操作系统设计及实践

《操作系统原理》配套实验

操作系统课程组

2022年10月

操作系统设计实验系列（四）

中断与异常

武漢大学

一、实验目标

- 理解中断与异常机制的实现机理
- 对应章节：第三章3.4节
- 3.5节大家了解即可。

二、本次实验内容

1. 理解中断与异常的机制
2. 调试8259A的编程基本例程
3. 调试时钟中断例程
4. 实现一个自定义的中断向量，功能可自由设想。

三、完成本次实验要回答的问题

1. 什么是中断，什么是异常
2. 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？
3. 如何建立IDT，如何实现一个自定义的中断
4. 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？
5. 简要解释一下IOPL的作用与基本机理

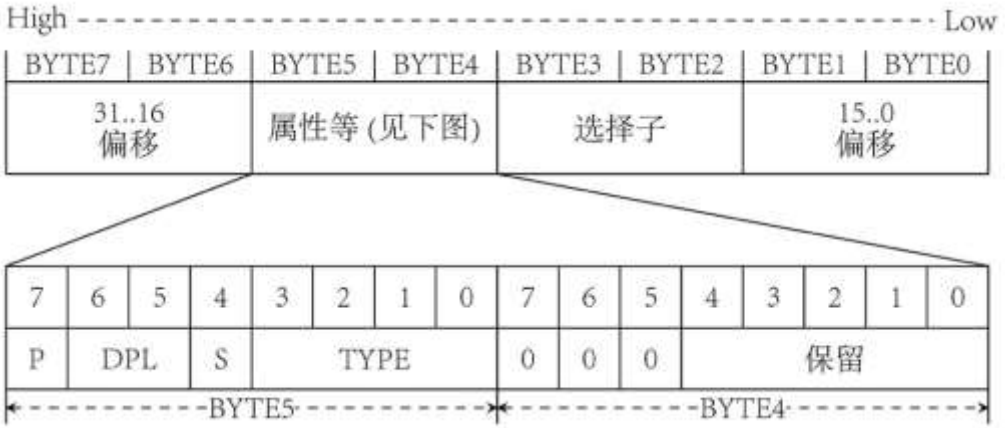
四、需了解的知识

1.为什么要IDT（中断描述符表）

- 实模式：BIOS中断
- 保护模式：IDT机制

2.IDT描述符分类

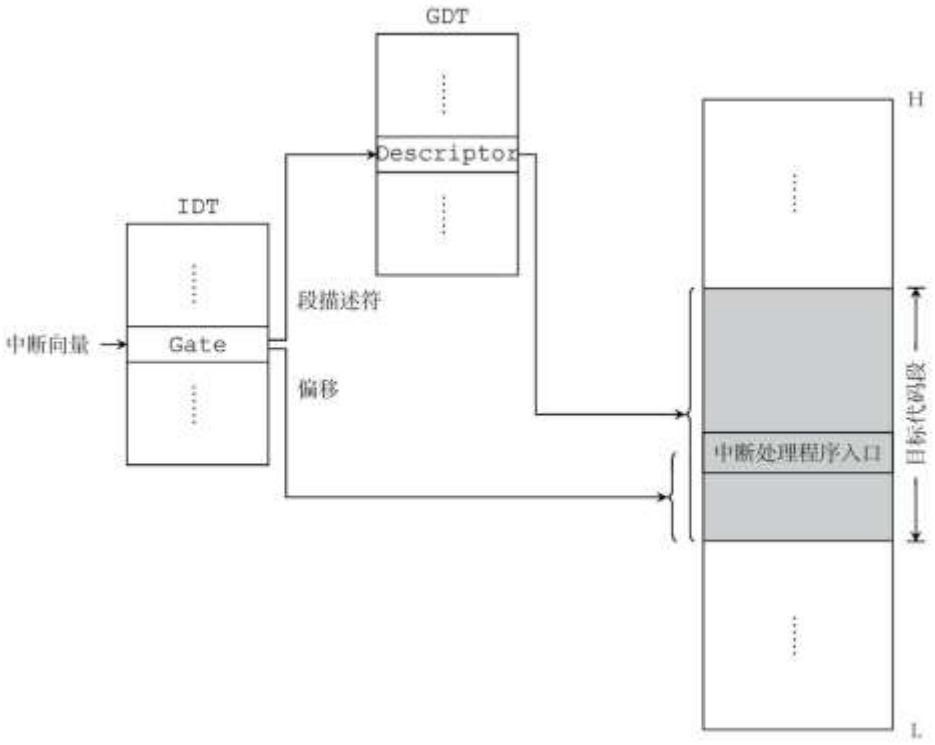
- 中断门描述符
- 陷阱门描述符
- 任务门描述符



四、需了解的知识

3.IDT的作用与基本流程

- 关联中断向量和描述符



四、需了解的知识

4.回顾什么是中断和异常

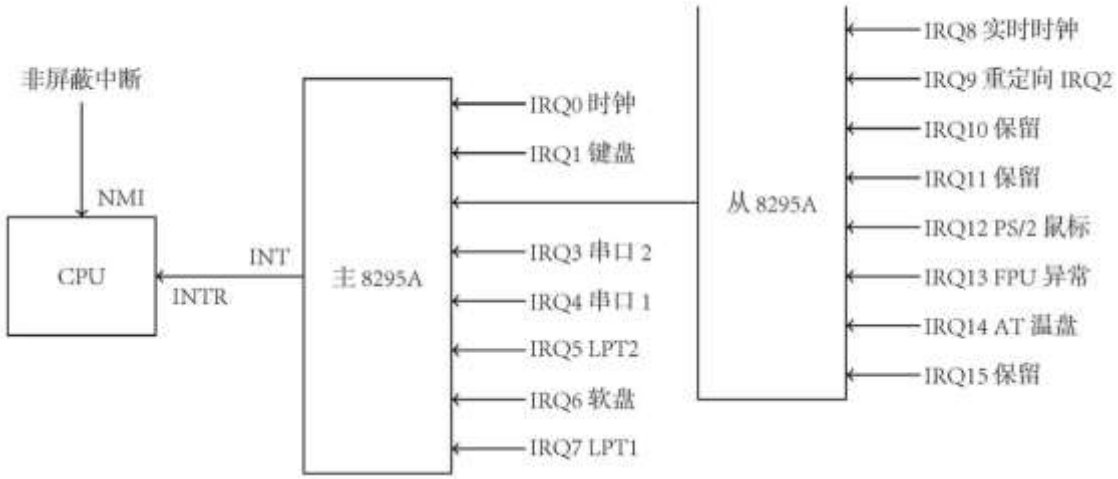
- 同步中断
- 异步中断
- Fault、Trap、Abort的区别

向量号	助记符	描述	类型	出错码	源
0	#DE	除法错	Fault	无	DIV 和 IDIV 指令
