



中国科学技术大学

University of Science and Technology of China

011174.01: Operating System 操作系统原理与设计

Project 3: Interrupt&timer&shell

陈香兰(xlanchen@ustc.edu.cn)

高效智能计算实验室, CS, USTC @ 合肥

嵌入式系统实验室, CS, USTC @ 苏州

实验3基础



中国科学技术大学
University of Science and Technology of China

- 本实验在实验2的基础上进行
- 在实验2提交的截止时间过后，同学们可以就实验2的内容互通有无
- 实验3可以在其他同学实验2的基础上进行
 - 无论你使用哪一个（包括自己的），请在实验报告中标注，实验3的基础来自哪个同学（可以是自己）
 - 给你使用的实验2 打分



- **【必须】**简单的shell程序，提供cmd和help命令，允许注册新的命令
- **【必须】**中断机制和中断控制器i8259A初始化
- **【必须】**时钟i8253和周期性时钟中断
- **【必须】**VGA输出的调整：
 - 左下角：时钟中断之外的其他中断，一律输出“unknown interrupt”
 - 右下角：从某个时间开始，大约每秒更新一次
格式为：HH:MM:SS
- **【必须】**采用自定义测试用例和用户（助教）测试用例相结合的方式验收
- **【必须】**提供脚本完成编译和执行

提纲



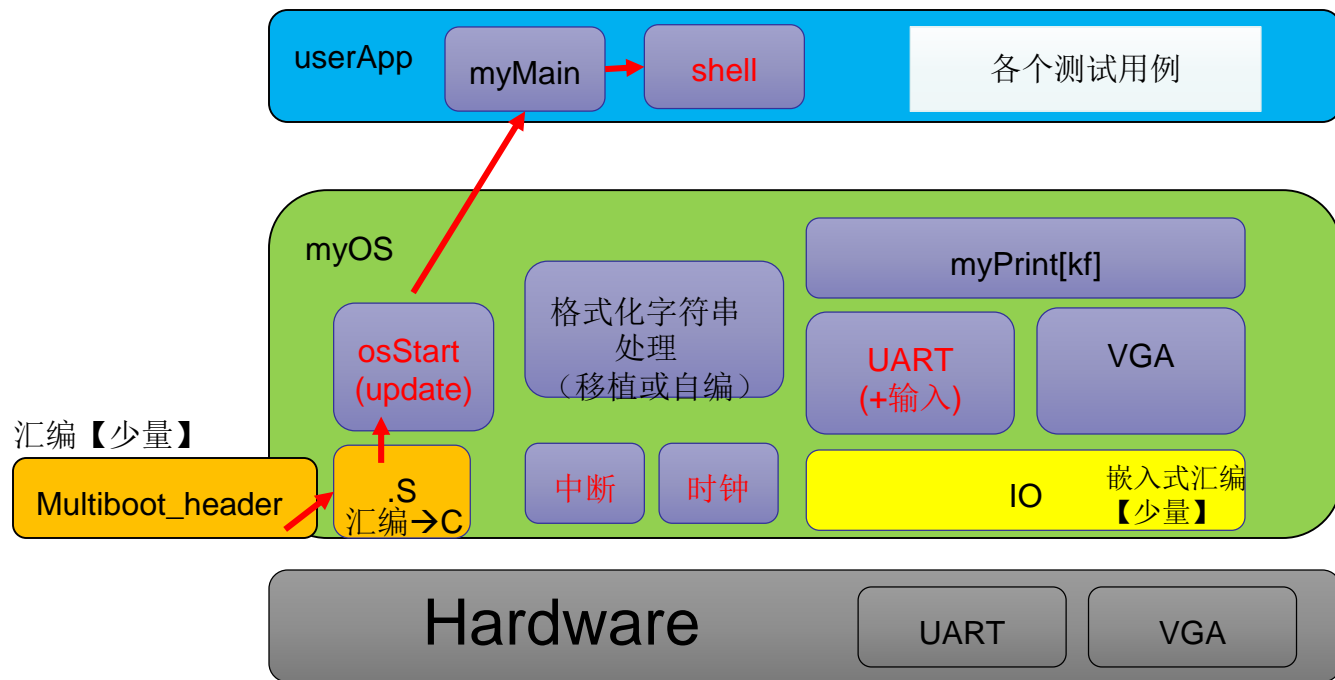
中国科学技术大学
University of Science and Technology of China

