|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 操作系统设计与实践 | | | 成 绩 |  | 教师签名 |  |
| 实验名称 | 中断与异常 | | | 实验序号 | 4 | 实验日期 |  |
| 姓 名 | 张子航 | 学 号 | 2021302181026 | | | | 组长 |
| 姓 名 | 辜汝曦 | 学 号 | 2021302141194 | | | | 组员 |
| 姓 名 | 杨馨悦 | 学 号 | 2021302181212 | | | | 组员 |
| 姓 名 | 赵敏 | 学 号 | 2021302181215 | | | | 组员 |

**《操作系统设计与实践》实验报告**

# 实验目的及实验内容

（本次实验所涉及并要求掌握的知识；实验内容等）

## 实验目的

理解中断与异常机制的实现机理

## 参考资料

《Orange’S 一个操作系统的实现》第三章3.4节

## 实验要求

* 理解中断与异常的机制
* 调试8259A的编程基本例程
* 调试时钟中断例程
* 实现一个自定义的中断向量，功能可自由设想
* 完成本次实验要回答的问题

# 实验环境及实验步骤

（列出本次实验所使用的软件、工具；简要概括实验步骤）

## 实验环境

虚拟机工具：VMWare Workstation 16

虚拟机版本：Ubuntu 14.04.6(内存4GB，硬盘40GB，双核处理器)

开发与调试工具：bochs 2.6.8

## 实验步骤

1.理解中断与异常的机制

2.调试8259A的编程基本例程，了解8259A的工作原理

3.调试时钟中断例程，了解如何控制时钟中断

4.实现一个自定义的中断，建立IDT

# 实验过程分析

（详细记录实验过程，通过截图展示得到的结果。特别是对于实验中发生的故障和问题，要进行故障分析，说明故障排除的过程及方法。）

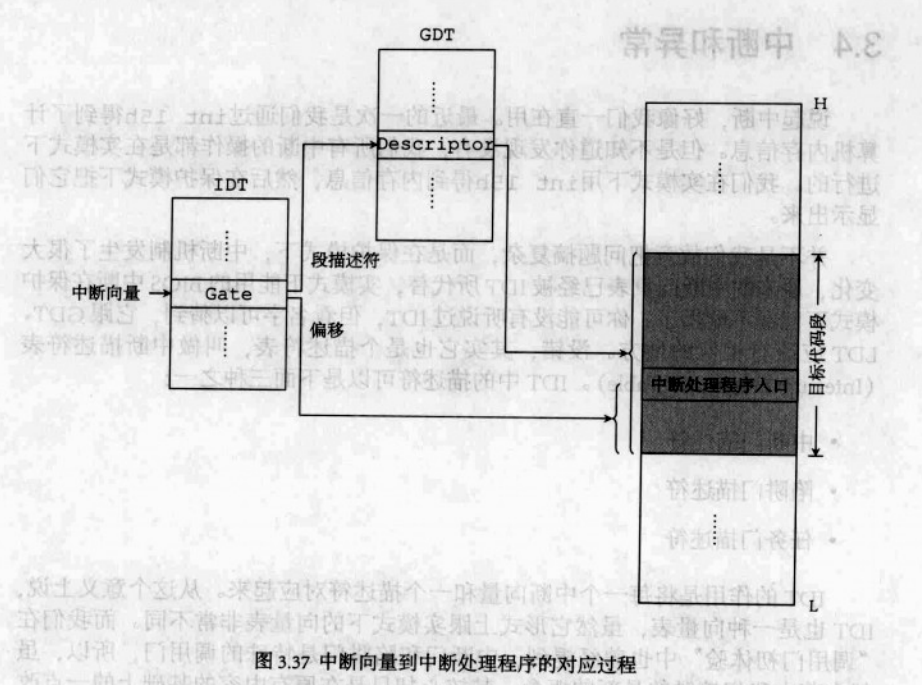
## 完成本次实验内容

### 理解中断与异常的机制

中断与异常都是软件或硬件发生了某种情形而通知处理器的行为。每一种中断（异常）都会对应一个中断向量号，而这个向量号通过IDT与相应的中断处理程序相对应起来实现处理异常的能力。