1	#DB	调试异常	Fault/Trap	无	任何代码和数据的访问
2	—	非屏蔽中断	Interrupt	无	非屏蔽外部中断
3	#BP	调试断点	Trap	无	指令 INT 3
4	#OF	溢出	Trap	无	指令 INTO
5	#BR	越界	Fault	无	指令 BOUND
6	#UD	无效（未定义的）操作码	Fault	无	指令 UD2 或者无效指令
7	#NM	设备不可用（无数学协处理器）	Fault	无	浮点或 WAIT/FWAIT 指令
8	#DF	双重错误	Abort	有 (0)	所有能产生异常或 NMI 或 INTR 的指令
9		协处理器段越界（保留）	Fault	无	浮点指令（386 之后的 IA32 处理器不再产生此种异常）
10	#TS	无效 TSS	Fault	有	任务切换或访问 TSS 时
11	#NP	段不存在	Fault	有	加载段寄存器或访问系统段时
12	#SS	堆栈段错误	Fault	有	堆栈操作或加载 SS 时
13	#GP	常规保护错误	Fault	有	内存或其他保护检验
14	#PF	页错误	Fault	有	内存访问
15	—	Intel 保留，未使用			
16	#MF	x87FPU 浮点错（数学错）	Fault	无	x87FPU 浮点指令或 WAIT/FWAIT 指令
17	#AC	对齐检验	Fault	有 (0)	内存中的数据访问（486 开始支持）
18	#MC	Machine Check	Abort	无	错误码（如果有的话）和源依赖于具体模式（奔腾 CPU 开始支持）
19	#XF	SIMD 浮点异常	Fault	无	SSE 和 SSE2 浮点指令（奔腾 III 开始支持）
20~31	—	Intel 保留，未使用			
32~255	—	用户定义中断	Interrupt		外部中断或 int n 指令