1. 软件架构和功能说明
2. 主要功能模块说明
 1. 中断机制的初始化
 2. Tick的实现
 3. wallClock的实现
 4. 部分汇编代码参考
 5. Shell的实现
3. 关于Qemu的串口重定向
4. 验收标准

1 软件架构和功能



中国科学技术大学
University of Science and Technology of China



multiboot_header → myOS → userApp

主要功能模块【新】：

内核：中断处理、时钟
用户：shell

流程：

Multiboot_header

为进入C程序准备好上下文

初始化操作系统

调用userApp入口
myMain（自测）+shell

测试：

被测功能：中断、时钟、shell

自测：userApp

他测：替换userApp或增加shell命令
制造除0错
按键制作键盘中断

参考目录结构



中国科学技术大学
University of Science and Technology of China

```
Makefile
README shell interrupt_timer.txt
multibootheader
  multibootHeader.S
myOS
  Makefile
  dev
    Makefile
    i8253.c
    i8259A.c
    uart.c
    vga.c
  i386
    Makefile
    io.c
    io.h
    irq.S
    irqs.c
  kernel
    Makefile
    tick.c
    wallClock.c
  lib
    Makefile
    string.c
  myOS.ld
  osStart.c
  printk
    Makefile
    myPrintk.c
    types.h
    vsprintf.c
  start32.S
source2img.sh
userApp
  Makefile
  main.c
  shell.c
```

放大了看吧 ^^

2.1 中断机制及其初始化



清华大学
Tsinghua University

1. 中断描述符表IDT及其初始化

- 为IDT分配一块内存（静态分配）
- 将所有中断处理程序初始化为合适的缺省处理函数，例如ignore_int1

2. 寄存器IDTR的初始化，指令：lidt

- Registers **IDTR** stores the base address of a IDT
 - The IDTR is 48 bits long
 - The lower 16 bits tell the size of the IDT, and the upper 32 bits tell the location of the IDT in memory.
 - instruction: `lidt src` 如: `lidt idtpr`



```
/* ===== data ===== */
.data

# IDT
.p2align 4
.globl IDT
IDT:
.rept 256
.word 0,0,0,0
.endr

idtpr:
.word (256*8 - 1)
.long IDT
```

```
.p2align 4
ignore_int1:
cld
pusha
call ignoreIntBody
popa
iret
```

```
void ignoreIntBody(void){
    put_chars("Unknown interrupt1\0",0x4,24,0);
}
```

3. 可编程中断控制器PIC i8259

4. 开关中断，指令：sti和cli

- 标志寄存器eflags的IF位

接口：

```
void enable_interrupt(void);
void disable_interrupt(void);
```

```
.globl enable_interrupt
enable_interrupt:
sti
ret

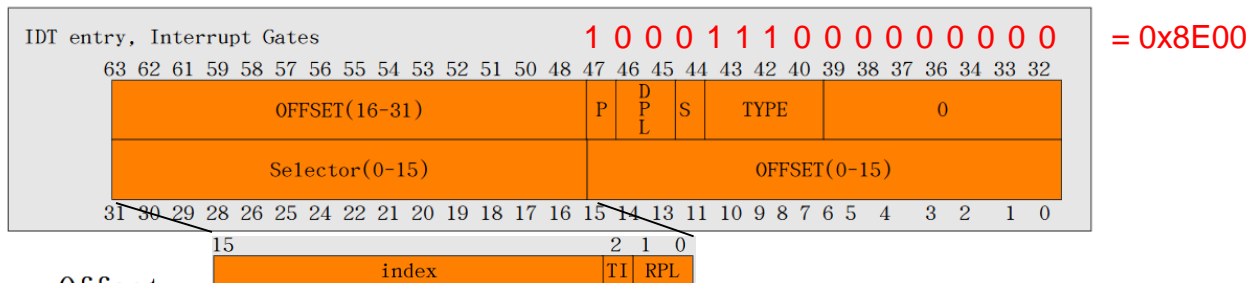
.globl disable_interrupt
disable_interrupt:
cli
ret
```

