操作系统 · Lab1

计01　容逸朗　2020010869

功能简述

- 本次实验主要实现了系统调用 `sys_task_info`。
  - 需要返回任务的运行状态、系统调用次数及运行时长；
  - 对于运行状态，由于查询的必定是当前执行中的任务，故返回 `TaskStatus::Running` 即可；
  - 系统调用次数可以在 syscall/mod.rs 中的 syscall 函数中维护，具体而言，就是把函数参数中的 syscall_id 对应的桶加一便可；
  - 运行时长需要通过 `run_next_task` 和 `run_first_task` 来记录，当上述函数被调用时，判断当前任务是否是首次被调用，若是则记录当前时间为任务开始时间，调用 `sys_task_info` 时把当前时间减去任务开始时间便可。

简答题

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 Rust 三个 bad 测例 (ch2b_bad_*.rs) ， 注意在编译时至少需要指定 `LOG=ERROR` 才能观察到内核的报错信息) ， 描述程序出错行为，同时注意注明你使用的 sbi 及其版本。

- 结果如下：

```
[kernel] PageFault in application, bad addr = 0x0, bad instruction = 0x80400414, kernel killed it.
[kernel] IllegalInstruction in application, kernel killed it.
[kernel] IllegalInstruction in application, kernel killed it.
```

- 使用了 RustSBI version 0.3.0-alpha.2, adapting to RISC-V SBI v1.0.0

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。

   答：`a0` 指向用户栈 TrapContent 所在的地址，`__restore` 的使用场景包括：第一次进入用户态和 Trap 处理完毕准备返回用户态。

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
```

答：处理了 sstatus, sepc 和 sscratch 寄存器，他们的意义如下：

- sstatus 记录了中断发生前的系统状态，如特权级、中断状态等；
- sepc 记录了异常发生的地址，或者是下一条执行指令的地址；
- sscratch 主要保存了内核栈的位置。

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
```

答：`x2` 是 `sp` ，保存在 `sscratch` 中需要使用 `csrr` 指令才可读出。`x4` 是 `tp` 寄存器，一般情况下用不到，因此无需保存，也就不需要恢复了。

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

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

答：`sp` 指向用户栈，而 `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？

答：`sret` ，这条指令会根据 `sstatus` 中 `SPP` 字段的设定更改当前系统特权级（此时一般为 U，即用户态），并跳转至 `sepc` 指向的指令继续执行程序。

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

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

答：`sp` 指向内核栈，而 `sscratch` 指向用户栈。

7. 从 U 态进入 S 态是哪一条指令发生的?

   答：`ecall` 。

Honor Code

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：

   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：

   rCore-Tutorial-Guide 2023 春季学期 的第二、三部分。

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。