四、需了解的知识

5.外部中断

- 早期Intel 80x86用PIC方式来实现中断控制系统
- 两片8259A（主、从）实现



四、需了解的知识

- 8259A内部有三种8位的寄存器：IRR，ISR，IMR。都是8位，每一位对应一个IRQ。
- IRR，中断请求状态寄存器
  - 用来标记到达的中断请求。
  - 当一个中断请求到达8259A的一个引脚的时候，对应的IRR上的位就被设置，表示该引脚上有一个中断到来了。
- 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。
  - 当一个中断请求到达并且被记录到IRR之后，8259A在适当的时候通知处理器产上了一个这样的中断，等待处理器处理，这个时候ISR对应的位就会被设置，IRR对应的位就被清除。
- IMR，中断屏蔽状态寄存器
  - 用来记录屏蔽的中断请求。
  - 当8259A要屏蔽某一中断的时候就将IMR对应的位设置成1。

四、需了解的知识

8259A的中断响应过程如下：

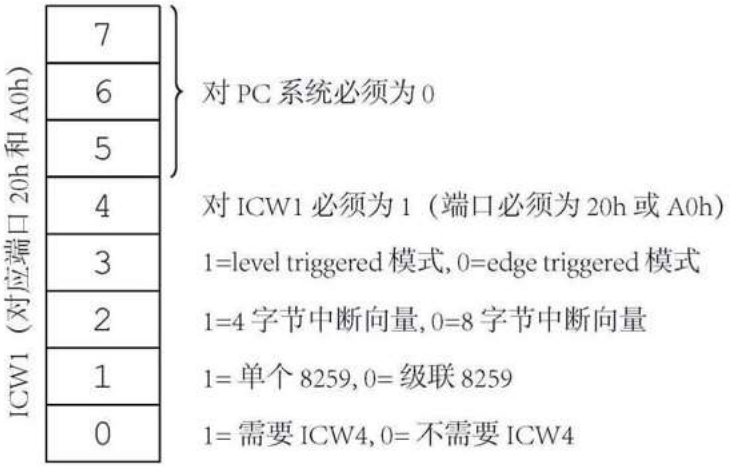
- 当某条IRQ线上发生了中断请求，8259A设置IRR相应的位，表示发生了中断请求。
- 查看IMR是否屏蔽了该中断，如果没有屏蔽则给CPU发送INTR。
- CPU在接收到INTR之后会回复INTA。
- 当收到第一个INTA之后，8259A进行优先级仲裁，优先级高的中断得到响应，设置相应的ISR位，并且清空对应的IRR位。
- 当收到第二个INTA之后，8259A将相应的中断向量通过数据总线传递给CPU。
- 如果是自动EOI模式，在第二个INTA处理完成之后ISR对应的位自动清空，否则必须接收到一个正常EOI之后8259A才能清空对应的ISR位。

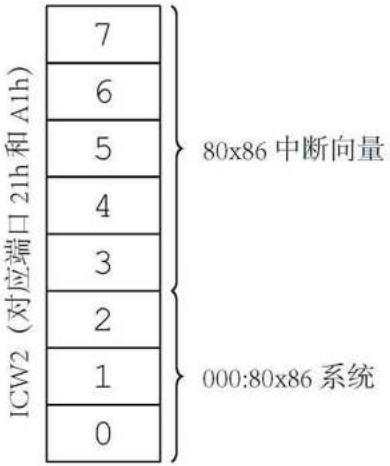
四、需了解的知识

5.外部中断：8259A的编程方式

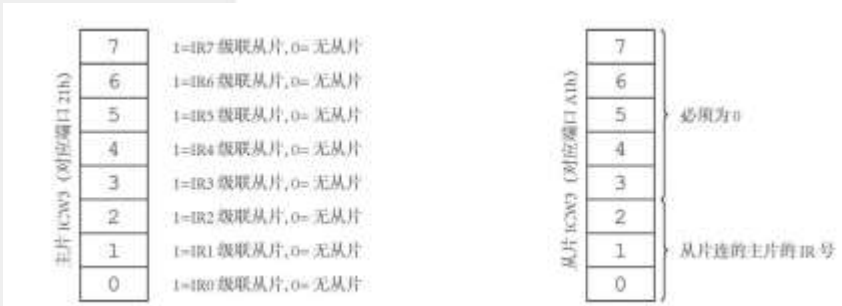
- 8259A工作模式包括：编程模式、操作模式
- 主8259A的端口地址20h，21h；从8259A端口A0h，A1h
- 指令格式
  - ICW，初始化命令字，ICW1—ICW4，描述详见P92
  - OCW，操作控制字，OCW1-OCW3
- 编程顺序，注意不能颠倒！详见代码3.34
  - 向20h或者A0h，写入ICW1，主从格式一致
  - 向21h或者A1h，写入ICW2，主从格式一致；定义INT0号腿，多对应的中断向量，后面的自动递增
  - 向21h或者A1h，写入ICW3，主从格式不同；定义的是主片的哪个IR腿脚级联从片
  - 向21h或者A1h，写入ICW4，主从格式一致
  - OCW1，屏蔽中断；OCW2:EOI
- 汇编编程tips：
  - out　端口号， 寄存器
  - 向该端口写入，对CPU相当于输出
  - ICW2初始化中断向量，只需要初始化每一片的INT0号即可