中断产生的原因有两种：

由指令int n产生的中断



由硬件产生的中断，也就是外部中断

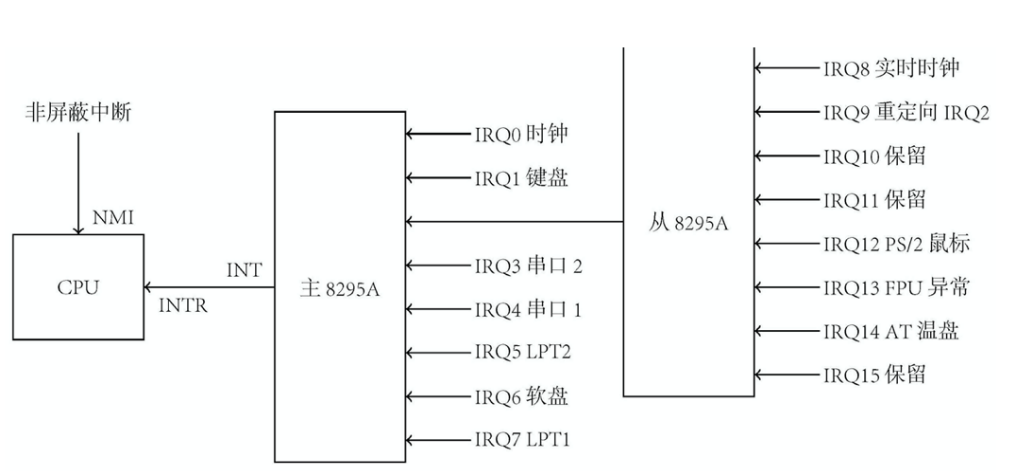
异常有fault、trap和abort三种类型：

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。当一个fault发生时，处理器会把产生fault的指令之前的状态保存起来。异常处理程序的返回地址将会是产成fault的指令，而不是其后的那条指令。

Trap是一种在发生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。异常处理程序的返回地址将会是产成 trap 的指令之后的那条指令

Abort是一种不总是报告精确异常发生位置的异常，它不允许程序或任务继续执行，而是用来报告严重错误的。

### 2.调试8259A的编程基本例程



我们使用主8259A和从8259A来实现可屏蔽中断INT。为此，我们需要在操作开始前由CPU向8259A输送初始化命令字ICW1-4来初始化8259A。具体来说，CPU需要：

* 往端口20h(主片)或A0h(从片)写入ICW1；
* 往端口21h(主片)或A1h(从片)写入ICW2
* 往端口21h(主片)或A1h(从片)写入ICW3
* 往端口21h(主片)或A1h(从片)写入ICW4

下面的语句用于向端口写入命令字ICW1：

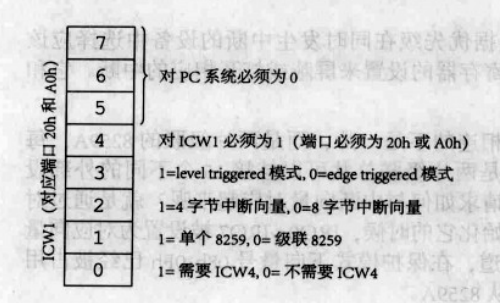
**mov al, 011h**

**out 020h, al ; 主8259, ICW1.**

**call io\_delay**

**out 0A0h, al ; 从8259, ICW1.**

上面的语句向端口写入0001 0001。本实验的系统为8086系统，而8086系统不使用D7-D5，D3-D1这些位，因此置0；D4表示标志位，取1；D0表示需要设ICW4，取1。