中断描述符表IDT



中国科学技术大学
University of Science and Technology of China

- 中断描述符表IDT: 256个中断描述符
- 每个描述符8个字节



- Offset
- P: Present; Set to 0 for unused interrupts or for Paging.
- DPL: Gate call protection.
- S: Set to 0 for interrupt gates
- Type:
 - 0b0101: 32bit task gate
 - 0b1110: 32-bit interrupt gate
 - 0b1111: 32-bit interrupt gate

```
setup_idt:
    movl $ignore_int1,%edx
    movl $0x00080000,%eax /* selector */
    movw %dx,%ax
    movw $0x8E00,%dx      /* interrupt gate - dpl=0, present */

    movl $IDT,%edi
    mov $256,%ecx

rp_sidt:
    movl %eax,(%edi)
    movl %edx,4(%edi)
    addl $8,%edi
    dec %ecx
    jne rp_sidt
```

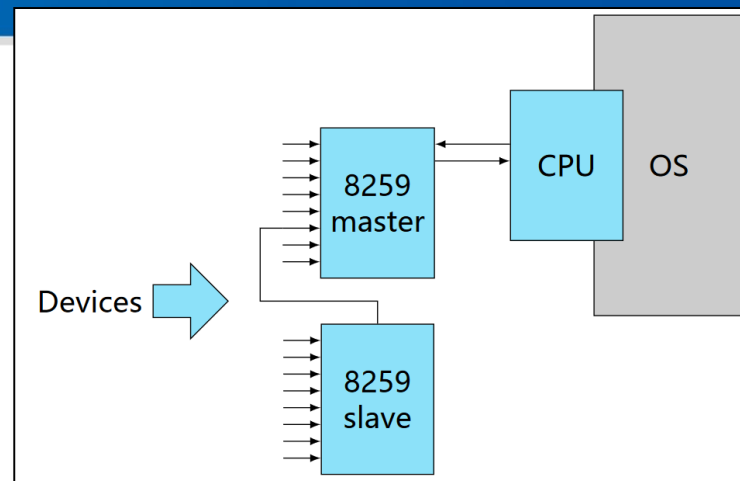

可编程中断控制器

PIC i8259



中国科学技术大学
University of Science and Technology of China

- 两个8259级联
- 需要对i8259进行初始化
接口: `void init8259A(void);`
 - 端口地址: 主片0x20~0x21
从片0xA0~0xA1
 - 屏蔽所有中断源: 0xFF ==》 0x21和0xA1
 - 主片初始化: ICW1: 0x11 ==》 0x20
ICW2: 起始向量号0x20 ==》 0x21
ICW3: 从片接入引脚位 0x04 ==》 0x21
ICW4: 中断结束方式 AutoEOI 0x3 ==》 0x21
 - 从片初始化: ICW1: 0x11 ==》 0xA0
ICW2: 起始向量号0x28 ==》 0xA1
ICW3: 接入主片的编号0x02 ==》 0xA1
ICW4: 中断结束方式 0x01 ==》 0xA1
- 读/写i8259的当前屏蔽字节: 即读写主片0x21或从片0xA1



2.2 Tick和tick维护



中国科学技术大学
University of Science and Technology of China

- Tick（嘀嗒）：周期性时钟中断
 - 频率：100HZ（可以不必是100HZ）
 - 通过对8253编程来触发周期性时钟中断（见8253初始化）

- Tick发生时，做些什么

接口： `void tick(void);`

- Tick的维护：
使用一个全局变量来维护tick发生的次数
 - 初始化为0；每次tick，加1
- 随时间变化而进行的维护（包括其他模块所需）
 - 如wallClock
 - 少量（思考：hook机制） VS 大量？用户需要？

```
.p2align 4
time_interrupt:
    cld
    pushf
    pusha
    call tick
    popa
    popf
    iret
```

