异步事件编程技术

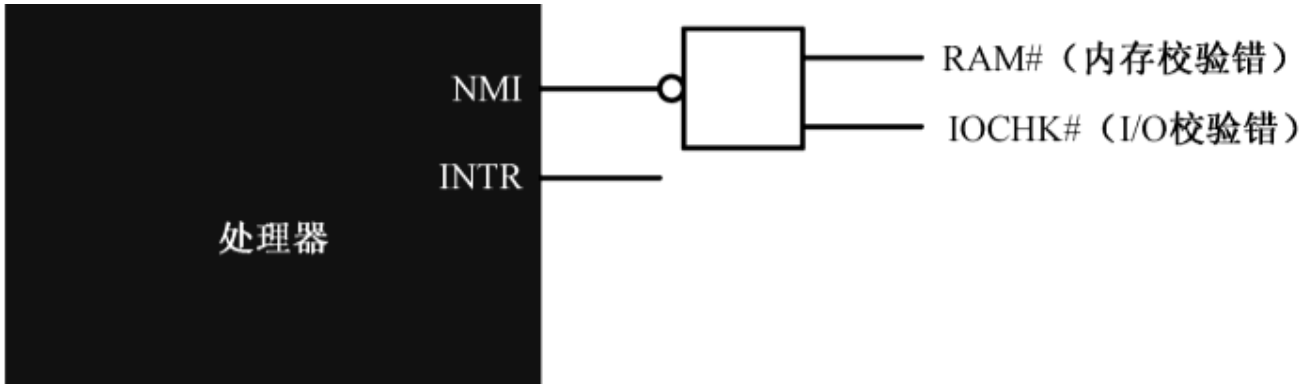
- 学习中断机制知识，掌握中断处理程序设计的要求
- 设计一个汇编程序，实现时钟中断处理程序
- 扩展MyOS2，增加时钟中断服务，利用时钟中断实现与时间有关的操作
- 实验项目4说明

异步事件

- 在操作系统世界里，许多活动或事件可能并发进行，随时可能发生或结束，不可预测。
  - 在计算机硬件系统上，硬件系统的各个工作部件之间也可以并行工作，如CPU与I/O设备可以并行工作、不同I/O设备之间也可并行工作。
  - 硬件系统的并发活动提高了计算机系统的效率，但这些活动必须由操作系统进行有效的管理。
  - 计算机硬件系统提供中断技术，支持CPU与外部设备的并发工作，也利用中断技术处理硬件错误、支持程序调试、实现软件保护和信息安全等。

中断技术

- 中断(interrupt)是指对处理器正常处理过程的打断。中断与异常一样，都是在程序执行过程中的强制性转移，转移到相应的处理程序。
  - 硬中断（外部中断）——由外部（主要是外设[即I/O设备]）的请求引起的中断
    - 时钟中断（计时器产生，等间隔执行特定功能）
    - I/O中断（I/O控制器产生，通知操作完成或错误条件）
    - 硬件故障中断（故障产生，如掉电或内存奇偶校验错误）
  - 软中断（内部中断）——由指令的执行引起的中断
    - 中断指令（软中断int n、溢出中断into、中断返回iret、单步中断TF=1）
  - 异常/程序中断（指令执行结果产生，如溢出、除0、非法指令、越界）