下面的语句用于向端口写入命令字ICW2：

**mov al, 020h ; IRQ0 对应中断向量 0x20**

**out 021h, al ; 主8259, ICW2.**

**call io\_delay**

**mov al, 028h ; IRQ8 对应中断向量 0x28**

**out 0A1h, al ; 从8259, ICW2.**

**call io\_delay**

上面的语句向主片端口写入0010 0000。D7-D3这些位可以确定中断类型码N的高5位，为2；D2-D0则是由8259A基于IRi自动填入，其初值可以置为000。D2-D0的取值000-111对应i的值：比如，键盘中断对应IR1，因此键盘调用为0x21；时钟调用对应IR0，因此时钟调用为0x20。



下面的语句用于向端口写入命令字ICW3：

**mov al, 004h ; IR2 对应从8259**

**out 021h, al ; 主8259, ICW3.**

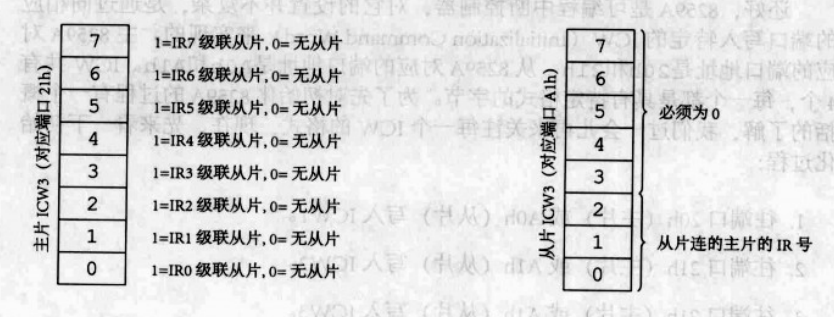
**call io\_delay**

**mov al, 002h ; 对应主8259的 IR2**

**out 0A1h, al ; 从8259, ICW3.**

**call io\_delay**

上面的语句向主片端口写入0000 0100，从片端口写入0000 0010。ICW3用于级联，主8259A上，Di为0则表示对应的IRi没有接从片，Di为1则表示对应的IRi接有从片；从8259A上，低3位表示从片接到主片的哪一个引脚，其数字表示000-111代表IR0到IR7。从上面的语句中，可以看出从片连接到主片的IR2上。



下面的语句用于向端口写入命令字ICW4：

**mov al, 001h**

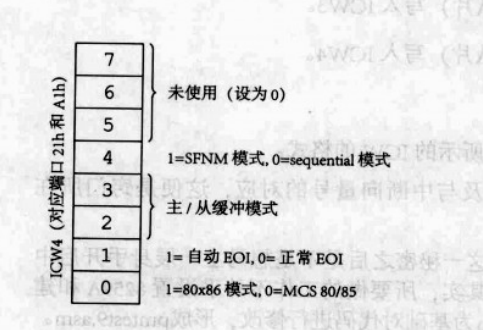
**out 021h, al ; 主8259, ICW4.**

**call io\_delay**

**out 0A1h, al ; 从8259, ICW4.**

**call io\_delay**

上面的语句向主片和从片的端口写入0000 0001，结合ICW4字段的意义，可以得知，代码使用非缓冲方式、全嵌套方式、非自动结束中断与8086配置。



总体来说，本实验初始化8259A的流程如下：

* 首先写入ICW1；
* 写入ICW2，确定中断类型码的高5位；
* ICW1选择级联方式，接着写入ICW3，确定主片和从片的连接关系；
* 写ICW4，确定中断结束方式、中断嵌套方式等属性；
* 当这些步骤做完，8259A的初始化完成，写入操作命令字OCW1。

下面的语句用于写入操作命令字OCW1：

**;mov al, 11111111b ; 屏蔽主8259所有中断**

**mov al, 11111110b ; 仅仅开启定时器中断**

**out 021h, al ; 主8259, OCW1.**

**call io\_delay**

**mov al, 11111111b ; 屏蔽从8259所有中断**

**out 0A1h, al ; 从8259, OCW1.**

**call io\_delay**

它直接操作IMR，当IMR的Mi位为0时，表示允许来自IRi的中断；当Mi位为1时，表示屏蔽来自IRi的中断。1111 1110表示开启定时器中断；而在之后编写自定义中断的实验中，1111 1101表示开启键盘中断。



