

CH3报告

实现了 `sys_task_info` 系统调用,该调用可以查询当前正在执行的任务信息(TCB,系统调用与调用次数,当前运行时长)

简答作业

1. 正确进入 U 态后,程序的特征还应有:使用 S 态特权指令,访问 S 态寄存器后会报错。请同学们可以自行测试这些内容(运行 [三个 bad 测例\(ch2b bad *.rs\)](#)),描述程序出错行为,同时注意注明你使用的 sbi 及其版本。
2. 深入理解 `trap.S` 中两个函数 `__alltraps` 和 `__restore` 的作用,并回答如下问题:
 1. L40: 刚进入 `__restore` 时, `a0` 代表了什么值。请指出 `__restore` 的两种使用情景。
 2. L43-L48: 这几行汇编代码特殊处理了哪些寄存器? 这些寄存器的值对于进入用户态有何意义? 请分别解释。

```
ld t0, 32*8(sp)
ld t1, 33*8(sp)
ld t2, 2*8(sp)
csrw sstatus, t0
csrw sepc, t1
csrw sscratch, t2
```

3. L50-L56: 为何跳过了 `x2` 和 `x4` ?

```
ld x1, 1*8(sp)
ld x3, 3*8(sp)
.set n, 5
.rept 27
    LOAD_GP %n
    .set n, n+1
.endr
```

4. L60: 该指令之后, `sp` 和 `sscratch` 中的值分别有什么意义?

```
csrrw sp, sscratch, sp
```

5. `__restore` : 中发生状态切换在哪一条指令? 为何该指令执行之后会进入用户态?
6. L13: 该指令之后, `sp` 和 `sscratch` 中的值分别有什么意义?

```
csrrw sp, sscratch, sp
```

7. 从 U 态进入 S 态是哪一条指令发生的?

答:

1.

```
1.[kernel] PageFault in application, bad addr = 0x0, bad instruction =
0x804003a4, kernel killed it.
2.[kernel] IllegalInstruction in application, kernel killed it.
3.[kernel] IllegalInstruction in application, kernel killed it.
1.程序出于U态, 访问地址非法, 触发异常 --物理内存保护 (PMP: Physical Memory
Protection) ?
2.ch2b_bad_instructions, 程序运行于U态, sret是S态指令, 故触发一次
3.ch2b_bad_register, 访问CSR寄存器, 程序处在U态, 触发异常
sbi: RustSBI version 0.4.0-alpha.1, adapting to RISC-V SBI v2.0.0
```

2

1. 根据riscv读本, 查看到a0保存的是函数参数, 我说实话, 我没在ch3的trap.S, `__restore`处看到a0, 所以, 我在这里回答ch2的~ a0: 进程context的指针
用处: 开始运行程序, 从S-mode返回U-mode
2. 恢复U-mode的CSR寄存器,sstatus: SPP 等字段给出 Trap 发生之前 CPU 处在哪个特权级 (S/U) 等信息;sepc: 当 Trap 是一个异常的时候, 记录 Trap 发生之前执行的最后一条指令的地址;sscratch:这个寄存器在用户态时, 保存的是内核栈地址, 这段代码恢复用户态的数据, 即: 内核栈。。。这很怪异

3

1. 应用不使用x4
2. 不保存 sp(x2), 因为它在第 9 行 后指向的是内核栈。

4

从sp->kernel stack, sscratch->user stack到sscratch->kernel stack, sp->user stack

5

sret

6

与4相反

7

用户态的ecall，或者中断与异常(这些，不知道该不该算指令，比如说，造成 segment fault的代码，在这个过程中也会导致cpu进入内核态？)

1. 在完成本次实验的过程（含此前学习的过程）中，我曾分别与 **以下各位** 就（与本次实验相关的）以下方面做过交流，还在代码中对应的位置以注释形式记录了具体的交流对象及内容：

暂无

2. 此外，我也参考了 **以下资料**，还在代码中对应的位置以注释形式记录了具体的参考来源及内容：

RISC-V 手册(一生一芯的翻译是riscv读本)

uc-osii的相关代码(寒假时看的) <https://github.com/weston-embedded/uC-OS2>

其他系统调用的实现（注释内包含了我当时的一些想法）

课程的实验文档

9月20多号看过xv6 的一部分文档与代码，但是应该与当前部分无关

3. 我独立完成了本次实验除以上方面之外的所有工作，包括代码与文档。我清楚地知道，从以上方面获得的信息在一定程度上降低了实验难度，可能会影响起评分。
4. 我从未使用过他人的代码，不管是原封不动地复制，还是经过了某些等价转换。我未曾也不会向他人（含此后各届同学）复制或公开我的实验代码，我有义务妥善保管好它们。我提交至本实验的评测系统的代码，均无意于破坏或妨碍任何计算机系统的正常运转。我清楚地知道，以上情况均为本课程纪律所禁止，若违反，对应的实验成绩将按“-100”分计。

你对本次实验设计及难度/工作量的看法

问答题很奇怪，是不是应该在CH2？

前两章亦有记下一些东西，看起来篇幅有限，就不放在这了~