浙江大学实验报告

课程名称：操作系统分析及实验 实验类型：综合型/设计性

实验项目名称：Lab 2: RV64 时钟中断处理

学生姓名：官泽隆 专业：计算机科学与技术 学号：3180103008

电子邮件地址：3180103008@zju.edu.cn 手机：18888910213

实验日期：2020 年 11 月 10 日

一、实验目的

学习在 RISC-V 上的异常处理相关机制，以时钟中断为例，编写时钟中断处理函数。

二、实验内容

改写head.S，初始化时钟中断使能，设置时钟中断委托；编写M-mode下时钟中断处理；完成S-mode下的时钟中断处理；利用Makefile完成对整个工程的管理。

三、主要仪器设备（必填）

宿主机操作系统：Windows 10 专业版 1903
Docker Desktop for Windows，基于WSL2
Image tarbull来源
https://magiclink.teambition.com/shares/5f6ee504674e61d264aa0ef8

四、操作方法和实验步骤

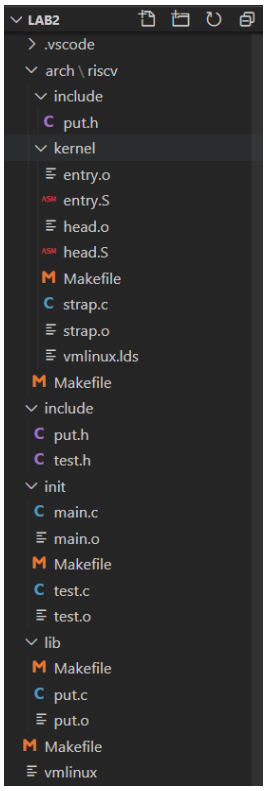
五、实验结果和分析

七、讨论、心得

4.1.1 建立映射

```
docker run -it -v ${pwd}:/home/oslab/lab2 -u oslab -w /home/oslab 6014
/bin/bash
```

4.1.2 目录结构



思考题

bss 段

看vmlinux没观察出来，但是根据 *Wiki* 的说法:

BSS（Block Started by Symbol）段，通常是指用来存放程序中未初始化的全局变量的一块内存区域。 静态变量、未显式初始化、在变量使用前由运行时初始化为零。

epc

同步异常就是mepc处的指令引发的，这条指令无法正常执行。因此在退出后肯定不能再执行这条指令，否则死循环。 本实验同步异常处理完退出应该给到ECALL下一条指令；

当处理中断时，mepc处的指令在正常的执行流程中，但被打断而没有执行完成，因此退出后应重新执行。

中断编程

按实验指导 4.2 - 4.4 一步步编码，即得。代码有注释对应的步骤。 其中，*strap.c* 中的 *print_timer_int()* 是C函数，不带参数无返回值，由 RISC-V汇编 *jal* 调用。

debug

本实验的程序不会自行退出，因此需要用QEMU来terminate:

```
During emulation, the following keys are useful:

When using -nographic, press 'ctrl-a h' to get some help.
```

没能一次跑成。第一次跑的现象是，只打印出第一行

```
ZJU OS LAB 2 3180103008
```

随后死循环。

先用 *objdump -d* 看反汇编，没头绪。 之后听实验指导建议，"如果觉得直接完成实验困难，可以先完成*machine mode*下的时钟中断处理"，即留在 entry.S 中的 trap_m0 函数。

这次跑的现象是，程序打印出第一次中断处理响应

```
[S] Supervisor Mode Timer Interrupt 0
```

后失去响应。

只能打印出第一次中断处理响应的现象很有指示性。 检查 *trap_m0* 中 *jal print_timer_int* 前后代码，尤其注意调用环境， 一定是自己在 *print_timer_int()* 后做了什么事情，导致程序执行不按预期，有自己想当然的assumption. 用 *objdump -d* 看*print_timer_int*反汇编，发现

汇编与C语言的接口

调用规范没仔细看，放在a*寄存器的存储被C函数摧毁了。 害，虽然计组里有学到过这事，可是自己编程时对寄存器的使用了如指掌，调用规范的印象也就不深。 从汇编到**C**进行编程时，汇编的变量应放到**s***寄存器或者栈上.

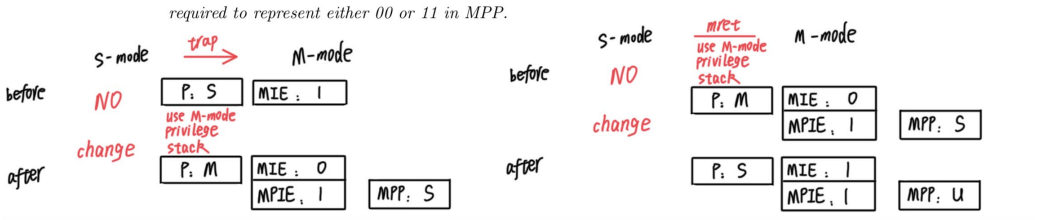
2-level stack

修完1个bug后，*machine mode*下的时钟中断处理正确了。但回到S-mode，仍然不对。 基于之前debug的观察，我想到可以用汇编给C函数准备参数，并调用C函数：

```
csrr a0, sstatus # dbg
jal puti
```

这让我对程序运行时刻的状态有所了解。我惊讶地发现 *j start_kernel* 前，sstatus的值还是0. 检查发现，

2-level stack 没搞清楚，对MPIE的作用对象产生了误解，误以为mret后SIE已经被打开。理解应该是这样的：



完成截图

```
qemu-system-riscv64: warning: No -bios option specified. Not loading a firmware.
qemu-system-riscv64: warning: This default will change in a future QEMU release. Please use
the -bios option to avoid breakages when this happens.
qemu-system-riscv64: warning: See QEMU's deprecation documentation for details.
ZJU OS LAB 2 3180103008
[S] Supervisor Mode Timer Interrupt 0
[S] Supervisor Mode Timer Interrupt 1
[S] Supervisor Mode Timer Interrupt 2
[S] Supervisor Mode Timer Interrupt 3
[S] Supervisor Mode Timer Interrupt 4
[S] Supervisor Mode Timer Interrupt 5
[S] Supervisor Mode Timer Interrupt 6
[S] Supervisor Mode Timer Interrupt 7
[S] Supervisor Mode Timer Interrupt 8
[S] Supervisor Mode Timer Interrupt 9
[S] Supervisor Mode Timer Interrupt 10
[S] Supervisor Mode Timer Interrupt 11
[S] Supervisor Mode Timer Interrupt 12
[S] Supervisor Mode Timer Interrupt 13
[S] Supervisor Mode Timer Interrupt 14
[S] Supervisor Mode Timer Interrupt 15
[S] Supervisor Mode Timer Interrupt 16
[S] Supervisor Mode Timer Interrupt 17
[S] Supervisor Mode Timer Interrupt 18
[S] Supervisor Mode Timer Interrupt 19
[S] Supervisor Mode Timer Interrupt 20
[S] Supervisor Mode Timer Interrupt 21
[S] Supervisor Mode Timer Interrupt 22
```

心得体会

推测：中断编程覆写了trap vector，调试器应该无法介入，因此务必小心编程

RISC-V工具链生成的反汇编代码还是比较清晰的，适合初学者学习；打通了硬件软件接口的模糊地带，带来自信

Makefile的问题：link的自动化程度不是很高