### 3.调试时钟中断例程

1. 时钟中断处理程序

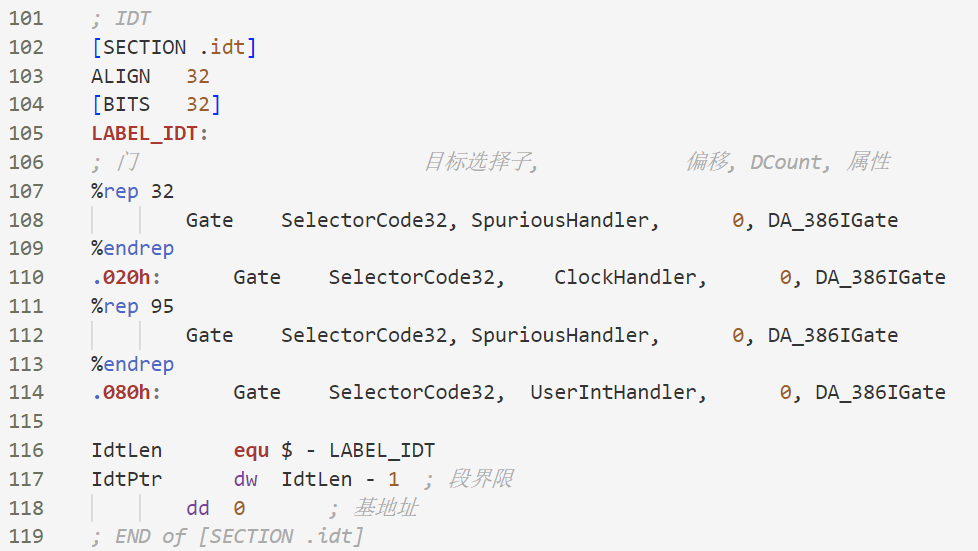


功能：发送EOI的两行语句以及iretd，把屏幕第0行、第70列的字符增一，变成ASCII 码表中位于它后面的字符。如果我们在调用80h号中断之后打开中断的话，由于第0行、第70列处已被写入字符1，所以第一次中断发生时那里会变成字符J，再一次中断则变成K，以后每发生一次时钟中断，字符就会变化一次，就会看到不断变化中的字符。

1. 打开时钟中断

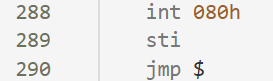


1. 添加IDT选择子

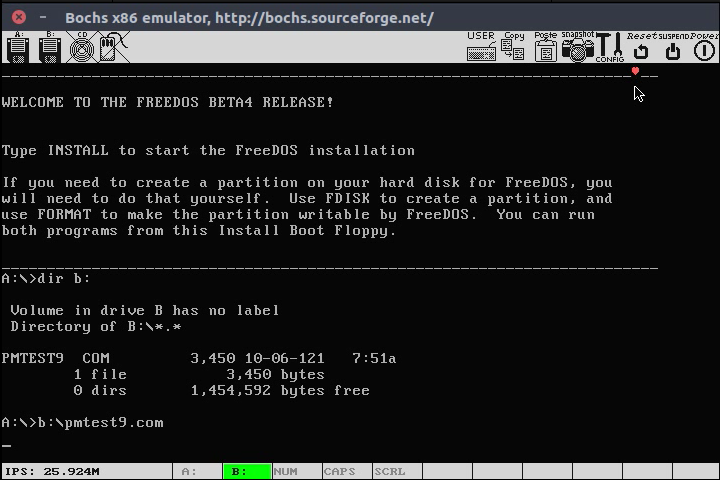


1. 调试结果

在调用80h号中断之后执行sti来打开终端



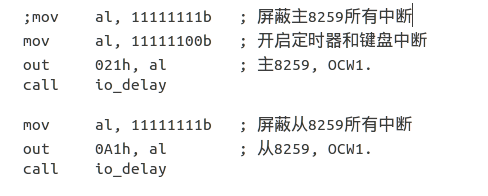
pmtest9.com运行结果，程序右上角成功输出连续的不同的红色字符，证实时钟中断成功（ pmtest9.asm静态截图）



