TaskInfo 系统调用实现总结

1.核心功能： 添加新的系统调用*sys_task_info*，用于获取当前任务的运行信息
2.实现要点：

- 在TaskControlBlock中增加系统调用计数和首次调度时间字段
- 在任务首次调度时记录时间戳
- 在系统给调用处理时统计调用次数

此实现完成了任务信息查询功能，支持获取任务状态、系统调用统计和运行时间信息。

问答题

使用的SBI版本: RustSBI-QEMU Version 0.2.0-alpha.3
1. `ch2b_bad_instructions.rs` (访问地址0x0) 错误日志： “[kernel] PageFault in application, bad addr = 0x0, bad instruction = 0x804003a4, kernel killed it.”
写0地址，是一个非法的内存访问，触发PageFault
2. `ch2b_bad_register.rs` (读取sstatus寄存器) 错误日志： “[kernel] IllegalInstruction in application, kernel killed it.”程序访问特权寄存器，触发IllegalInstruction
3. `ch2b_bad_instructions.rs` (执行sret指令) 错误日志： “[kernel] IllegalInstruction in application, kernel killed it.”程序访问特权寄存器，触发IllegalInstruction

Trap.S深入理解

1. __restore的a0含义与使用场景：

- 场景1：首次运行应用程序，a0包含用户程序的Trap上下文
- 场景2：Trap处理完返回，a0包含内核栈上保存的Trap上下文

2. 特殊寄存器处理：

    - sstatus：控制特权级状态，决定返回模式
    - sepc：保存返回地址，确定恢复执行位置
    - sscratch：保存用户栈指针，用于特权级切换

3.跳过x2和x4原因：

- x2(sp)：需要特殊处理栈切换
- x4(tp)：用户程序不使用，可跳过

4.csrrw后的值：

- sp：指向用户栈
- sscratch：指向内核栈

5.状态切换：

- 在sret指令发生
- 通过设置sstatus.SPP实现特权级切换

6. L13后的值：

    - sp：指向内核栈
    - sscratch：指向用户栈

7.U态到S态切换：

* 通过*ecall*指令触发*Trap*进入*S*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 **无** 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：
2. 此外，我也参考了 **无** ，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：
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。