# 第三次实验问题清单

By 张洪胤

## 问题一：解释中断向量

中断向量将中断/异常与相应的处理方法对应起来。

每一种中断会对应一个中断向量号，这些向量会顺序的存储在主存储器的特定存储区。中断向量号通过中断向量表就可以和相应的中断处理程序对应起来。中断向量包含了中断服务程序的起始地址和处理状态字。

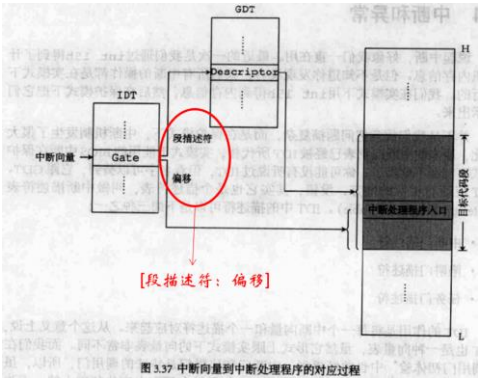


图 3.37 中断向量到中断处理程序的对应过程

## 问题二：解释中断类型码

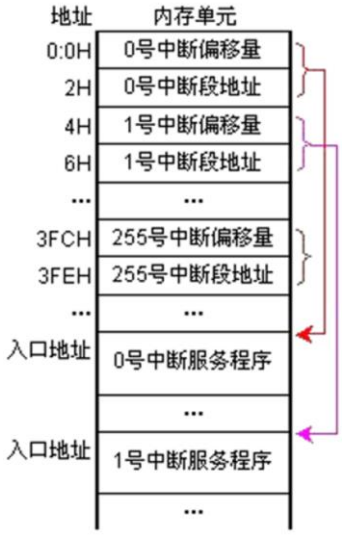
我们将每一个中断服务程序进行编号，这个编号代表一个中断服务程序，就是中断类型码。这个中断类型码是计算机用来查找中断向量的。中断指令一般格式为“Int n”，其中n我们称为中断类型码。

## 问题三：解释中断向量表

中断向量表是指中断服务程序入口地址的偏移量与段基址，一个中断向量占据4字节空间。中断向量表是8086系统内存中最低端的1K字节空间，其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，其中前32个为硬件系统预留，剩下的224个可由用户自定义。

地址空间:00000H-003FFH（0-1024B）,中断类型码 * 4 得到的就是这个中断向量的首地址。中断向量所包含的地址以低位2字节存储偏移量，高位2字节存储段地址。按照实地址的寻址方式找到对应的中断处理的入口。




IDTR：权限控制+中断向量放置位置允许放置到其他位置(保存中断向量表头地址)。

https://blog.csdn.net/regionyu/article/details/1708084

## 问题四：实模式下中断程序地址如何得到?

中断指令“INT n”表示调用n号中断处理程序，根据中断类型码n * 4得到中断向量首地址，从中断向量表中取得到中断的段基址和偏移，段地址存储到CS，而偏移量存入到IP，即得到中断程序的地址。

