福昕高级PDF编辑器

高效·安全·专业

立即下载　　点击购买

- OFFICE格式互转
- OCR文字识别
- 文本图像编辑
- 加密和签署
- 交互式动态表单
- 互联PDF文档

福昕高级PDF编辑器

高效·安全·专业

立即下载　　点击购买

- OFFICE格式互转
- OCR文字识别
- 文本图像编辑
- 加密和签署
- 交互式动态表单
- 互联PDF文档

实模式和保护模式下的中断

中断

IDT

对应每个中断源设置一个向量。这些向量顺序存在主存储器的特定存储区。向量的内容是相应中断服务程序的起始地址和处理机状态字。在响应中断时，由中断系统硬件提供向量地址，处理机根据该地址取得向量，并转入相应的中断服务程序

中断类型码

- 我们把每个中断服务程序进行编号，这个号就代表一个中断服务程序，就是中断类型码。这个中断类型码是计算机用来查找中断向量用的。

- 中断指令的一般格式为“INT n”，其中，n被称为“中断类型码”

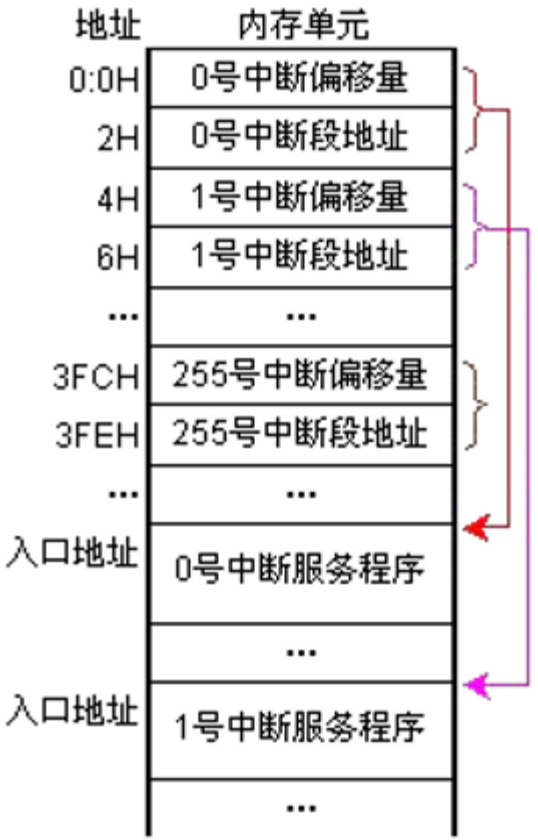
```
DispStr:
    mov ax, BootMessage
    mov bp, ax              ; ES:BP = 串地址
    mov cx, 16              ; CX = 串长度
    mov ax, 01301h          ; AH = 13,  AL = 01h
    mov bx, 000ch           ; 页号为0(BH = 0) 黑底红字(BL = 0Ch,高亮)
    mov dl, 0
    int 10h             ; 10h 号中断
    ret
BootMessage:            db  "Hello, OS world!"
times   510-($-$$)      db  0   ; 填充剩下的空间，使生成的二进制代码恰好为512字节
dw  0xaa55                      ; 结束标志
```

中断向量表

- 中断向量表是指中断服务程序入口地址的偏移量与段基值，一个中断向量占据4字节空间。中断向量表是8086系统内存中最低端1K字节空间，它的作用就是按照中断类型号从小到大的顺序存储对应的中断向量，总共存储256个中断向量。
- 中断向量表在内存单元的最低处，地址空间为00000H----003FFH(0-1024B)
- 这个地址正好和中断类型码有一种对应的关系：中断类型码*4(一个中断向量所占的空间) 就等于这个中断向量的首地址。

中断向量表

- 每一个中断向量所包含的地址以低位二字节存储偏移量，高位二字节存储段地址；
- 中断类型号×4=存放中断向量的首地址；
- 按照实模式的寻址方式找到对应的中断处理的入口；
- 在全部256个中断中，前32个（0—31）为硬件系统所预 留，后224个可由用户自定义；



中断（实模式）

- 中断指令”INT n”表示调用n号中断处理程序，在中断处理程序中，用中断返回指令IRET （interrupt return）指令使CPU返回主程序断点继续执行；
- 中断指令 “INT n” 和调用程序指令”CALL”很相似，它们均转入内存中其它程序段执行，执行完后再返回