PC中断系统

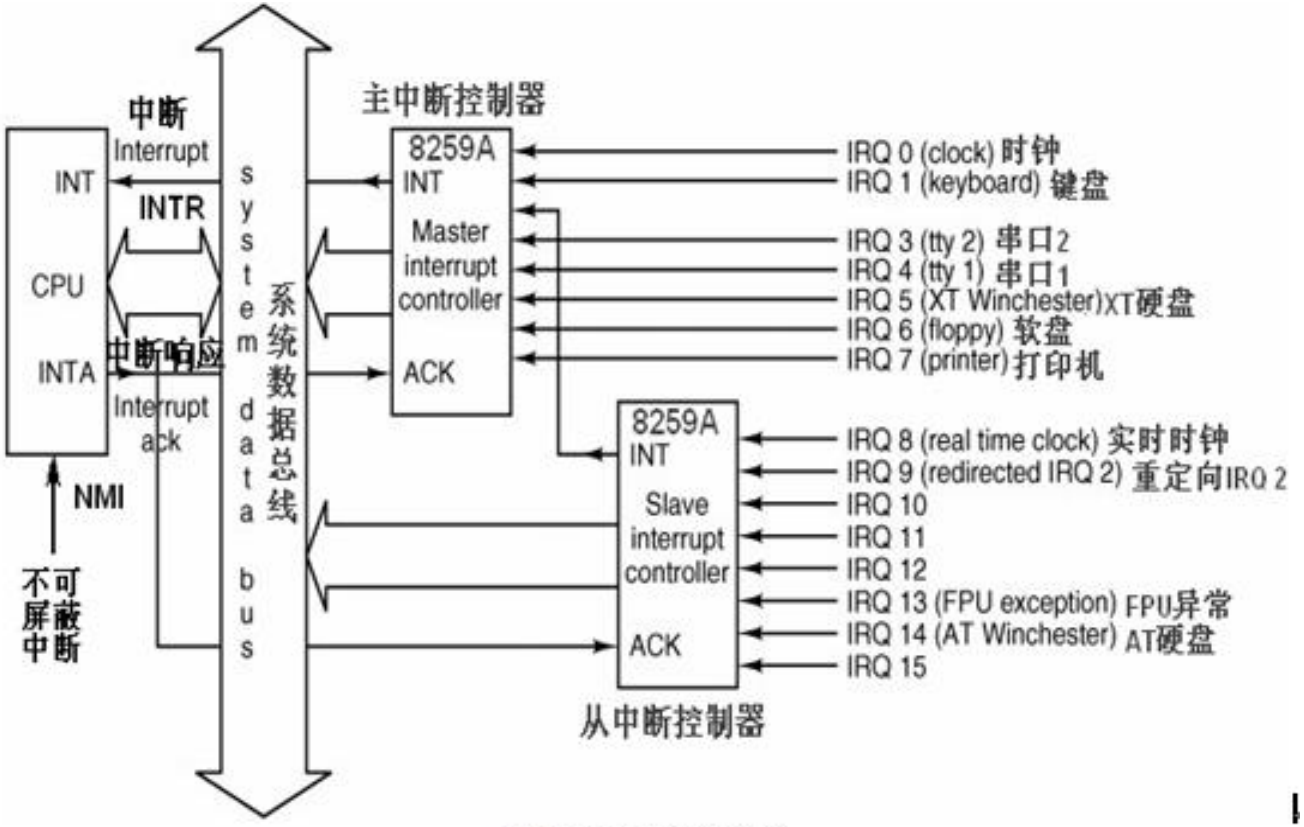
- x86 PC机的中断系统的功能强大、结构简单、使用灵活。采用32位的中断向量（中断处理程序的映射地址），可处理256种不同类型的中断。
- x86处理器有两条外部中断请求线
  - NMI（Non Maskable Interrupt，不可屏蔽中断）
  - INTR（Interrupt Request，中断请求[可屏蔽中断]）
  - CPU是否响应在INTR线上出现的中断请求，取决于标志寄存器FLAGS中的IF标志位的状态值是否为1。可用机器指令STI/CLI置IF标志位为1/0来开/关中断。
  - 在系统复位后，会置IF=0（中断响应被关闭）。在任意一中断被响应后，也会置IF=0（关中断）。若想允许中断嵌套，必须在中断处理程序中，用STI指令来打开中断
  - 在NMI线上的中断请求，不受标志位IF的影响。CPU在执行完当前指令后，会立即响应。不可屏蔽中断的优先级要高于可屏蔽中断的。

PC中断的处理过程

- 保护断点的现场
  - 要将标志寄存器FLAGS压栈，然后清除它的IF位和TF位
  - 再将当前的代码段寄存器CS和指令指针寄存器IP压栈
- 执行中断处理程序
  - 由于处理器已经拿到了中断号，它将该号码乘以4（毕竟每个中断在中断向量表中占4字节），就得到了该中断入口点在中断向量表中的偏移地址
  - 从表中依次取出中断程序的偏移地址和段地址，并分别传送到IP和CS，自然地，处理器就开始执行中断处理程序了。
  - 注意，由于IF标志被清除，在中断处理过程中，处理器将不再响应硬件中断。如果希望更高优先级的中断嵌套，可以在编写中断处理程序时，适时用sti指令开放中断。
- 返回到断点接着执行
  - 所有中断处理程序的最后一条指令必须是中断返回指令iret。这将导致处理器依次从堆栈中弹出（恢复）IP、CS和FLAGS的原始内容，于是转到主程序接着执行。