# 了解

INT指令

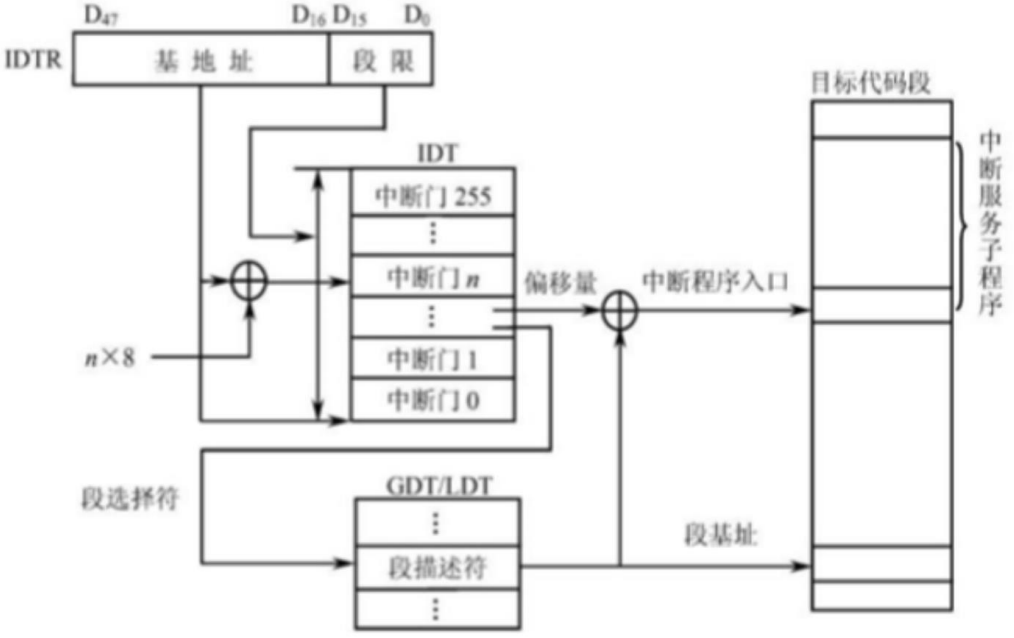
1. SP（Stack Pointer 堆栈指针）中的值减2，标志位寄存器的值入栈——保存中断前的状态
2. 标志位TF和IF清0——关闭中断 IF=0 ，CPU不响应外部的可屏蔽中断请求；TF=0，则处于连续工作模式
3. SP减2，把返回地址的段值（CS）推入堆栈
4. SP减2，把返回地址的偏移量（IP）推入堆栈
5. 根据中断类型码n，从中断向量表中取得中断处理程序地址，取得的段地址存入CS，偏移量 存入IP。从而使CPU转入中断处理程序运行。

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
4. 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。

## 问题五：保护模式下中断程序地址如何得到?

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基地址和段限装入IDTR，如果不完成这个则会导致系统崩溃。返回实模式或系统复位时，IDTR会自动装入000000H的基地址和03FFH的段限。
2. 查中断描述符表以IDTR指定的中断描述符表的基地址以起始地址，用调用号N * 8计算出偏移量，即为N好还在中断门描述符的首地址，由此取出中断门的8个字节
3. 查全局或局部描述符表根据还在中断门中的选择子（段选择符）和偏移量得到中断处理程序入口




## 问题六：中断向量的地址如何得到?

实模式：中断向量的地址可以通过中断类型码 * 4字节(一个中断向量大小)来获得。

保护模式：我们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，具体过程如问题五

**问题七：实模式下如何根据中断向量的地址得到中断程序地址?**

使用中断向量的地址（中断类型码 *4），拿到中断向量，解析出其中的段基址和偏移（高2字节是基址，低2字节是段偏移量），来形成中断程序地址。

**问题八：解释中断描述符**

中断描述符占用连续的8个字节，被分为三类：任务门、中断门和自陷门，CPU对不同的门有不用的处理方式

**中断描述符**

低地址的0和1字节是中断代码的偏移量A15-A0

高地址的6和7字节是中断代码的偏移量A31-A16。

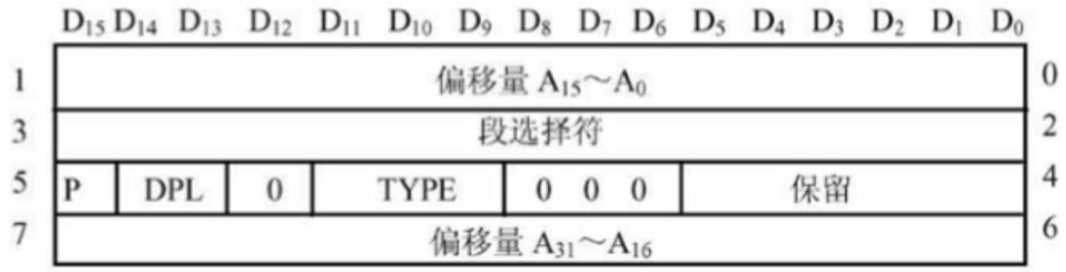
2和3字节是段选择符，段选择符和偏移量用来形成中断服务子程序的入口地址。

4和5字节是访问权限字节，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有效信息。

P(present)：标识中断描述符的有效性

DPL(Descriptor Privilege Level)

