



操作系统

第二章

# 操作系统运行机制与用户界面

文艳军（教授）  
计算机学院

# 本章讲授计划

- ① 中断和异常
- ② 中断/异常的响应和处理过程
- ③ 系统调用

# 目录

一. 版本1内核简介

二. 中断

三. 异常

四. 自学&讨论

五. 答疑

# 英特尔32位体系架构（IA32）

- Intel Architecture 32bit，简称IA32，常被称为i386、x86-32或是x86。
- 是英特尔公司推出的一种复杂指令集（CISC）架构，在80386微处理器（1985年）中首先采用，至今英特尔最受欢迎的处理器仍然采用此架构。

# IA32的寄存器结构

|     |    |    |    |   |   |              |     |    |    |    |   |  |    |
|-----|----|----|----|---|---|--------------|-----|----|----|----|---|--|----|
|     | 31 | 16 | 15 | 8 | 7 | 0            |     | 31 | 16 | 15 | 0 |  |    |
| EAX |    |    |    |   |   | (AH) AX (AL) | ESP |    |    |    |   |  | SP |
| EBX |    |    |    |   |   | (BH) BX (BL) | EBP |    |    |    |   |  | BP |
| ECX |    |    |    |   |   | (CH) CX (CL) | ESI |    |    |    |   |  | SI |
| EDX |    |    |    |   |   | (DH) DX (DL) | EDI |    |    |    |   |  | DI |

|        |   |  |               |   |    |   |    |   |
|--------|---|--|---------------|---|----|---|----|---|
| 15     | 0 |  | 31            | 0 | 19 | 0 | 11 | 0 |
| CS 选择器 |   |  | CS 描述符高速缓存寄存器 |   |    |   |    |   |
| SS 选择器 |   |  | SS 描述符高速缓存寄存器 |   |    |   |    |   |
| DS 选择器 |   |  | DS 描述符高速缓存寄存器 |   |    |   |    |   |
| ES 选择器 |   |  | ES 描述符高速缓存寄存器 |   |    |   |    |   |
| FS 选择器 |   |  | FS 描述符高速缓存寄存器 |   |    |   |    |   |
| GS 选择器 |   |  | GS 描述符高速缓存寄存器 |   |    |   |    |   |

|     |    |    |    |   |    |        |    |    |   |  |       |
|-----|----|----|----|---|----|--------|----|----|---|--|-------|
|     | 31 | 16 | 15 | 0 |    | 31     | 16 | 15 | 0 |  |       |
| EIP |    |    |    |   | IP | EFLAGS |    |    |   |  | FLAGS |

|      |    |   |    |   |
|------|----|---|----|---|
|      | 31 | 0 | 15 | 0 |
| GDTR |    |   |    |   |
| IDTR |    |   |    |   |

|      |          |   |  |           |   |    |   |    |   |
|------|----------|---|--|-----------|---|----|---|----|---|
|      | 15       | 0 |  | 31        | 0 | 19 | 0 | 11 | 0 |
| LDTR | LDTR 选择器 |   |  | LDTR 高速缓存 |   |    |   |    |   |
| TR   | TR 选择器   |   |  | TR 高速缓存   |   |    |   |    |   |

# 第2、3、5章实验内容(版本1内核)

## ■ 分析一个只具有两个进程的Linux 0.11内核映像的执行过程



demo

```

sched_init();
buffer_init(buffer_memory_end);
hd_init();
floppy_init();
sti(); /* wyj */
move_to_user_mode();

```

完成初始化，进入  
用户态

```

if (!fork()) { /* we count on this going ok */
    (void)mysignal(SIGALRM, SIG_IGN);
    for(;;) {
        task1(); /* wyj */
        alarm(1);
        pause();
    }
}
init(); /*

```

1秒后唤醒

睡眠

1号进程

NOTE!! For any other task 'pause()' would mean we have to get a signal to awaken, but task0 is the sole exception (see 'schedule()') as task 0 gets activated at every idle moment (when no other tasks can run). For task0 'pause()' just means we go check if some other task can run, and if not we return here.

```

for(;;) {
    task0(); /* wyj */
    pause();
}