两个级联的8259A芯片

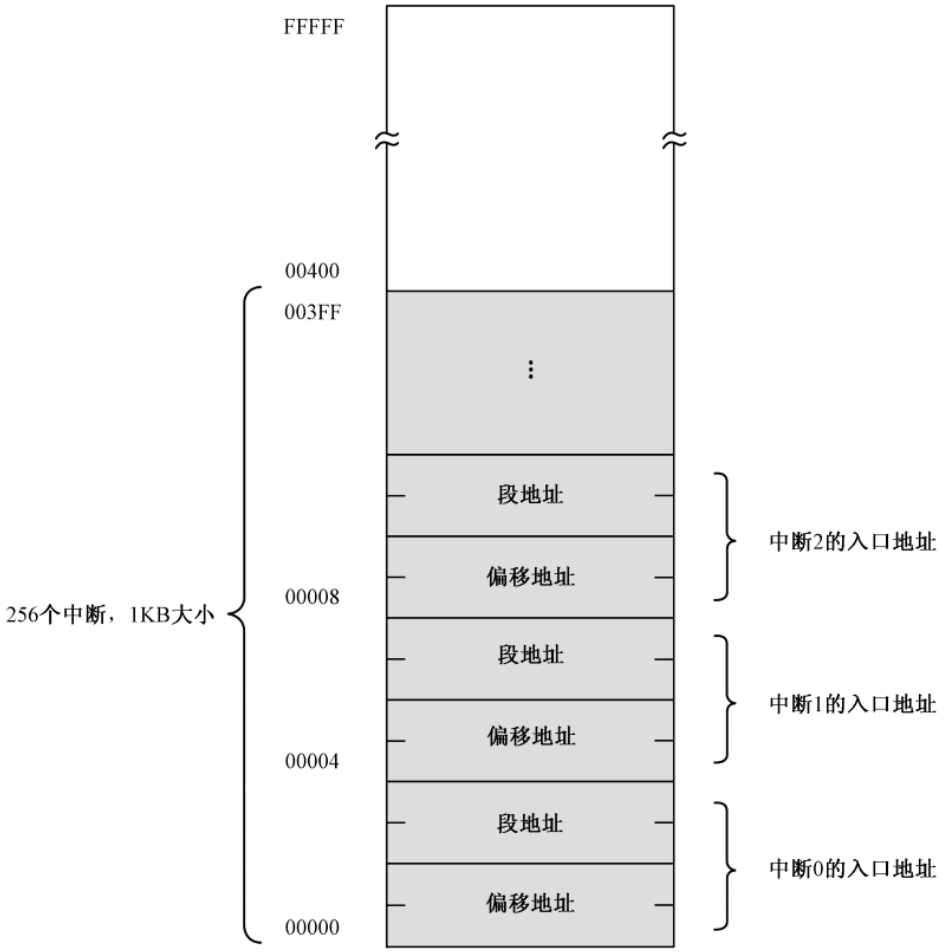
- NMI和INTR两条中断线是远不能满足PC需求的
- x86处理器用两个级联的8259A芯片作为外设向CPU申请中断的代理接口，使一条INTR线扩展成15条中断请求线。



