中断。

**异常**有三个来源：

1. **处理器检测到的程序错误异常**：在应用程序执行过程中，或操作系统执行中，当检测到程序错误时，处理器产生一个或多个异常，每个异常都有一个向量号
2. **软件产生的异常**：INTO、INT3和BOUND指令允许在软件中产生异常，这些指令允许在指令流中监测指定的异常条件
3. **机器监测异常**：当监测到一个机器检测错误时，处理器发出一个机器检测异常，并返回一个出错码

**异常的分类：故障、陷进和中止**

1. **错误**：错误通常能够被修复，一旦修正，程序能够不失连续性地接着执行，当报告错误发生时，处理器将机器状态恢复到执行错误之前的状态，错误处理例程的返回地址指向产生错误的指令，而不是产生错误指令之后的那条指令
2. **陷阱**：当引起陷阱的指令发生时，马上产生异常，陷阱允许程序不失连续性的继续执行。陷阱处理例程的返回地址指向引起陷阱指令的下一条指令
3. **终止**：并不总是报告产生异常的指令的确切位置，也不允许引起终止的进程或任务重新执行，用来报告严重错误。

**程序或任务的重新执行**

除“**终止**”之外的所有异常均严格地在前一条指令结束而下一条指令未开始时被报告，中断也是在该时刻被检测的

**错误类**的异常，返回地址指向产生错误的指令。所以当从错误处理例程返回时，产生

错误的指令将重新被执行。

**陷阱类**异常，返回地址指针指向的是产生陷阱指令的下一条指令。当一条转移指令

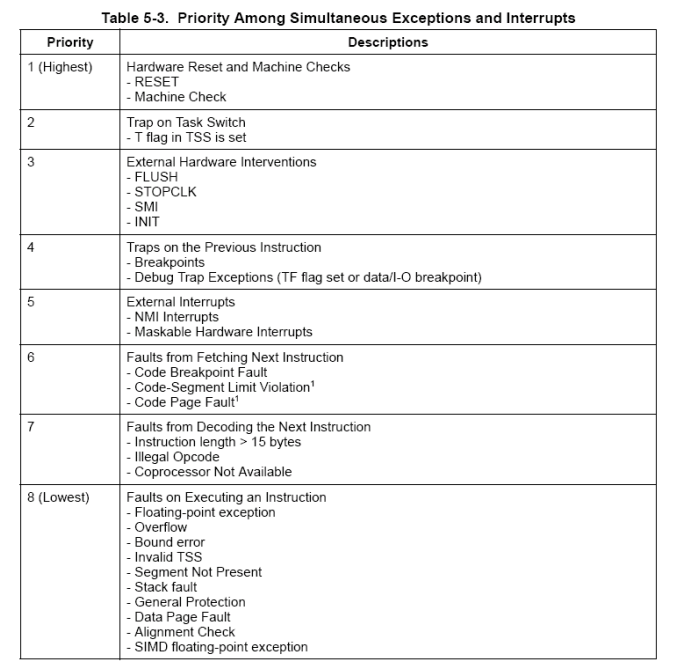
执行过程中检测到陷阱时，返回地址指针则反映了执行转向的情沉。

**终止类**异常不支持进程或任务的继续执行。终止处理例程的作用是：当有终止异常发生时，收集处理器的各种相关诊断信息，并关闭进程或系统。

**中断**则绝对保证了在不失连续性的条件下，使被中断的进程和任务能继续执行。返回地址指针指向发生中断时的下一条指令。对于带重复前缀的指令，中断发生在两次循环之间。

**开启和禁止中断**：根据处理器的状态和EFLAGES的IF位和RF位，处理器可以禁止某些中断的产生

**异常和中断的优先级：**

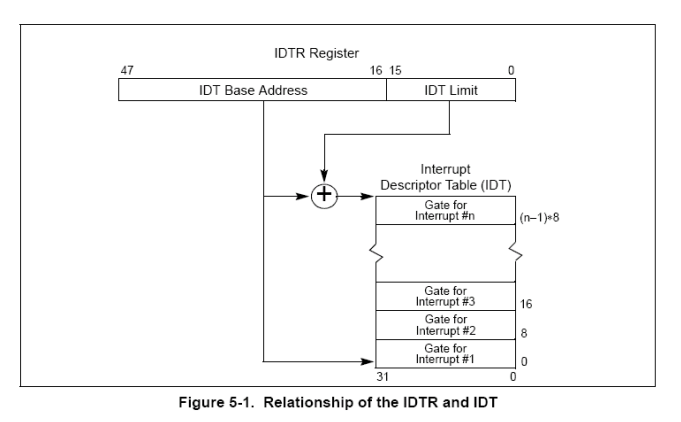