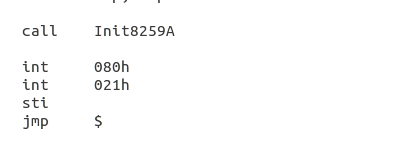
### 4.实现一个自定义的中断向量，功能可自由设想。

在pmtest9.asm的基础上添加一个键盘中断，每次接收到键盘输入，修改字符的颜色

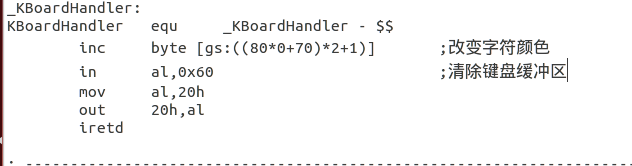
1. 开启键盘中断：



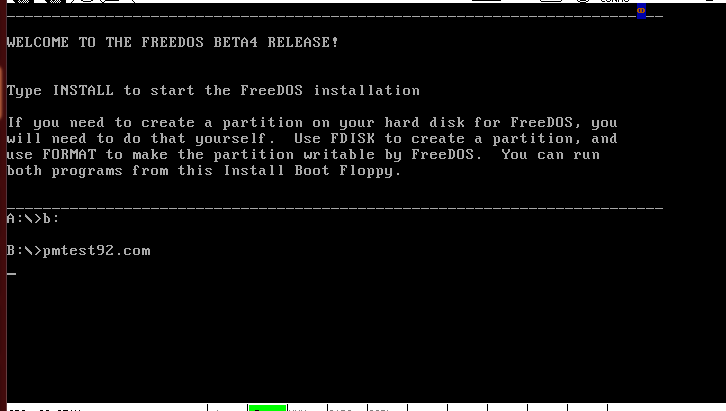
1. 添加调用中断21h

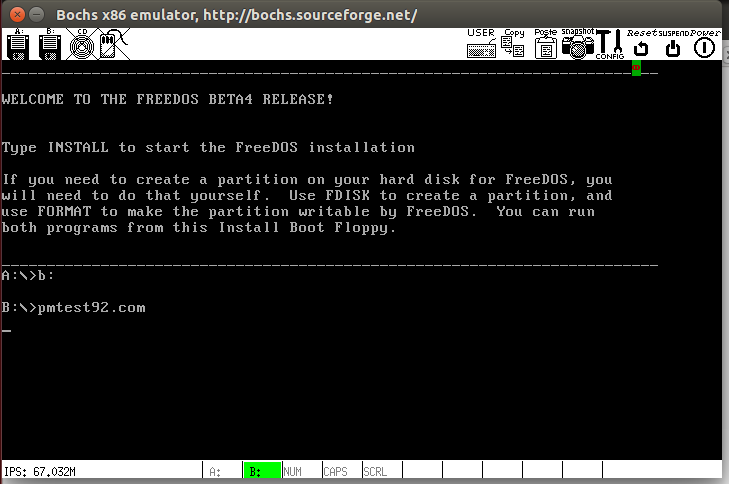


1. 编写键盘中断函数



1. 运行效果，每次按下键盘，正在变化的字符颜色将会发生改变







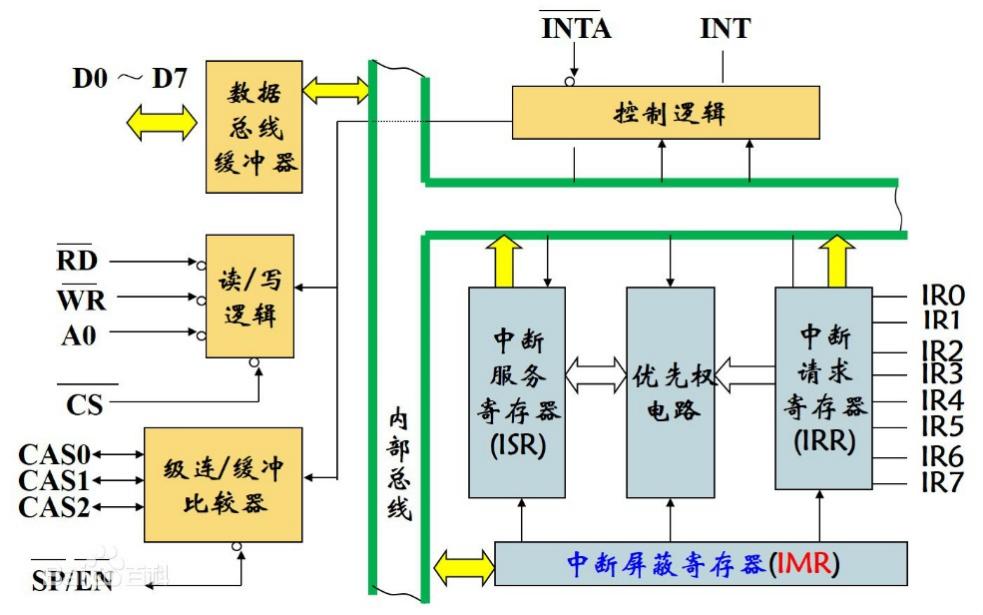
## 完成本次实验要回答的问题

### 什么是中断，什么是异常

**中断**通常指在程序执行时因为硬件而随机发生，他们通常用来处理处理器内部的事件，比如外围设备的请求，也就是由硬件产生的中断。软件也可通过int指令产生

**异常**则通常在处理器执行指令过程中检测到错误时产生，比如遇到零除、保护违例、页错误等情况

### 8259A的工作原理是怎样的？怎么给这些中断号的处理向量初始化值？



在前文中已经说到，可以使用8259A来实现可屏蔽中断。具体来说，CPU通过初始化命令寄存器ICW1-ICW4与操作命令寄存器OCW1-OCW3来管理8259A。命令字段的写入通过读写逻辑模块的A0-WR共同完成，读出通过读写逻辑模块的A0-RD共同完成，数据的传输通过数据总线缓冲器的D0-D7完成。

8259A还具有以下几种8位寄存器：

* IRR，中断请求状态寄存器，用来标记到达的中断请求。
* ISR，中断服务状态寄存器，用来记录被处理器处理的中断请求。