PC 的中断处理硬件

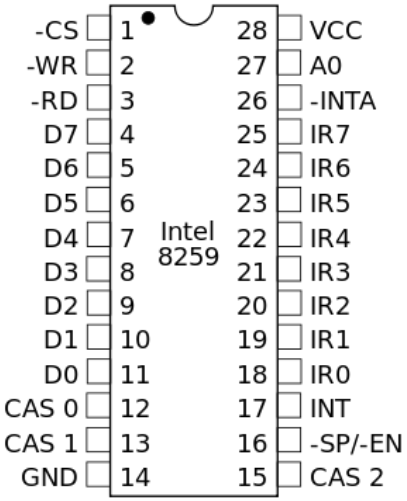
中断向量

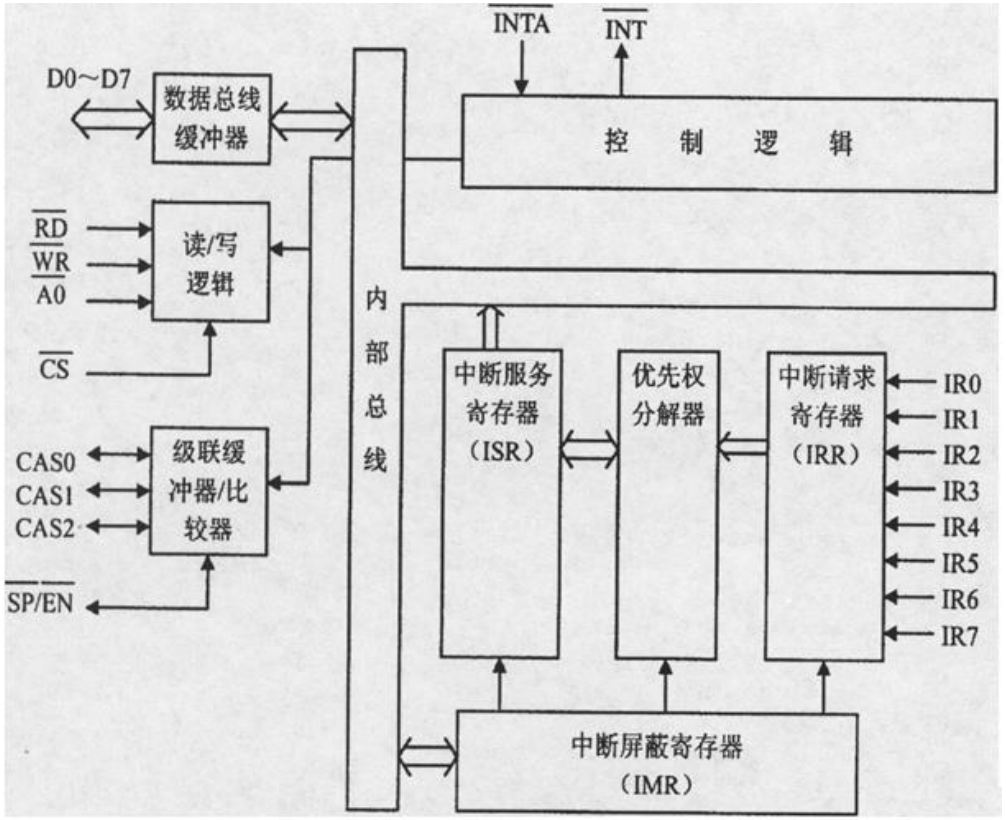
- x86计算机在启动时会自动进入实模式状态
  - 系统的BIOS初始化8259A的各中断线的类型（参见前图），
  - 在内存的低位区（地址为0~1023[3FFH]，1KB）创建含256个中断向量的表IVT （每个向量[地址]占4个字节，格式为：16位段值:16位偏移值）。
- 当系统进入保护模式
  - IVT（Interrupt Vector Table，中断向量表）会失效
  - 需改用IDT（Interrupt Descriptor Table，中断描述表），必须自己编程来定义8259A的各个软中断类型号和对应的处理程序。



8259A

- 8259A是一种PIC（Programmable Interrupt Controller，可编程中断控制器）
- 8259是Intel于1976年作为8位处理器8085支持芯片的一部分引进的， 8259A作为ISA总线的中断控制器被包含在1981年推出的最初IBM PC机（采用8088 CPU）中，1984年推出的PC/AT机（采用80286 CPU）中添加了第二个8259A芯片，现代PC的两个级联8259A芯片被集成在主板上的南桥中。





8259A的主要功能

- 在有多个中断源的系统中，接受外部的中断请求
- 进行判断，选中当前优先级最高的中断请求
- 将此请求送到CPU的INTR端
- 当CPU响应中断并进入中断子程序的处理过程后，中断控制器仍负责对外部中断请求的管理。
- 在一个8259A芯片中，有如下三个内部寄存器：
  - IMR（Interrupt Mask Register，中断屏蔽寄存器）——用作过滤被屏蔽的中断
  - IRR（Interrupt Request Register，中断请求寄存器）——用作暂时放置未被进一步处理的中断
  - ISR（In-Service Register，在使用中断）——当一个中断正在被CPU处理时，此中断被放置在ISR中。
- 8259A还有一个单元叫做优先级分解器（Priority Resolver），当多个中断同时发生时，优先级分解器根据它们的优先级，将高优先级者优先传递给CPU。