```

0号进程

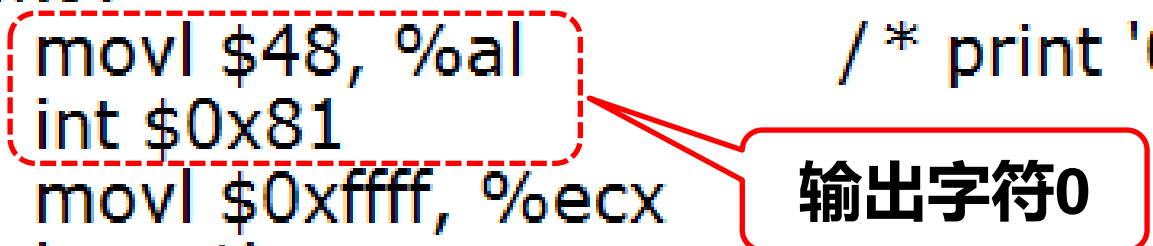
} ? end main ?

```
_task0:
    movl $48, %al          /* print '0' */
    int $0x81
    movl $0xffff, %ecx
1:    loop 1b

    movl $32, %al          /* print ' ' */
    int $0x81
    movl $0xffff, %ecx
1:    loop 1b

    movl $32, %al          /* print ' ' */
    int $0x81
    movl $0xffff, %ecx
1:    loop 1b

    ret
```