INT指令

1. SP（Stack Pointer 堆栈指针）中的值减2，标志位寄存器的值入栈——保存中断前的状态

2. 标志位TF和IF清0——关闭中断 IF=0，CPU不响应外部的可屏蔽中断请求；TF=0，则处于连续工作模式

3. SP减2，把返回地址的段值（CS）推入堆栈

4. SP减2，把返回地址的偏移量（IP）推入堆栈

5. 根据中断类型码n，从中断向量表中取得中断处理程序地址，取得的段地址存入CS，偏移量 存入IP。从而使CPU转入中断处理程序运行。

IRET指令

1. 从堆栈中取出一字（INT指令保存的返回地址偏移量），送给 IP，然后使SP加2

2. 从堆栈中取出一字（INT指令保存的返回地址段值），送给 CS，然后使SP加2

3. 从堆栈中取出一字（INT指令保存的标志寄存器的值），送给 标志寄存器，然后使SP加2 IRET执行后，CPU返回到INT指令后面的一条指令

其实同函数调用call和ret相类似，在调用时保存返回地址和标志位，但同时还会设置屏蔽请求。iret时则还原调用前状态。

IRET指令

1. 从堆栈中取出一字（INT指令保存的返回地址偏移量），送给 IP，然后使SP加2

2. 从堆栈中取出一字（INT指令保存的返回地址段值），送给 CS，然后使SP加2

3. 从堆栈中取出一字（INT指令保存的标志寄存器的值），送给 标志寄存器，然后使SP加2 IRET执行后，CPU返回到INT指令后面的一条指令

其实同函数调用call和ret相类似，在调用时保存返回地址和标志位，但同时还会设置屏蔽请求。iret时则还原调用前状态。

保护模式下的中断处理

- 保护模式下的中断处理与实模式下的中断处理最大区别在于寻找中断处理代码入口的方式
- 在保护模式下，为每一个中断和异常定义了一个中断描述符，来说明中断和异常服务程序 的入口地址的属性
- 由中断描述符表取代实地址模式下的中断向量表

- 中断描述符除了含有中断处理程序地址信息外，还包括许多属性和类型位
- 每个中断描述符占用连续的8个字节，中断描述符分为三类：任务门、中断门和自陷门，CPU对不同的门有不同的处理方式

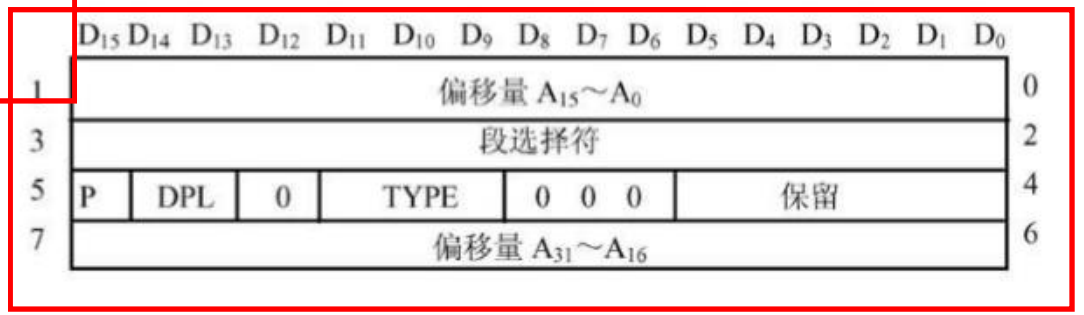
中断描述符(cont.) continue

- 低地址的0和1两个字节是中断代码的偏移量A15～A0；高地址的6和7两个字节是中断代码的偏移量A31～A16；
- 2和3两个字节是段选择符，段选择符和偏移量用来形成中断服务子程序的入口地址；
- 4和5两个字节称为访问权限字节，它标识该中断描述符是否有效、服务程序的特权级和描述符的类型等信息；