Type：指示中断描述符的不同类型




**问题九：保护模式下中断描述符表如何得到?**

装载中断描述符表寄存器CPU切换到保护模式前，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（32位）与段界值（16位）装入IDTR（一个48位的全地址寄存器，中断描述符表寄存器），基地址定义IDT在存储器中的起始点，段界值定义IDT所占的字节数。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。

**问题十：保护模式下中断门如何得到?**

查描述符表以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计算出偏移量，即为N号中断门描述符的首地址，由此取出中断门的8个字节。

**问题十一：保护模式下如何根据中断门得到中断处理程序地址?**

取出中断门中的8个字节，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

## 问题十二：中断的分类，举例不同类型的中断?

### 从中断源分类

1. 内部异常中断：由计算机硬件异常或故障引起的中断，CPU本身故障。
2. 软中断：由于程序执行中断指令引起的中断，当做trap处理，实现系统调用，使用Int或者Int3命令触发。
3. 外部中断或I/O中断：外部设备(如输入输出设备)请求引起的中断。
    a) 可屏蔽中断：禁止响应某个中断，保证在执行一些重要的程序中不响应中断，以免造成迟缓而引起错误。
    b) 不可屏蔽中断：重新启动、电源故障、内存出错、总线出错。

### 另一个角度分类

1. 中断：由CPU以外的事件引起的中断，如I/O中断、时钟中断、控制台中断等；
2. 异常：来自CPU的内部事件或程序执行中的事件引起的过程，如由于CPU本身故障、程序故障和请求系统服务的指令引起的中断等。

## 问题十三：中断与异常的区别?

参考问题二十一

## 问题十四：实模式和保护模式下的中断处理差别

实模式下的中断向量表(IVT)会被放置在0地址开始的空间中，然后根据中断向量值进行正常访问，对应位置存放的是子例程的首地址，跳转执行子例程即可。

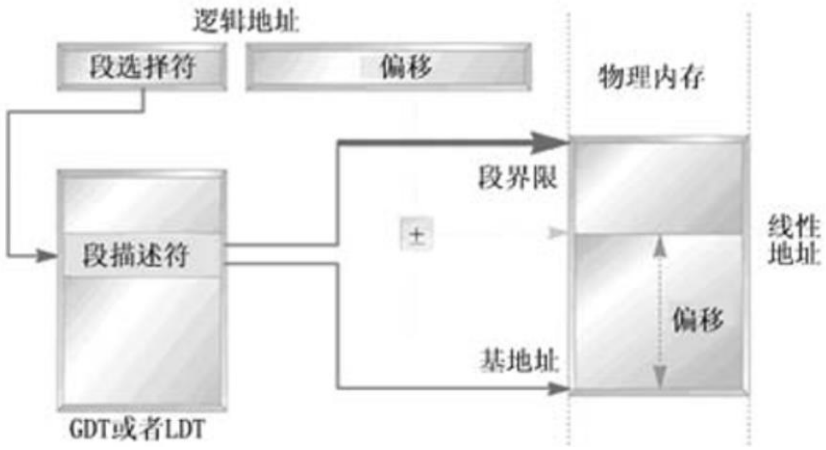
保护模式下，我们使用IDTR（中断向量表寄存器），存放中断描述符表(IDT)的首地址，中断描述符表同样也有256项，除了段描述符和偏移量以外还存放了权限信息，可以存放到内存中的任何地址。