输出字符0



# 目录

一. 版本1内核简介

二. 中断

三. 异常

四. 自学&讨论

五. 答疑

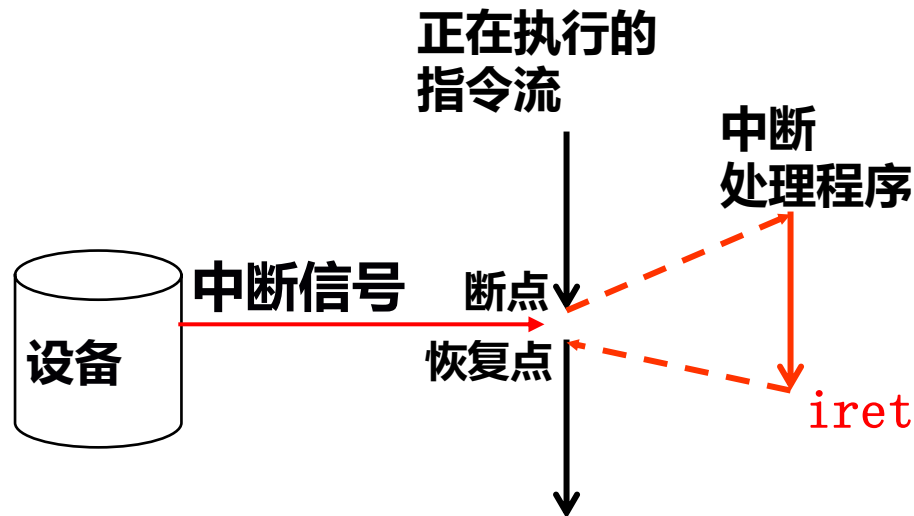
## 二. 中断

### ■ 什么是中断？

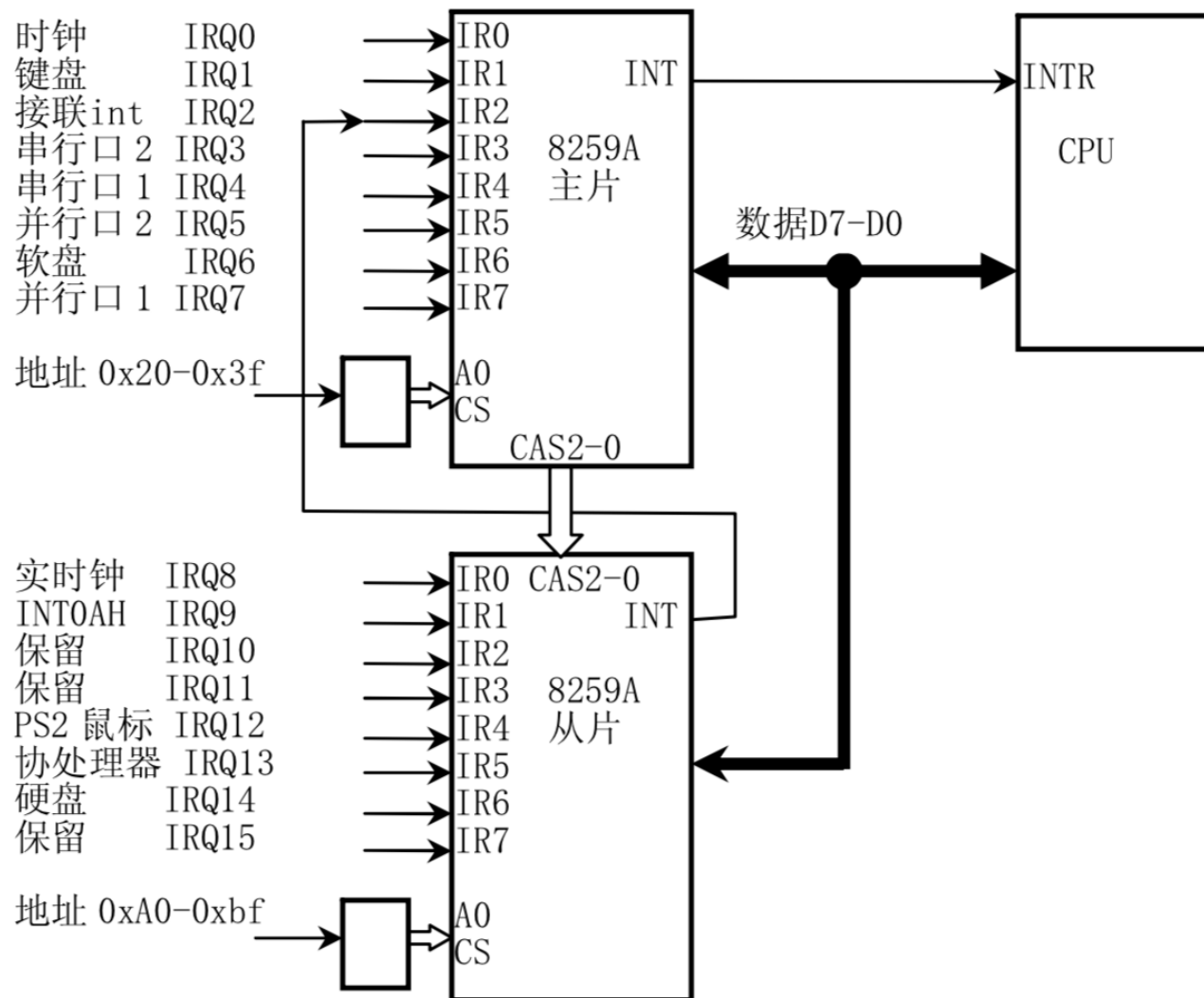
- 中断是CPU收到的一种来自外部设备的控制信号
- 又称为**外部中断**

例：

- I/O中断
- 时钟中断
- `ctl-c`



# 80x86微机的中断子系统



## 二. 中断

### ■ 演示:

#### 时钟中断的发生

- 观察: `do_timer`, `jiffies`
- 分析: 断点和恢复点

# 操作系统的整体运行视图

进程1:

```
main() {  
    ...  
}
```

进程2:

```
main() {  
    ...  
}
```

□ □ □

用户态

核心态

timer\_interrupt:

```
{  
    ...  
}
```

keyboard\_interrupt:

```
{  
    ...  
}
```

hd\_interrupt:

```
{  
    ...  
}
```

□ □ □

# 目录

一. 版本1内核简介

二. 中断

三. 异常

四. 自学&讨论

五. 答疑

# 四. 异常

## ■ 什么是异常？

□ **异常**是CPU在执行指令时内部产生的一种信号，通常意味着某种意外事情的发生

□ 也称为**内中断**

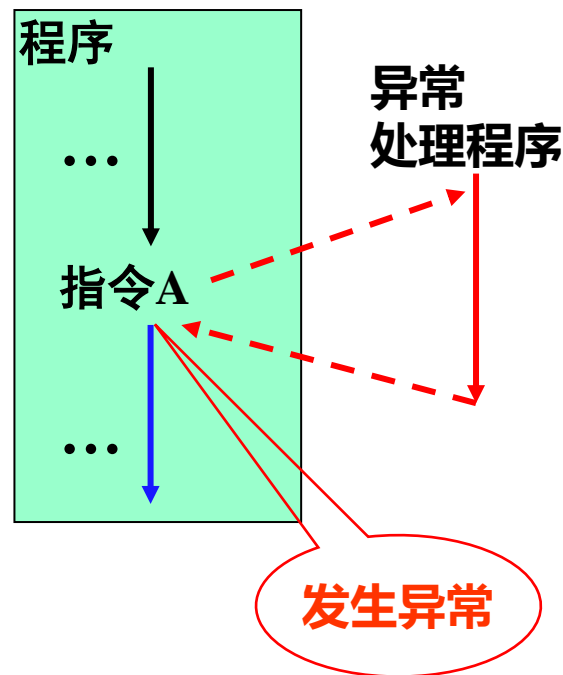
□ 处理机制与中断类似

中断(狭义)与异常的区别：

- 中断：与正执行指令无关，可以屏蔽
- 异常：与正执行指令有关，不可屏蔽

例：

- 系统调用
- 缺页
- 断点指令
- 算术溢出



# 80x86 CPU的中断/异常ID分配

Table 9-1. Interrupt and Exception ID Assignments

| Identifier | Description                                    |
|------------|--|
| 0          | Divide error                                   |
| 1          | Debug exceptions                               |
| 2          | Nonmaskable interrupt                          |
| 3          | Breakpoint (one-byte INT 3 instruction)        |
| 4          | Overflow (INTO instruction)                    |
| 5          | Bounds check (BOUND instruction)               |
| 6          | Invalid opcode                                 |
| 7          | Coprocessor not available                      |
| 8          | Double fault                                   |
| 9          | (reserved)                                     |
| 10         | Invalid TSS                                    |
| 11         | Segment not present                            |
| 12         | Stack exception                                |
| 13         | General protection                             |
| 14         | Page fault                                     |
| 15         | (reserved)                                     |