可编程间隔定时器

PIT: i8253



中国科学技术大学
University of Science and Technology of China

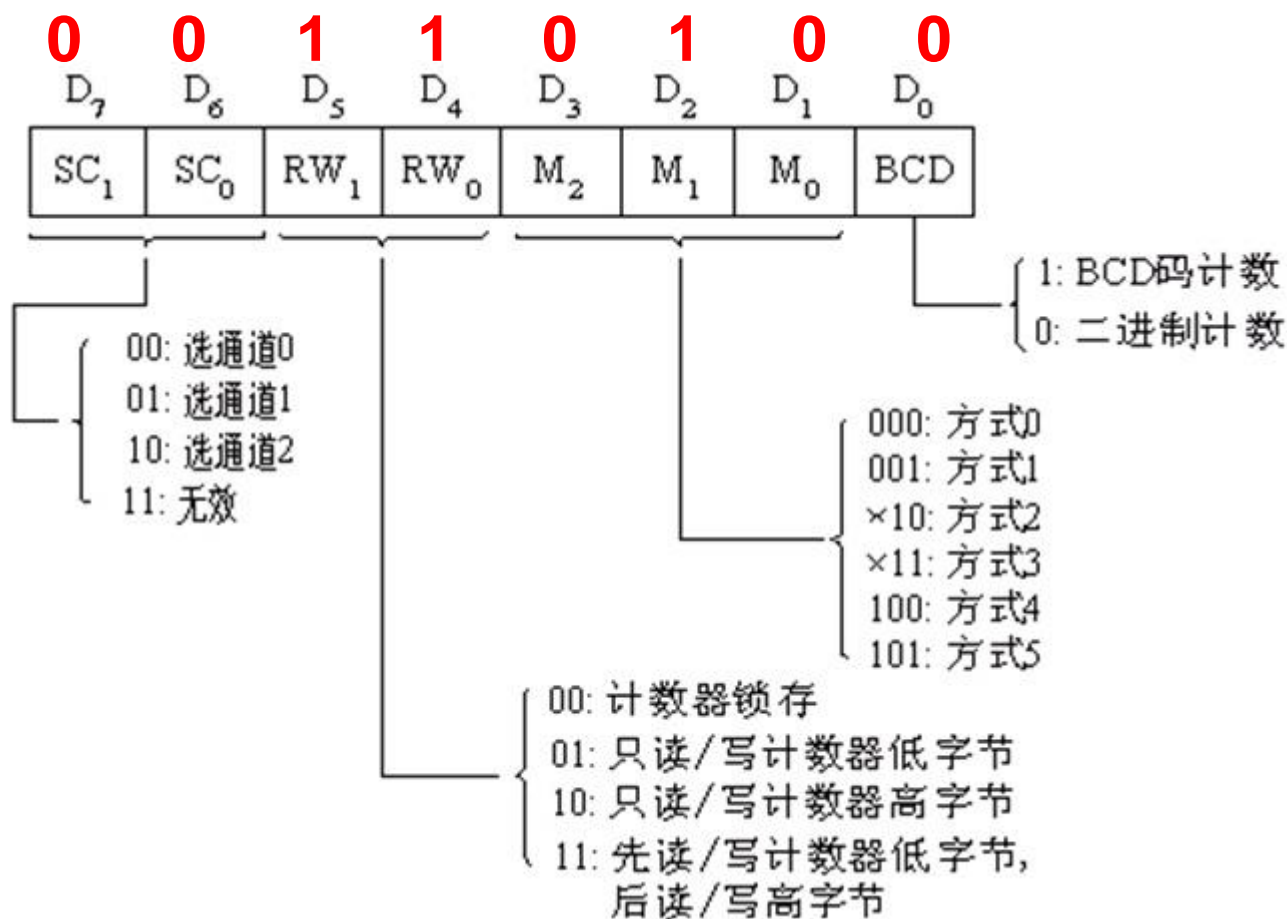
- 需要对PIT: i8253进行初始化，接口：void init8253(void)
 - 端口地址 0x40~0x43;
 - 14,3178 MHz crystal
4,772,727 Hz system clock
1,193,180 Hz to 8253
 - 设定时钟中断的频率为100HZ，分频参数是多少？
 - 初始化序列为：
 - 0x34 ==》端口0x43（参见控制字说明）
 - 分频参数 ==》端口0x40，分两次，先低8位，后高8位
 - 通过8259控制，允许时钟中断
 - 读取原来的屏蔽字，将最低位置0

