## 中断描述符表（Interrupt Descriptor Table，IDT）

中断描述符表（Interrupt Descriptor Table，IDT）将每个异常或中断向量分别与它们的处理过程联系起来。与GDT和LDT表类似，IDT也是由8字节长描述符组成的一个数组。

```
#pragma pack(push, 1)
//IDT的内存空间是一个数组。每个元素都有如下的结构
typedef struct P2C_IDT_ENTRY_ {
    P2C_U16         offset_low;
    P2C_U16         selector;
    P2C_U8         reserved;
    P2C_U8         type:4;
    P2C_U8         always0:1;
    P2C_U8         dpl:2;
    P2C_U8         present:1;
    P2C_U16         offset_high;
}P2C_IDTENTRY, *PP2C_IDTENTRY;
#pragma pack(pop)
```

中断描述符表
可以包含描述符。为了构成IDT表中的一个索引值，处理器把异常或中断的向量号乘以8。因为最多只
有256个中断或异常向量，所以IDT无需包含多于256个描述符。IDT中可以含有少于256个描述符，因为
只有可能发生的异常或中断才需要描述符。不过IDT中所有空描述符项应该设置其存在位（标志）为0

```
//
//现在这个中断门没有用了，设置type = 0使之空闲
//
idt_addr[old_id].type = 0;
```

。
IDT表可以驻留在线性地址空间的任何地方，处理器使用IDTR寄存器来定位IDT表的位置。这个寄存器
中含有IDT表32位的基地址和16位的长度（限长）值。IDT表基地址应该对齐在8字节边
界上以提高处理器的访问效率。限长值是以字节为单位的IDT表的长度。

```
#pragma pack(push, 1)
//从sidt指令获得一个如下的结构，从这里可以得到IDT的开始地址
typedef struct P2C_IDTR_ {
    P2C_U16         limit; // 范围
    P2C_U32         base; //基地址
}P2C_IDTR, *PP2C_IDTR;
#pragma pack(pop)
```

中断描述符表IDT和寄存器IDTR
**LIDT和SIDT指令分别用于加载和保存IDTR寄存器的内容。**
**LIDT**指令用于把内存中的限长值和基地址操作数加载到IDTR寄存器中。该指令仅能由当前特权级CPL是0的代码执行，通常被用于创建IDT时的操作系统初始化代码中。
**SIDT**指令用于把IDTR中的基地址和限长内容复制到内存中。该指令可在任何特权级上执行。

```
VOID    *p2cGetIdt()
{
    PAGED_CODE();

    P2C_IDTR idtr;

    //
    //一句汇编读取到IDT的位置
    //
    _asm sidt idtr;
    return (void *)idtr.base;
}
```

如果中断或异常向量引用的描述符超过了IDT的界限，处理器会产生一个一般保护性异常
**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，□除了两个字节的段描述符，偏移量必用四字节来表示；,要有反映模式切换的信息

# 12.3 中断

CPU 用 8 位的中断类型码通过中断向量表找
什么是中断向量表呢？中断向量表就是中断向量
中断向量，就是中断处理程序的入口地址。展开
口地址的列表。

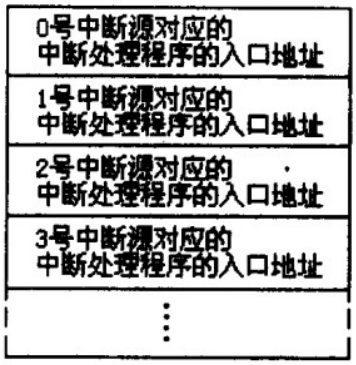


图 12.1 中断向量表

中断向量表
所对应的中断处

可以看到，
断类型码作为中
而得到中断处理

可见，CPU
就可以得到中断
个首要的问题是
中断向量表成了通过中断类型码找到中断处理程

中断向量表在内存中存放，对于 8086PC 机，

内存 0000:0000 到 0000:03E8 的 1000 个单元中
能，如果使用 8086CPU，中断向量表就必须放
因为 8086CPU 就从这个地方读取中断向量表。

那么在中断向量表中，一个表项占多大的
是一个中断处理程序的入口地址，对于 8086CP
所以一个表项占两个字，高地址字存放段地址，

**在保护模式下**，中断向量表中的表项由8个字节组成，中断向量表也改叫做中断描述符表IDT（InterruptDescriptor Table）。其中的每个表项叫做一个门描述符（gate descriptor），“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。

```
#pragma pack(push, 1)
//IDT的内存空间是一个数组。每个元素都有如下的结构
typedef struct P2C_IDT_ENTRY_ {
    P2C_U16         offset_low;
    P2C_U16         selector;
    P2C_U8         reserved;
    P2C_U8         type:4;
    P2C_U8         always0:1;
```

```
    P2C_U8          dpl:2;
    P2C_U8          present:1;
    P2C_U16          offset_high;
}P2C_IDTENTRY, *PP2C_IDTENTRY;
#pragma pack(pop)
```

主要门描述符是：
· 中断门（Interrupt gate）
其类型码为110,中断门包含了一个中断或异常处理程序所在段的选择符和段内偏移量。当控制权通过中断门进入中断处理程序时，处理器清IF标志，即关中断，以避免嵌套中断的发生。中断门中的DPL（Descriptor Privilege Level）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。
· 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IF标志位不变，也就是说，不关中断。
· 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。通过系统门来激活4个Linux异常处理程序，它们的向量是3、4、5及128，也就是说，在用户态下，可以使用int3、into、bound 及int0x80四条汇编指令。
最后，在保护模式下，中断描述符表在内存的位置不再限于从地址0开始的地方，而是可以放在内存的任何地方。为此，CPU中增设了一个中断描述符表寄存器IDTR，用来存放中断描述符表在内存的起始地址。中断描述符表寄存器IDTR是一个48位的寄存器，其低16位保存中断描述符表的大小，高32位保存IDT的基址.