* IMR，中断屏蔽状态寄存器，用来记录屏蔽的中断请求。

下面，假设8259A的各种设置均已完成，我们再来看以下其工作流程：

* 一个外部中断请求信号通过中断请求线IRQ，传输到IMR，IMR根据所设定的中断屏蔽字OCW1，决定是屏蔽还是处理该中断请求。如果OCW1对应的值为0，即可以处理，则8259A将IRR中代表此IRQ的位置置1，以表示此IRQ有中断请求信号、
* 同时，8259A向CPU的INTR管脚发送信号，表示此时存在中断请求。
* 通常情况下，CPU正在执行指令，而不会立即响应此中断请求；同时，也可能会有更多来自IRQ的新的中断请求。经过IMR筛选后，不被屏蔽的请求也会被储存到IRR的其他对应位置中。
* 当CPU执行完指令后，会检查INTR管脚是否有信号。如果发现有信号，就会转到中断服务。此时，CPU会向8259A芯片的INTA管脚发送一个信号。
* 当芯片收到此信号后，8259A会在IRR中挑选优先级最高的中断，将中断请求送到ISR，将ISR中对应此IRQ的位置置1，并将IRR中相应位置置0，表明此中断正在接受CPU的处理。
* 同时，将此中断的编号写入IVR（中断向量寄存器）的低三位。IVR的低三位对应8259A初始化时全部置0的ICW2低三位。
* 此时，CPU还会送来第二个INTA信号，当收到此信号后，8259A将IVR中的内容，也就是此中断的中断号送到CPU。

上面就是8259A的工作原理。给中断号的处理向量初始化值的过程，在实验内容的对应部分已经有比较详细的解答，可以参考。简单来说，初始化具有以下流程：

* 首先写入ICW1；
* 写入ICW2，确定中断类型码的高5位；
* ICW1选择级联方式，接着写入ICW3，确定主片和从片的连接关系；
* 写ICW4，确定中断结束方式、中断嵌套方式等属性；
* 当这些步骤做完，8259A的初始化完成，写入操作命令字OCW1。