PIT控制字



中国科学技术大学
University of Science and Technology of China

0x34的含义是？



2.3 维护墙钟和显示墙钟



- wallClock: 时hh:分mm:秒ss:毫秒ms
 - 根据tick和ms的关系, 维护ms, 取值范围[0~999]
 - 根据ms维护ss, 取值范围[0~59]
 - 根据ss维护mm, 取值范围[0~59]
 - 根据mm维护hh, 取值范围[0~23]
- 接口:
 - `void setWallClock(int h, int m, int s);` //设置墙钟的开始时间
 - `void getWallClock(int *h, int *m, int *s);` //读取墙钟
- 显示墙钟, 随着wallClock的维护而显示
 - 思考: hook机制
 - 接口: `void setWallClockHook(void (*func)(void));`
机制和策略相分离

允许User注册WallClock相关的操作, 使得该操作随时钟中断周期性运行

2.4 部分汇编代码



中国科学技术大学
University of Science and Technology of China

```
/* ===== code32 ===== */
```

```
.text
```

```
.globl _start      # GNU default entry point  
.globl osStart  
.globl ignoreIntBody
```

```
.code32
```

```
_start:  
    jmp establish_stack
```

```
dead: jmp dead      # Never here
```

```
# Set up the stack
```

```
establish_stack:  
    movl $0x80000, %eax
```

```
    movl %eax, %esp    # set stack pointer  
    movl %eax, %ebp    # set base pointer
```

```
# Zero out the BSS segment
```

```
zero_bss:
```

```
    cld                # make direction flag count up  
    movl $_end, %ecx   # find end of .bss  
    movl $_bss_start, %edi # edi = beginning of .bss  
    subl %edi, %ecx    # ecx = size of .bss in bytes  
    shrl %ecx          # size of .bss in longs  
    shr    %ecx
```

```
    xorl %eax, %eax    # value to clear out memory  
    repne stosl        # while ecx != 0  
                      # clear a long in the bss
```

```
setup_idtpr:
```

```
    call setup_idt  
    lidt idtpr
```

```
# Transfer control to main
```

```
to_osStart:  
    call osStart
```

```
shut_down:
```

```
    jmp shut_down      # Never here
```

仅供参考，不必完全一样



```
.p2align 4
time_interrupt:
    cld
    pushf
    pusha
    call tick
    popa
    popf
    iret

.p2align 4
ignore_int1:
    cld
    pusha
    call ignoreIntBody
    popa
    iret
```

```
setup_idt:
    movl $ignore_int1,%edx
    movl $0x00080000,%eax
    movw %dx,%ax      /* selector = 0x0010 = cs */
    movw $0x8E00,%dx  /* interrupt gate - dpl=0, present */

    movl $IDT,%edi
    mov $256,%ecx

rp_sidt:
    movl %eax,(%edi)
    movl %edx,4(%edi)
    addl $8,%edi
    dec %ecx
    jne rp_sidt

    # ret /* if do not set timer*/

setup_time_int_32:
    movl $time_interrupt,%edx
    movl $0x00080000,%eax /* selector: 0x0010 = cs */
    movw %dx,%ax
    movw $0x8E00,%dx    /* interrupt gate - dpl=0, present */

    movl $IDT,%edi
    addl $(32*8), %edi
    movl %eax,(%edi)
    movl %edx,4(%edi)

    ret
```

```
/* ===== data ===== */
.data

# IDT
    .p2align 4
    .globl IDT

IDT:
    .rept 256
    .word 0,0,0,0
    .endr

idtptr:
    .word (256*8 - 1)
    .long IDT
```