## 中断描述符表

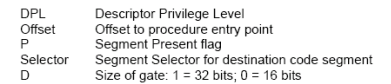
**如何构成**：中段描述符表将每一个中断异常向量和一个门描述符联系起来，用来处理相关的中断或是异常。与GDT不同的是，IDT的第一个条目是允许放置一个描述符的。为了在IDT中形成索引，处理器会以8为单位压缩中断异常向量，这样的话可以和门描述符的字节对齐。

**如何获得中断处理程序的地址**：通过中段描述符基地址加便宜的形式获得中断处理程序的地址。需要注意的是，IDT的基地址应该在一个8字节的边界处对齐。

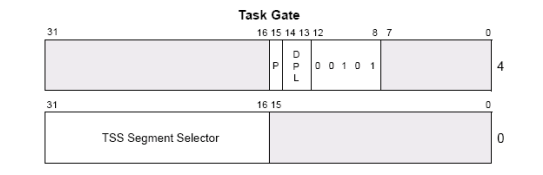
**如何设置中断描述符表寄存器**：可以分别使用指令LIDT和SIDT加载和存储IDTR的内容。LIDT指令可以基地址加操作数偏移来加载到IDRT，但此指令只能在CPL 0等级使用；SIDT指令可以将存储在IDTR中的基地址和偏移量存储到内存中。



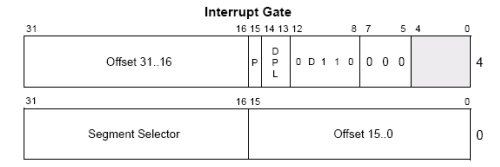
## IDT描述符



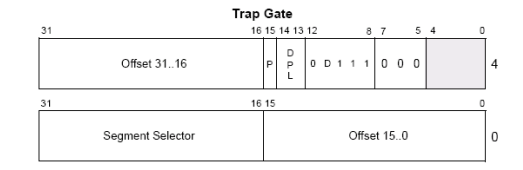
**任务门**：



**中断门**：



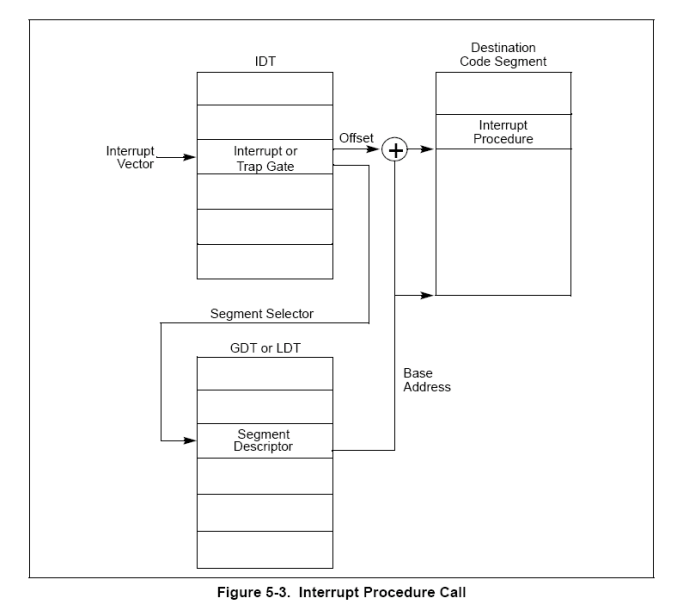
**陷阱门**：



## 中断与异常处理

中断过程调用的**流程**是怎样的？

当响应异常或中断时，处理器使用异常或中断向量作为IDT中描述符的索引。如果索引指向中断门或陷阱门，处理器调用异常或中断处理程序的方式类似于调用门的CALL；如果index指向一个任务门，处理器执行一个任务切换到异常或中断处理程序任务，方式类似于CALL到任务门。



