1. 解释中断向量

中断向量可以确定中断服务程序位置，中断向量的地址就是中断服务程序入口地址的地址，它需要4个内存空间存储。

1. 解释中断类型码

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，即中断类型码，用来查找中断向量。

中断指令的一般格式为 “INT n”其中，n称为“中断类型码”，n=0-255；

1. 解释中断向量表

一片专门用来存放所有的中断向量的内存空间。

中断向量表在内存单元的最低处，地址空间为00000H----003FFH(0-1024B)，和中断类型码对应。

1. 实模式下中断程序地址如何得到？

根据中断向量的地址得到中断程序地址。

中断类型号×4=存放中断向量的首地址；

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；

根据中断类型码n，从中断向量表中取得中断服务程序入口地址，取得的段地址存入CS，偏移量存入IP。从而使CPU转入中断服务程序运行。

1. 保护模式下中断程序地址如何得到？

在保护模式下，为每一个中断和异常定义一个中断描述符，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(IDT)取代实地址模式下的中断向量表。

1. 中断向量的地址如何得到？

实模式：中断类型号×4=存放中断向量的首地址；

保护模式：根据中断类型号n，得到中断门在IDT中的起始地址，即：起始地址＝IDT的基地址＋n×8，由中断门中的段选择符，从GDT或LDT中取出段描述符。根据段描述符中提供的段基址和中断描述符中提供的偏移地址，合成中断服务子程序的入口地址。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. 实模式下如何根据中断向量的地址得到中断程序地址？

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；

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。

1. 解释中断描述符

中断描述符分为三类：任务门、中断门和自陷门。

每个中断描述符占用连续的8个字节：

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；

P（present）：表示中断描述符的有效性；

DPL（descriptor privilege level）：特权级；

TYPE：指示中断描述符的不同类型；



1. 保护模式下中断描述符表如何得到？

装载IDTR：

CPU切换到保护模式之前，运行于实模式下的初始化程序使用LIDT指令装载中断描述符表IDT，将IDT基地址与段界值装入IDTR。

1. 保护模式下中断门如何得到？

查IDT：

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。

1. 保护模式下如何根据中断门得到中断处理程序地址？

1、装载IDTR

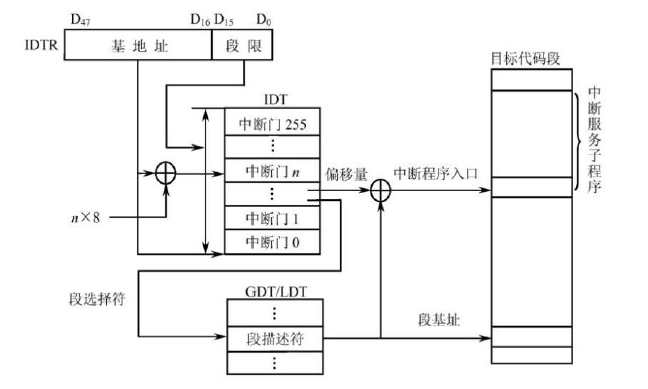
CPU切换到保护模式之前，运行于实模式下的初始化程序使用LIDT指令装载中断描述符表IDT，将IDT基地址与段界值装入IDTR。

2、查IDT

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和偏移量得到中断处理程序入口



根据段寄存器的内容(选择子)，首先判断描述符是在GDT中还是在LDT中，如果是在GDT中，根据GDTR以及该段寄存器的内容找到相应的“描述符”；如果是在LDT中，根据LDTR（选择子）以及GDTR的内容找到LDT的描述符，得到LDT的地址，然后再根据段寄存器内容找到相应的“描述符”