仅供参考，不
必完全一样

2.5 用户态Shell的实现



中国科学技术大学
University of Science and Technology of China

- 接口： `void startShell(void);`
- 功能：
 - 显示交互界面
 - 接收并处理命令行
 - 建议：一个命令的元信息为
(命令名, `func`, `help_func`, 描述命令的字符串)
 - 采用静态定义或动态注册的方式管理命令，至少提供2个命令
 - `cmd`，列出所有命令
 - `help [cmd]`，调用指定命令的`help`函数，若没有指定命令，则调用`help`的`help`函数
- 输入输出
 - 输入设备： `uart`
 - 回显和输出设备： `uart` 和 `VGA`

Uart的端口区间0x3F8~0x3FF;

假设已经初始化完毕:

数据输入输出: 0x3F8

状态寄存器: 0x3FD, 最低位: 1=数据可读

```
unsigned char uart_get_char(void){  
    while (!(inb(uart_base+5)&1));  
  
    return inb(uart_base);  
}
```


3 Qemu串口重定向



中国科学技术大学
University of Science and Technology of China

qemu-system-i386 -kernel output/myOS.elf **-serial pty &**

- 配合shell的串口输入输出

- 原来: **-serial stdio** //标准输入输出
- 改为: **-serial pty** //伪终端

tty: teletype
pty: pseudo-tty
pts: pseudo-terminal slave
是pty的实现方法

- 后台执行命令 &

- 将串口重定向到伪终端，运行时会告知具体是哪个
例如: /dev/pts/1

- 使用screen命令进入交互界面（与QEMU）

sudo screen /dev/pts/1

如果qemu命令没有后台执行，
可以另起一个Linux终端

4 验收标准



中国科学技术大学
University of Science and Technology of China

- 提交：源代码打包 + 实验报告；验收标准如下：
 - 完成源代码编写和调试，能够编译正确运行
 - 实现主流程，提供规定接口
 - 实现主要功能，提供规定接口
 - 将源代码进行合理的组织、提供相应的Makefile，能够生成myOS
 - 提供编译和运行脚本
 - 提交实验报告，实验报告中包括
 - 给出软件的框图，并加以概述
 - 详细说明主流程及其实现，画出流程图
 - 详细说明主要功能模块及其实现，画出流程图
 - 源代码说明（目录组织、Makefile组织）
 - 代码布局说明（地址空间）
 - 编译过程说明
 - 运行和运行结果说明
 - 遇到的问题和解决方案说明

演示



```
QEMU
Machine View
xlanchen >:
19:00:06
```

```
QEMU
Machine View
xlanchen >:_
Unknown interrupt1
19:00:32
```

```
xlanchen@DESKTOP-9IQ7F8: ~/workspace/OSlab/lab3
help
help
USAGE: help [cmd]

list all registered commands:
command name: description
cmd: list all registered commands
help: help [cmd]
xlanchen >:cmd
cmd
list all registered commands:
command name: description
cmd: list all registered commands
help: help [cmd]
xlanchen >:help cmd
help cmd
USAGE: help [cmd]

list all registered commands
xlanchen >:

QEMU
Machine View
xlanchen >:help
USAGE: help [cmd]

list all registered commands:
command name: description
cmd: list all registered commands
help: help [cmd]
xlanchen >:cmd
list all registered commands:
command name: description
cmd: list all registered commands
help: help [cmd]
xlanchen >:help cmd
USAGE: help [cmd]

list all registered commands
xlanchen >:_
Unknown interrupt1
19:06:29
```