中断向量

主/从	中断请求	中断类型
主8259A	IRQ0	Intel 8253/8254可编程间隔计时器，即系统计时器
	IRQ1	Intel 8042键盘控制器
	IRQ2	级联从8259A
	IRQ3	8250 UART串口COM2和COM4
	IRQ4	8250 UART串口COM1和COM3
	IRQ5	在PC/XT中为硬盘控制器 在PC/AT以后为Intel 8255并行端口LPT2
	IRQ6	Intel 8272A软盘控制器
	IRQ7	Intel 8255并行端口LPT1/伪中断
从8259A	IRQ8	RTC（Real-Time Clock，实时时钟）
	IRQ9	无公共的指派
	IRQ10	无公共的指派
	IRQ11	无公共的指派
	IRQ12	Intel 8042 PS/2鼠标控制器
	IRQ13	数学协处理器
	IRQ14	硬盘控制器1
	IRQ15	硬盘控制器2

中断向量

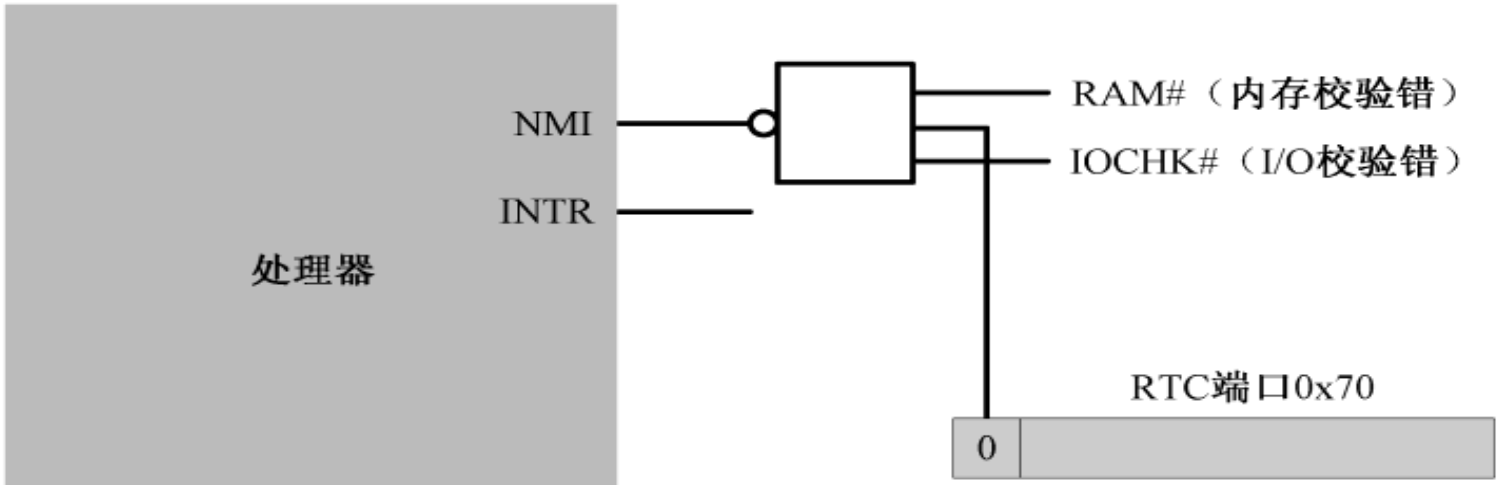
- UART=Universal Asynchronous Receiver/Transmitter，通用异步接收器/发送器。
- IRQ7曾被用于声卡，后来改用IRQ5。
- 串口IRQ3/4常被屏蔽，用于其他设备。
- IRQ2/9是MPU-401 MIDI的传统中断线，但在2000年末Intel公司为SMP/多核处理器引进的新中断控制器规范Intel APIC（Advanced Programmable Interrupt Controller，先进可编程中断控制器）架构中，IRQ9被用于SCI（Serial Communication Interface，串行通信接口）。

8259A的I/O端口

- 每个可编程中断控制器8259A都有两个I/O端口
- 主8259A所对应的端口地址为20h和21h，
- 从8259A所对应的端口地址为A0h和A1h。
- 程序员可以通过in/out指令读写这些端口，来操作这两个中断控制器。

时钟中断编程

- 任务：在屏幕右下角利用时钟中断轮流显示|、/、-和\
- 性质：编程一个中断服务程序
  - 两种定时方法：系统时钟和实时钟RTC
  - 确定中断号，系统时钟为08h/ RTC为70h
  - 设置中断向量：位置=中断号X4，4字节CS-IP
  - 中断服务程序设计



