Chapter 3

王拓为 2018011917

一、实现

当前正在执行的任务信息分为 3 部分：

- `status` 可以直接设为 `TaskStatus::Running`；
- `syscall_times` 可以通过在 `TaskControlBlock` 内增加记录系统调用的数组，在进入内核态系统调用异常处理函数之后，进入具体系统调用函数之前维护；
- `time` 可以通过在 `TaskControlBlock` 内增加记录任务首次调用时间的变量，在该任务首次被执行时维护；在调用 `sys_task_info` 时与当前时间做差输出；

二、问答

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，描述程序出错行为，同时注意注明你使用的 sbi 及其版本。

   sbi：RustSBI version 0.2.0-alpha.4

   - `ch2b_bad_address.rs`：

     > [ERROR] [kernel] PageFault in application, bad addr = 0x0, bad instruction = 0x8040008a, core dumped.

     程序尝试向 `0x0` 地址写入数据。

   - `ch2b_bad_instructions.rs`：

     > [ERROR] [kernel] IllegalInstruction in application, core dumped.

     程序尝试使用 S 态特权指令 `sret`。

   - `ch2b_bad_register.rs`：

     > [ERROR] [kernel] IllegalInstruction in application, core dumped.

     程序尝试访问 S 态寄存器 `sstatus`。

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` 的两种使用场景。

   `a0` 指向分配 Trap 上下文之后的内核栈栈顶。

   场景 1：通过 `__restore` 开始运行 app。

   场景 2：处理 Trap 后返回 U 态。

   （2）L46-L51：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？

   特殊处理了 `sstatus`、`sepc` 和 `sscratch`。

`sstatus` 中的 `SPP` 字段给出了 Trap 发生之前 CPU 处于哪一特权级，用于恢复用户态特权级；

`sepc` 中记录了 Trap 发生之前执行的最后一条指令的地址，作为返回地址；

`sscratch` 作为中转寄存器，进入用户态前指向用户栈栈顶。

（3）L53-L59：为何跳过了 `x2` 和 `x4` ？

跳过 `x2` 是因为我们要基于它来找到每个寄存器应该被保存到的正确位置；

跳过 `x4` 是因为除非手动处于一些特殊用途使用它，一般不会被用到。

（4）L63：该指令之后，`sp` 和 `sscratch` 中的值分别有什么意义?

`sp` 指向用户栈，`sscratch` 指向内核栈。

（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?

`sret` 之后发生状态切换。

因为该条指令会讲当前特权级按照 `sstatus` 的 `SPP` 字段设置为 U。

（6）L13：该指令之后，`sp` 和 `sscratch` 中的值分别有什么意义?

`sp` 指向内核栈，`sscratch` 指向用户栈。

（7）从 U 态进入 S 态是哪一条指令发生的?

在执行完一条触发 Trap 的指令（如 `ecall`）时。