如何判断中断处理过程与被中断任务的**优先级**？

中断描述符的权限等级和当前执行的任务CPL

不同优先级上，处理方式一样吗？

肯定不一样啊

如果**发生堆栈切换**，处理器会做哪些操作？

（1）处理程序要使用的堆栈的段选择器和堆栈指针从当前执行的任务的TSS中获得。在这个新的堆栈上，处理器推送被中断过程的堆栈段选择器和堆栈指针

（2）处理器然后保存EFLAGS, CS和EIP寄存器在新堆栈上的当前状态

（3）如果异常导致错误码被保存，则将错误码压入EIP值之后的新堆栈

如果**没发生堆栈切换**，处理器会做哪些操作？

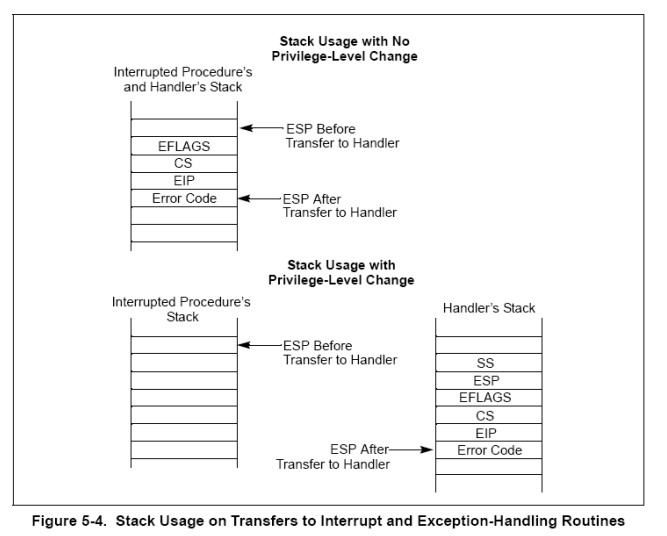
（1）处理器将EFLAGS、CS和EIP寄存器的当前状态保存在当前堆栈上。

（2）如果由于异常导致错误码被保存，则该错误码被压入当前堆栈的EIP值之后

中断处理过程后，**如何返回，处理器做了哪些操作**？

要从异常处理程序或中断处理程序返回，处理程序必须使用IRET(或IRETD)指令。

IRET指令与RET指令类似，除了它将保存的标志恢复到EFLAGS寄存器中。只有当CPL为0时，EFLAGS寄存器的IOPL字段才会恢复。只有当CPL小于或等于IOPL时，IF标志才会被改变。



异常和中断处理过程的**保护**

异常处理程序和中断处理程序的特权级保护类似于通过调用门调用的普通过程调用。处理器不允许在特权比CPL低的代码段中将执行转移到异常处理程序或中断处理程序。

异常和中断处理过程的**标志**使用方式

当通过中断门或陷阱门访问异常或中断处理程序时，处理器在将EFLAGS寄存器的内容保存在堆栈上之后清除EFLAGS寄存器中的TF标志。(在调用异常和中断处理程序时，处理器还清除EFLAGS寄存器中的VM、RF和NT标志，然后将它们保存在堆栈上。)清除TF标志可以防止指令跟踪影响中断响应。随后的IRET指令将TF(以及VM、RF和NT)标志恢复为堆栈上EFLAGS寄存器保存的内容中的值

中断门与陷阱门的唯一**区别**是什么？

处理器在EFLAGS寄存器中处理IF标志的方式不同