4、从描述符中得到基地址；

5、将指令中发出的地址作为位移，与描述符中界限相比，看是否越界；将指令的性质与描述符中的访问权限来确定是否越权；

6、将指令中发出的地址作为位移，与基地址相加得出实际的“物理地址”。

1. 中断的分类，举例不同类型的中断？

* 从中断源的角度分类

➀ 由计算机硬件异常或故障引起的中断，也称为**内部异常中断**。

➁ 由程序中执行了中断指令引起的中断，也称为**软中断**。由程序员通过INT或INT3指令触发，通常当做trap处理，用处：实现系统调用。

➂ 外部设备（如输入输出设备）请求引起的中断，也称为**外部中断或Ｉ／Ｏ中断**。

* 从CPU内外的角度分类

➀ 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。

➁ 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。

* 外部中断的分类

➀ 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。

➁ 不可屏蔽中断：重新启动、电源故障、内存出错、总线出错等影响整个系统工作的中断是不能屏蔽的。

13. 中断与异常的区别？

* ① 中断，是CPU所具备的功能。通常因为“硬件”而随机发生。

异常，是“软件”运行过程中的一种开发过程中没有考虑到的程序错误。

* ② 中断是CPU暂停当前工作，有计划地去处理其他的事情。中断的发生一般是可以预知的，处理的过程也是事先制定好的。处理中断时程序是正常运行的。

异常是CPU遇到了无法响应的工作，而后进入一种非正常状态。异常的出现表明程序有缺陷。

* ③ 中断是异步的，异常是同步的。

中断是来自处理器外部的I/O设备的信号的结果，它不是由指令流中某条指令执行引起的，从这个意义上讲，它是异步的，是来自指令流之外的。

异常是执行当前指令流中的某条指令的结果，是来自指令流内部的，从这个意义上讲它们都是同步的。

* ④ 中断或异常的返回点

良性的如中断和trap，只是在正常的工作流之外执行额外的操作，然后继续干没干完的活。因此处理程序完了后返回到原指令流的下一条指令，继续执行。

恶性的如fault和abort，对于可修复fault，由于是在上一条指令执行过程中发生（是由正在执行的指令引发的）的，在修复fault之后，会重新执行该指令；至于不可修复fault或abort，则不会再返回。

* ⑤ 中断是由于当前程序无关的中断信号触发的，CPU对中断的响应是被动的，且与CPU模式无关。既可以发生在用户态，又可以发生在核心态。

异常是由CPU控制单元产生的，大部分异常发生在用户态。

14. 实模式和保护模式下的中断处理差别

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。

15. 如何识别键盘组合键（如 Shift+a）。是否还有其他解决方案？

声明1个变量make来记录是Make code还是Break code（按下那一刻产生一个通码Make code,释放的时候产生一个断码Break code）//int make; /\* 1: make; 0: break. \*/

声明2个变量shift\_l，shift\_r来记录shift键的状态。

用一个局部变量caps，记录shift键是否被按下。//int caps = shift\_l || shift\_r;

当其中的某一个键被按下时，相应的make值变为1，如果它立即被释放，则make值变回0。如果shift键被按下且未被释放时，又按下a键，则if(shift\_l|shift\_r)成立，于是column值变为1，keymap[column]的取值就是keymap[]中第二列中相应的值，即大写字母A。

16. IDT 是什么，有什么作用？

在保护模式下，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），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；

IDT的作用是将每一个中断向量和一个描述符对应起来，并引入一个48位的全地址寄存器存放IDT的内存地址。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）。

17. IDT 中有几种描述符？

三种：任务门、中断门和自陷门。

18. 异常的分类？

* Fault，是一种可被更正的异常，而且一旦被更正，程序可以不失连续性地继续执行。返回地址是产生fault的指令。
* Trap，一种在发生trap的指令执行之后立即被报告的异常，它也允许程序或任务不失连续性地继续执行。返回地址是产生fault的指令之后的那条指令。
* Abort，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。

19. 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）

20. 中断向量表中，每个中断有几个字节？里面的结构是什么？

中断向量表起始地址为0，包含256个中断向量，每一个中断向量包含四字节，以低位二字节存储偏移量，高位二字节存储段地址。

21. 中断异常共同点 (至少两点)，不同点 (至少三点)

共同点：

* 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。
* 都是软件或者硬件发生了某种情形而通知处理器的行为

不同点：

* ① 中断，是CPU所具备的功能。通常因为“硬件”而随机发生。

异常，是“软件”运行过程中的一种开发过程中没有考虑到的程序错误。

* ② 中断是CPU暂停当前工作，有计划地去处理其他的事情。中断的发生一般是可以预知的，处理的过程也是事先制定好的。处理中断时程序是正常运行的。

异常是CPU遇到了无法响应的工作，而后进入一种非正常状态。异常的出现表明程序有缺陷。

* ③ 中断是异步的，异常是同步的。

中断是来自处理器外部的I/O设备的信号的结果，它不是由指令流中某条指令执行引起的，从这个意义上讲，它是异步的，是来自指令流之外的。

异常是执行当前指令流中的某条指令的结果，是来自指令流内部的，从这个意义上讲它们都是同步的。

* ④ 中断或异常的返回点

良性的如中断和trap，只是在正常的工作流之外执行额外的操作，然后继续干没干完的活。因此处理程序完了后返回到原指令流的下一条指令，继续执行。

恶性的如fault和abort，对于可修复fault，由于是在上一条指令执行过程中发生（是由正在执行的指令引发的）的，在修复fault之后，会重新执行该指令；至于不可修复fault或abort，则不会再返回。

* ⑤ 中断是由于当前程序无关的中断信号触发的，CPU对中断的响应是被动的，且与CPU模式无关。既可以发生在用户态，又可以发生在核心态。

异常是由CPU控制单元产生的，大部分异常发生在用户态。