| 16         | Coprocessor error                              |
| 17-31      | (reserved)                                     |
| 32-255     | Available for external interrupts via INTR pin |



# IA32的中断/异常分类

| Class     | Cause                         | Async/Sync | Return behavior                     |
|-----------|-------------------------------|------------|-------------------------------------|
| Interrupt | Signal from I/O device        | Async      | Always returns to next instruction  |
| Trap      | Intentional exception         | Sync       | Always returns to next instruction  |
| Fault     | Potentially recoverable error | Sync       | Might return to current instruction |
| Abort     | Nonrecoverable error          | Sync       | Never returns                       |

| Exception number | Description              | Exception class   |
|------------------|--------------------------|-------------------|
| 0                | Divide error             | Fault             |
| 13               | General protection fault | Fault             |
| 14               | Page fault               | Fault             |
| 18               | Machine check            | Abort             |
| 32–127           | OS-defined exceptions    | Interrupt or trap |
| 128 (0x80)       | System call              | Trap              |
| 129–255          | OS-defined exceptions    | Interrupt or trap |

## 四. 异常

### ■ 演示:

#### 异常的发生

□ 位置: 除零指令 (**idiv**)

**task0, int 0x81**

□ **int** 指令: 软中断指令, 会产生异常

# 操作系统的整体运行视图

进程1:

```
main() {  
    ...  
}
```

进程2:

```
main() {  
    ...  
}
```

□ □ □

用户态

核心态

timer\_interrupt:

```
{  
    ...  
}
```

hd\_interrupt:

```
{  
    ...  
}
```

divide\_error:

```
{  
    ...  
}
```

□ □ □

# 目录

一. 版本1内核简介

二. 中断

三. 异常

四. 自学&讨论

五. 答疑

# 目录

一. 版本1内核简介

二. 中断

三. 异常

四. 自学&讨论

五. 答疑

# 答疑

- 中断处理程序可以嵌套执行吗？可以重入吗？
- 异常处理程序可以嵌套执行吗？可以重入吗？

# 小结

## 一. 版本1内核简介

两个进程，task0，task1，int 0x81

## 二. 中断：INTR，中断处理程序，断点和恢复点

## 三. 异常：中断号的分配，异常的类型