## 问题十五：如何识别键盘组合键(如Shift+A) 是否还有其他解决方案?

定义6个Boolean变量来表示键盘左右shift、alt、ctrl键状态，当被按下时，对应变量变为true，放开弹起时对应变量改为false。如果键（如左或右shift）状态值为true，此时按下另一个键（a），column值变为1，此时则取keymap[column]，即keymap[]中第二列相应的值，此处为A。

## 问题十六：IDT 是什么，有什么作用?

中断描述符表(IDT)是80x86系列中为中断服务提供中断/陷阱描述符，我们引入了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，因此不限制在底部1K地址。




## 问题十七：IDT 中有几种描述符?

1. 任务门描述符
2. 中断门描述符
3. 自陷门描述符

## 问题十八：异常的分类?

1. Fault：
   a) 一种可被更正的异常，而且一旦被更改，程序可以不失连续性的继续执行，返回地址是产生Fault的指令。
   b) 比较常见的是操作系统的系统调用，通常可以被纠正EIP中的保存的是引起故障的指令地址，然后重执指令。

2. Trap：
   a) 一种在发生Trap的指令执行之后立即被报告的异常，它允许程序或任务不失连续性的继续执行，返回地址是Trap指令之后的那条指令。

b) 如果可以修复则无误，如果不能修复则转化为Abort，并进入下一步，比如有缺页。只有没有必要重新执行已终止的指令时才触发陷阱

3. Abort：

a) 不总是报告精确异常发生位置的异常，不允许程序或任务继续执行，而是用来报告严重错误的。

b) 是不可恢复的致命的错误造成结果。终止处理程序不再将控制权交还给引发终止的应用程序，而是交给系统。

**问题十九：用户态和内核态的特权级分别是多少?**

用户态：特权级为3

内核态：特权级为0

**问题二十：中断向量表中，每个中断有几个字节? 里面的结构是什么?**

一个中断向量占据4个字节

低地址两个字节存放偏移

高地址两个字节存放段描述符

**问题二十一：中断异常共同点(至少两点)，不同点(至少三点)**

| 中断 | 异常 |
|---|---|
| **相同点** | |
| 都是程序执行过程中的强制性转移，转移到相应的处理程序。 | |
| 都是软件或硬件发生了某种情形而通知处理器的行为。 | |

| 不同点 | |
|---|---|
| CPU具备的功能，通常因为硬件而随机发生 | 软件运行过程中一种开发过程没有考虑到的程序错误。 |
| CPU暂停处理当前工作，有计划处理其他事情，中断一般是可预知的，处理过程也是事先指定好的，中断时程序是正常运行的 | CPU遇到了无法响应的工作，而后进入一种非正常状态，表明程序有缺陷。 |
| 异步操作：中断是处理器外部的I/O设备的信号的结果，不是指令流中某条指令执行引起的，来自于指令之外。 | 同步操作：执行当前指令流的某条指令的结果上，是来自指令流内部的。 |
| 良性的，在正常的工作流之外执行额外的操作，然后继续执行原指令流的下一条指令，继续执行。 | 有良性和恶性的。良性的，对于可修复的Fault，修复后重执指令。恶性的，如不可修复fault或abort则不会再返回。 |
| 中断是由于当前程序无关的中断信号触发，CPU对中断响应是被动的，并且于CPU模式无关，既可以发生下用户态，又可以发生在核心态。 | 异常是CPU控制单元产生的，主要发生在用户态。 |

添加引脚是困难的，添加内部晶体管是容易的

补充：

处理器状态（处理器模式）可分为*核心态*和*用户态*。

当处理器处于核心态时，CPU运行可信软件，硬件允许执行全部机器指令。

当处理器处于用户态时， CPU运行非可信软件，程序无法执行特权指令，且访问权限仅限于当前CPU上进程的地址空间。