系统时钟的中断号和设置中断向量

- 连接在主8259A的0号引脚上
- 系统时钟中断号为08h
- 中断向量：地址为32、33、34、35共4字节
  - “高高低低”原则
  - 32和33保存IP
  - 34和35保存CS

程序说明与操作示范

```
    org 100h                                ; 程序加载到100h，可用于生成COM
    ; 设置时钟中断向量（08h），初始化段寄存器
            xor ax,ax                       ; AX = 0
            mov es,ax                       ; ES = 0
            mov word[es:20h],Timer          ; 设置时钟中断向量的偏移地址
            mov ax,cs
            mov [es:22h],ax                 ; 设置时钟中断向量的段地址=CS
            mov ds,ax                       ; DS = CS
            mov es,ax                       ; ES = CS
    ; 在屏幕右下角显示字符‘!’
            mov     ax,0B800h               ; 文本窗口显存起始地址
            mov     gs,ax                   ; GS = B800h
            ;mov ah,0Fh                     ; 0000：黑底、1111：亮白字（默认值为07h）
            ;mov al,'!'                     ; AL = 显示字符值（默认值为20h=空格符）
            ;mov [gs:((80*24+79)*2)],ax     ; 屏幕第 24 行, 第 79 列
            jmp $                           ; 死循环
    ; 时钟中断处理程序
            delay equ 4                     ; 计时器延迟计数
            count db delay                  ; 计时器计数变量，初值=delay
    Timer:
            dec byte[count]                 ; 递减计数变量
            jnz end                         ; >0：跳转
            inc byte[gs:((80*24+79)*2)]     ; =0：递增显示字符的ASCII码值
            mov byte[count],delay           ; 重置计数变量=初值delay
    end:
            mov al,20h                      ; AL = EOI
            out 20h,al                      ; 发送EOI到主8529A
            out 0A0h,al                     ; 发送EOI到从8529A
            iret                            ; 从中断返回
```

程序说明与操作示范(NASM版本)

```
    org 100h                                ; 程序加载到100h，可用于生成COM
    ; 设置时钟中断向量（08h），初始化段寄存器
            xor ax,ax                       ; AX = 0
            mov es,ax                       ; ES = 0
            mov word [es:20h],Timer         ; 设置时钟中断向量的偏移地址
            mov ax,cs
            mov word [es:22h],ax            ; 设置时钟中断向量的段地址=CS
            mov ds,ax                       ; DS = CS
            mov es,ax                       ; ES = CS
    ; 在屏幕右下角显示字符‘!’
            mov     ax,0B800h               ; 文本窗口显存起始地址
            mov     gs,ax                   ; GS = B800h
            mov ah,0Fh                      ; 0000：黑底、1111：亮白字（默认值为07h）
            mov al,'!'                      ; AL = 显示字符值（默认值为20h=空格符）
            mov [gs:((80*12+39)*2)],ax      ; 屏幕第 24 行, 第 79 列
            jmp $                           ; 死循环
    ; 时钟中断处理程序
            delay equ 4                     ; 计时器延迟计数
            count db delay                  ; 计时器计数变量，初值=delay
    Timer:
            dec byte [count]                ; 递减计数变量
            jnz end                         ; >0：跳转
            inc byte [gs:((80*12+39)*2)]    ; =0：递增显示字符的ASCII码值
            mov byte[count],delay           ; 重置计数变量=初值delay
    end:
            mov al,20h                      ; AL = EOI
            out 20h,al                      ; 发送EOI到主8529A
            out 0A0h,al                     ; 发送EOI到从8529A
            iret                            ; 从中断返回
```

操作系统核心的基本任务

- 操作系统内核的一项基本任务是捕捉和响应各种中断事件，实现硬件系统操作的控制、为进程提供功能服务或安全保护。中断处理程序（interrupt handler）通常是操作系统核心中最基础的一部分，负责确定中断的性质并执行所需的操作。

实验四说明

- 操作系统工作期间，利用时钟中断，在屏幕24行79列位置轮流显示’|’、’/’和’\’，适当控制显示速度，以方便观察效果。
- 编写键盘中断响应程序，原有的你设计的用户程序运行时，键盘事件会做出有事反应：当键盘有按键时，屏幕适当位置显示”OUCH! OUCH!”。
- 在内核中，对33号、34号、35号和36号中断编写中断服务程序，分别在屏幕1/4区域内显示一些个性化信息。再编写一个汇编语言的程序，作为用户程序，利用int 33、int 34、int 35和int 36产生中断调用你这4个服务程序。