根据具体的实验需求与系统类型（是否为8086系统）不同，其中的细节也会不一样。

### 如何建立IDT，如何实现一个自定义的中断

1. 建立一个IDT（中断描述符表），可以按照以下步骤进行：

* 创建一个包含中断描述符的数组，数组的每个元素都是一个中断描述符的结构体。中断描述符包含了中断处理程序的地址、段选择子和一些标志位等信息。
* 初始化每个中断描述符，将中断处理程序的地址和段选择子等信息填入到对应的中断描述符中。
* 将中断描述符表的地址加载到IDTR（中断描述符表寄存器）中，IDTR是一个特殊的寄存器，用于存储中断描述符表的地址。
* 启用中断，通过设置处理器的中断使能位，使得处理器能够响应中断。

1. 要实现一个自定义的中断，可以按照以下步骤进行：

* 编写一个中断处理程序，中断处理程序是一个特殊的函数，用于处理特定的中断事件。中断处理程序通常会保存当前的上下文，执行特定的处理逻辑，然后恢复之前的上下文并返回。
* 将中断处理程序的地址填入到对应的中断描述符中，可以通过修改IDT来实现。
* 启用中断，通过设置处理器的中断使能位，使得处理器能够响应自定义的中断。
* 需要注意的是，在实现自定义中断时，需要考虑中断处理程序的执行环境和上下文保存恢复等细节，以确保中断处理程序能够正确地处理中断事件并返回到原来的执行流程中。

### 如何控制时钟中断，为什么时钟中断时候，没有看到int的指令？

时钟和CPU是独立的，时钟中断程序时这样的：在CPU执行指令时，计数器（计时器）同时在计时，当计时器溢出时，就向CPU申请中断，如果允许响应中断，CPU就转到中断服务程序执行相关的程序。

单片机的时钟中断相应要满足两个条件：第一，允许中断源申请中断。第二，允许cpu响应中断。二者缺一不可。

当我们设置了允许时钟中断时，程序跑起来时，时钟时间到了，CPU就转入中断服务程序了，执行完了再返回主程序。因此时钟中断不需要额外添加int指令。

# 实验结果总结

（对实验结果进行分析。并理论联系实际，思考并列出本实验对应的OS原理的知识点，并说明本实验中的实现部分如何对应和体现了原理中的基本概念和关键知识点。）

1. **中断机制**

中断信号由外部设备发起，准确来说是由外部设备的控制器发起，因为外部设备本身并不能发起信号。必须网卡设备，的那个网络数据包到达网卡，网卡的控制器就向IO APIC发送中断信号，IO APIC把信号发送给本地APIC，本地APIC把信号传送给CPU，如果根据当时情况，要处理这个中断，就保存当时的运行上下文，切换到中断上下文中，根据IDT查找对应的处理函数进行处理。处理完成后，需要恢复中断之前的状态。

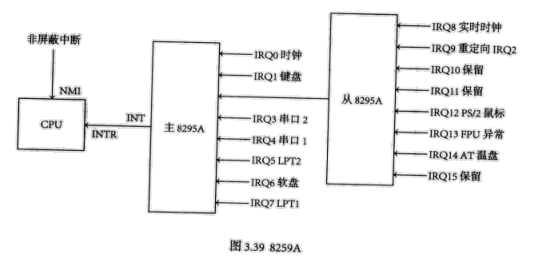
关键几点如下：

1、设备控制器发送中断信号

2、APIC接受中断信号以及相应处理

3、处理器接受中断信号以及相应处理

可屏蔽中断与CPU的关系是通过对可编程中断控制器8259A建立起来的。8259A可以认为是中断机制中所有外围设备的一个代理，这个代理不但可以根据优先级在同时发生中断的设备中选择应该处理的请求，而且可以通过对其寄存器的设置来屏蔽或打开相应的中断。它和CPU的连接下图所示。