I. P（present）：表示中断描述符的有效性；

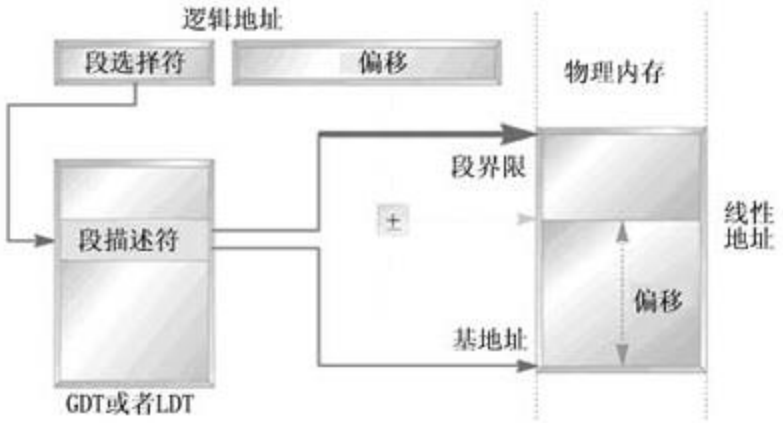
II. DPL（descriptor privilege level）；

III. TYPE：指示中断描述符的不同类型



中断描述符表（IDT）

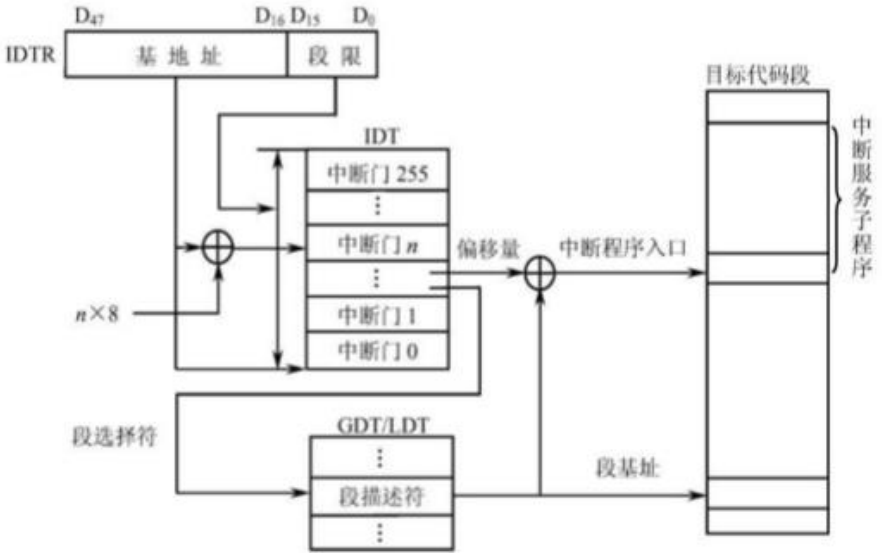
- 在80x86系列中为中断服务提供中断/陷阱描述符，这些描述符构成中断描述符表（IDT）
- 引入一个48位的全地址寄存器（即中断描述符表寄存器IDTR）存放IDT的内存地址，因此不再限于底部1K位置
- 和GDTR一样，IDTR包含32位的基地址和16位段限，基地址定义中断描述符表IDT在存储器中的起始点，段限定义中断描述符表所占的字节个数
- 理论上IDT表同样可以有8K项，可是因为80x86只支持256个中断，因此IDT实际上最大只能有256项（2K大小）



中断（保护模式）

保护模式下的中断过程则较为复杂，它要借助中断门描述符来获取中断子程序这个目标 段的描述符，也就是说必须经过两次查表才能获得中断服务子程序的入口地址

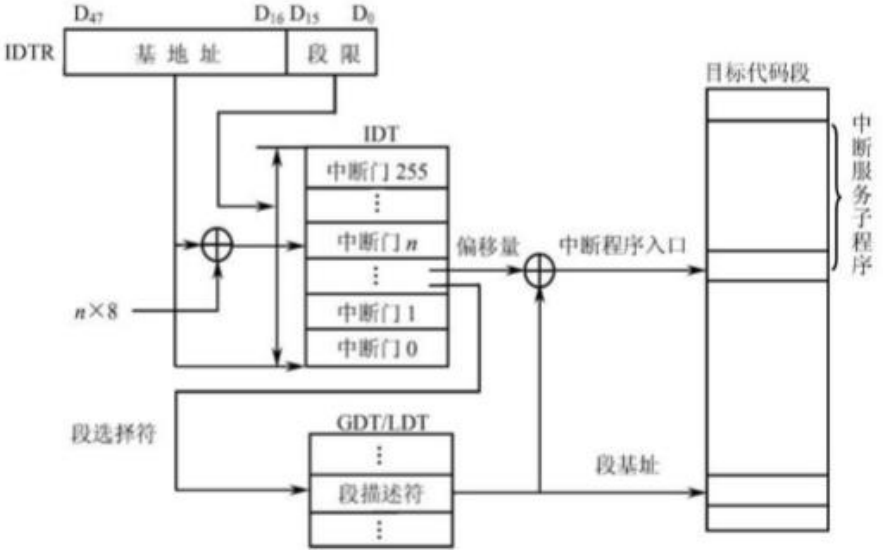
1. 装载中断描述符表寄存器CPU切换到保护模式之前，运行于实模式下的初始化程序必须使用LIDT指令装载中断描述符表IDT，将IDT基地址与段界值装入IDTR。如果不完成这一步操作，系统就会100%崩溃。在返回实模式或系统复位时，IDTR中自动装入000000H的基地址值与03FFH的段界值。可见实模式的中断向量表是固定在存储器的最底部，而保护模式下的IDT则是可以改变的。



中断（保护模式）

2. 查中断描述符表以IDTR指定的中断描述符表的基地址为起始地址，用调用号N×8算出偏移量，即为N号中断门描述符的首地址，由此处取出中断门的8个字节

3.查全局或局部描述符表根据中断门中的选择子（段选择符）和偏移量得到中断处理程序入口



Thanks !