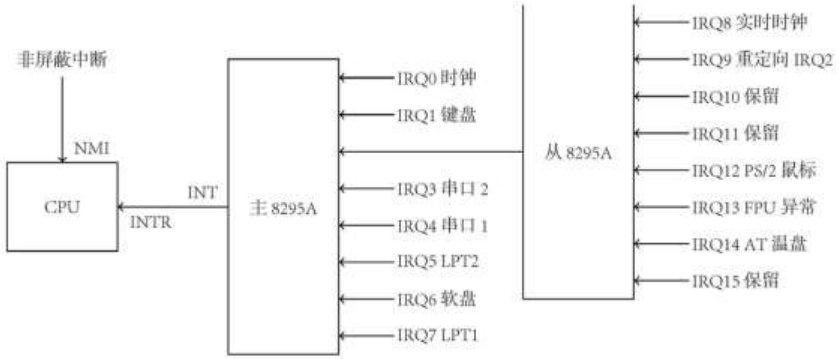
```
283  Init8259A:
284          mov         al, 011h
285          out         020h, al        ; 主8259, ICW1.
286          call        io_delay
287
288          out         0A0h, al        ; 从8259, ICW1.
289          call        io_delay
290
291          mov         al, 020h        ; IRQ0 对应中断向量 0x20
292          out         021h, al        ; 主8259, ICW2.
```





```
293         call      io_delay
294
295         mov       al, 028h          ; IRQ8 对应中断向量 0x28
296         out       0A1h, al          ; 从8259, ICW2.
297         call      io_delay
298
299         mov       al, 004h          ; IR2 对应从8259
300         out       021h, al          ; 主8259, ICW3.
301         call      io_delay
302
303         mov       al, 002h          ; 对应主8259的IR2
304         out       0A1h, al          ; 从8259, ICW3.
305         call      io_delay
306
307         mov       al, 001h
308         out       021h, al          ; 主8259, ICW4.
309         call      io_delay
310
311         out       0A1h, al           ; 从8259, ICW4.
312         call      io_delay
313
314         mov       al, 11111110b     ; 仅仅开启定时器中断
315         ;mov      al, 11111111b     ; 屏蔽主8259所有中断
316         out       021h, al          ; 主8259, OCW1.
317         call      io_delay
318
319         mov       al, 11111111b     ; 屏蔽从8259所有中断
320         out       0A1h, al          ; 从8259, OCW1.
321         call      io_delay
322
```





四、需了解的知识

6.建立IDT：

```
100  LABEL_IDT:
101  ; 门                    目标选择子,            偏移, DCount, 属性
102  %rep 255
103                  Gate    SelectorCode32, SpuriousHandler, 0, DA_386IGate
104  %endrep
105
106  IdtLen          equ     $ - LABEL_IDT
107  IdtPtr          dw      IdtLen - 1        ; 段界限
108                  dd      0                 ; 基地址
```

```
 97  [SECTION .idt]
 98  ALIGN   32
 99  [BITS   32]
100  LABEL_IDT:
101  ; 门                      目标选择子,          偏移,  DCount, 属性
102  %rep 128
103                  Gate    SelectorCode32, SpuriousHandler, 0, DA_386IGate
104  %endrep
105  .080h:          Gate    SelectorCode32, UserIntHandler,  0, DA_386IGate
106
107  IdtLen          equ     $ - LABEL_IDT
108  IdtPtr          dw      IdtLen - 1        ; 段界限
109                  dd      0                 ; 基地址
110  ; END of [SECTION .idt]
```

谢　谢！