由上图我们知道，与CPU相连的不是一片，而是两片级联的8259A，每个8259A有8根中断信号线，于是两片级联总共可以挂接15个不同的外部设备。这些设备发出的中断请求是通过对8259A的设置来完成与中断向量对应的。在BIOS初始化它的时候，IRQ0~IRQ7被设置为对应向量号08h~OFh，而在保护模式下向量号08h-OFh已经被占用了，所以我们不得不重新设置主从8259A。

还好，8259A是可编程中断控制器，对它的设置并不复杂，是通过向相应的端口写入特定的ICW (Initialization Command Word)来实现的。主8259A对应的端口地址是20h和21h，从8259A对应的端口地址是Aoh和A1h。ICw共有4个，每一个都是具有特定格式的字节。为了先对初始化8259A的过程有一个概括的了解，初始化过程如下:

1. 往端口20h(主片)或A0h(从片)写入ICW1。
2. 往端口21h(主片)或A1h(从片)写入ICW2。
3. 往端口21h(主片)或A1h(从片)写入ICW3。
4. 往端口21h(主片)或A1h(从片)写入ICW4。

这4步的顺序是不能颠倒的。

ICW1负责启动8259A和进行初始化工作

ICW2中断类型号的设置

ICW3主从片初始化设置

ICW4方式控制设置

想要屏蔽或打开外部中断，只要往8259A中对应的OCW中写入1或0即可。

# 个人分工及心得体会

（每个人分别填写自己在本次实验中的分工，并总结实验的心得体会。）

## 张子航

完成而不能此实验内容的第1题、本次实验要回答的第1题，并撰写实验报告中的对应部分

通过本次实验，我深刻意识到了中断与异常的重要性，也意识到中断与异常的机制作为计算机系统中的一种高效处理外部事件和错误的方式，通过理解这些机制，可以帮助我们更好地设计和优化系统，提高操作系统的可靠性和性能。在调试中断服务例程的实验过程中，我不仅学到了具体的技术细节，还锻炼我解决问题和优化代码的能力，也加深了我对操作系统的认识

## 辜汝曦

完成本次实验内容的第2题、本次实验要回答的问题的第2题，并撰写实验报告中的对应部分、和实验目的及实验内容部分。除此之外，也独立完成了本次实验内容的第4题，即实现一个自定义的中断向量。

通过本次实验，我对中断和异常有了更深的理解。以前在汇编语言与计算机组成原理课程上，我们对中断只需要了解int指令就行了；而在本次实验中，我认识了中断请求是如何被8259A处理、进而被CPU处理的，了解了IMR、IRR、ISR等寄存器在这一过程中都起到了什么作用。我还掌握了8259A的初始化与操作方法，主要包括对某种类型的中断进行屏蔽与取消屏蔽的操作。这在我之后编写自定义中断向量中起到了很大作用。

## 杨馨悦

独立完成本次实验，并撰写实验报告中本次实验内容的第3题、本次实验要回答的问题的第3题，以及实验结果总结的中断机制部分。

通过本次实验，了解了IDT和中断异常机制的相关概念和具体工作方式，了解了建立IDT、实现自定义中断的基本流程，通过对各实验结果输出与代码阅读分析加深了对操作系统中断异常机制知识细节的了解，并且通过实操，添加键盘中断处理函数实现不同输出，加深了对中断机制的理解，为进一步搭建简易操作系统打下了良好基础，也激发了进一步学习探索的热情。

## 赵敏

独立完成本次实验，并撰写实验报告中本次实验内容的第4题、本次实验要回答的问题的第4题.

通过本次实验，对中断和异常处理程序添加了新的理解，了解了IDT和中断异常机制的相关概念和具体工作方式，在编写自定义的中断程序的过程中，熟悉了添加中断程序的操作步骤，仔细探究了8259A初始化各个步骤中初始化字段的含义，加深了对时钟中断和键盘中断的实现步骤的印象，不同中断的实现也激发了我的学习热情，期待进一步学习不同的知识。