# Project 2 细节问题

卢天越

2019/10/9

# Project 2 细节问题

- 提醒一些P2编写代码时需要注意的问题：
- 1、开关中断
- 2、syscall时的epc处理
- 3、汇编语言调用C语言函数
- 4、其他问题

# 1、开关中断

- Status寄存器
  - 龙芯2F处理器用户手册 5.10节

| 31 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 16 | 15 8 | 7 5 | 4 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CU (cu3:cu0) | 0 | FR | 0 | NO-FDIV | NO-FSQR | BEV | 0 | SR | 0 | IM7-IM0 | 0 | KSU | ERL | EXL | IE |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 4 | 8 | 3 | 2 | 1 | 1 | 1 |

图 5-11 Status 寄存器

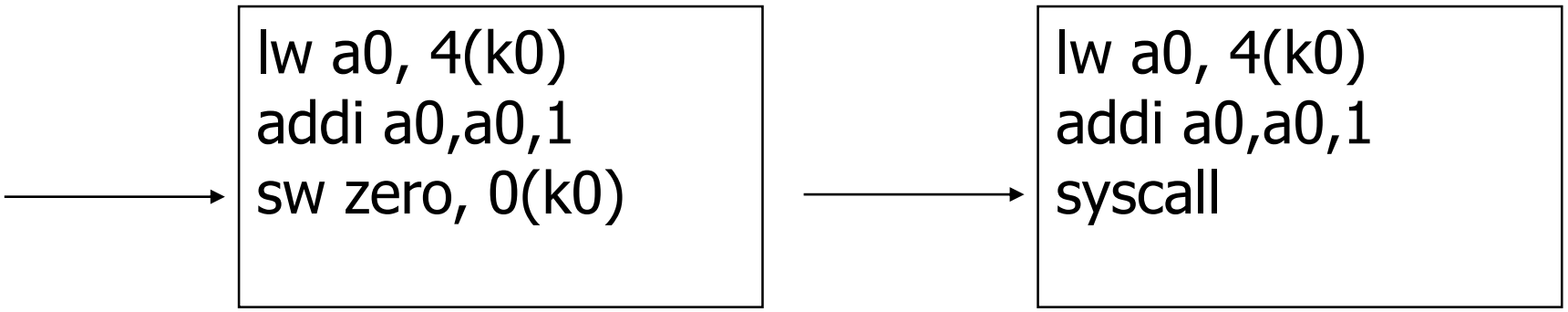
  - 中断使能：符合以下条件时：
    - ERL, EXL, IE：0,0,1
    - IM位的设置允许中断
  - 硬件自动置位EXL：
    - 发生例外时，自动置1；调用eret指令时，自动置0

# 1、开关中断

- 所以在MIPS架构下，开始和结束例外处理流程里面的开关中断是由硬件自动完成的
  - 与x86不同
- 注意：在初始化中手动开中断时，最后三位一定不要设成001，而要设成011，由eret指令来开中断。
- 否则：系统可能在eret指令之前就发生中断，导致不可预知的错误

# 2、syscall时的epc处理

- 系统发生例外时，认为当前指令未执行




- 但是syscall指令是执行完了之后才会发生例外，因为如果不对epc做任何处理，从syscall的例外处理回来之后，系统又会再执行一遍syscall指令，形成死循环

# 2、syscall时的epc处理

- 解决办法：
  - 当例外是syscall时，要将保存的user_context里面的epc加4
- 这部分在龙芯2F处理器手册的6.15节有介绍

- epc寄存器：系统在发生例外时会将发生例外时的PC寄存器的值放在epc里

# 3、汇编里面调用C语言函数

- gcc交叉编译器：gcc-4.3-ls232
  - 在传递参数时，要求堆栈保留参数的位置
- 例如这样的代码，反汇编之后会发现

```
// interrupt handler
// L3 exception Handler.
SAVE_CONTEXT(USER)
mfc0 a0, CP0_STATUS
mfc0 a1, CP0_CAUSE
jal interrupt_helper
RESTORE_CONTEXT(USER)
eret
```

```
a08014f4:    addiu   sp,sp,-32
a08014f8:    sw      ra,28(sp)
a08014fc:    sw      s8,24(sp)
a0801500:    move    s8,sp
a0801504:    sw      a0,32(s8)
// 原栈顶
a0801508:    sw      a1,36(s8)
// 原栈顶+4
```

# 3、汇编里面调用C语言函数

```
// interrupt handler
// L3 exception Handler.
SAVE_CONTEXT(USER)
mfc0 a0, CP0_STATUS
mfc0 a1, CP0_CAUSE
jal interrupt_helper
RESTORE_CONTEXT(USER)
eret
```

```
a08014f4:    addiu   sp,sp,-32
a08014f8:    sw      ra,28(sp)
a08014fc:    sw      s8,24(sp)
a0801500:    move    s8,sp
a0801504:    sw      a0,32(s8)
// 原栈顶
a0801508:    sw      a1,36(s8)
// 原栈顶+4
```

- 这样的后果导致中断处理会随机改写堆栈顶部的数据内容，产生随机的错误
- 解决办法：按编译器要求预留堆栈位置

# 3、汇编里面调用C语言函数

- 解决办法：按编译器要求预留堆栈位置
- 例如：在汇编里调用num个参数的C函数
  - 在前后增加对堆栈的操作

```
addiu  sp,sp, -4*num
jal  C_FUNCTION
addiu sp,sp, 4*num
```

# 4、关于检查时的问题

- 任务4之后打印不能出现乱码
  - init_screen里面要调用screen_clear
- 时间片要足够小
  - 板子的频率是300MHz，按一般的1ms一次中断处理，就是300000个cycle一次中断
  - 由于count寄存器是两个cycle加1，所以请同学们compare寄存器最大设置成150000
  - 由于QEMU模拟的频率较高，所以两边的中断时间会不一样
  - 调试的时候可以先改大了调

# 4、其他需要注意的问题

- 有的同学用gdb在eret指令的地方单步执行时可能gdb会死住。目前这好像是gdb或者QEMU的一个bug，暂时请大家不要在eret指令上单步执行
- P1遗留问题：
  - 没有对bss段清零导致初始化为0的全局变量是随机错误值。bss段：memsz-filesz的部分