Chapter3报告

实现的功能

主要实现的功能即`sys_task_info`函数获取当前任务信息，其中主要包括

1. 在`TaskControlBlock`中添加task需要记录的信息域（注意爆栈错误）
2. 在`TaskManage`r的函数中添加对task信息的更新和维护
3. 添加取出当前任务信息的函数

问答题

1.

运行结果报错内容如下：

```
[ERROR] [kernel] PageFault in application, bad addr = 0x0, bad instruction = 0x80400410, core dumped.
[ERROR] [kernel] IllegalInstruction in application, core dumped.
[ERROR] [kernel] IllegalInstruction in application, core dumped.
```

即`ch2b_bad_address`程序触发了`PageFault`错误，`ch2b_bad_instriction`和`ch2b_bad_register`触发了`IllegalInstruction`错误

使用的sbi及版本为：`RustSBI version 0.3.0-alpha.4`

2.

深入理解`trap.S`

（1）刚进入`__restore`时，`a0`代表了内核栈栈顶（指向要恢复的`TrapContex`）。两个使用场景分别是：使用`__restore`开始运行app；使用`__restore`在处理好trap后返回到用户态继续运行app

（2）这几行代码特殊处理了`sstatus`，`sepc`，`ssratch`寄存器。它们的意义和作用为`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。

`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。

`sscratch` ：指向分配这个 `TrapContext` 之前的内核栈栈顶，恢复到进入 Trap 之前的状态。

（3）因为`x2`即`sp`，用户栈的栈顶已经使用 `sscratch` 进行了特殊的保存和恢复，目前`sp`指向内核栈栈顶。

而`x4`即`tp`，除非我们手动出于一些特殊用途使用它，否则一般不会被用到，无需保存。

（4）交换 sscratch 和 sp，现在 sp 重新指向用户栈栈顶，sscratch 也依然保存 进入 Trap 之前的状态并指向内核栈栈顶。

（5）发生在最后一行，使用 `sret`
指令回到 U 特权级继续运行应用程序控制流。会进入用户态的原因是恢复了`sstatus`的`SPP`字段为U态。

（6） `sp`指向内核栈栈顶，`sscratch`指向用户栈栈顶。

（7）CPU 在 U 特权级运行用户程序的时候触发 Trap时， CPU 会跳转到 `stvec` 所设置的 Trap 处理入口地址，并将当前特权级设置为 S ，然后从Trap 处理入口地址处开始执行。 具体发生的指令可能是中断、系统调用、出错指令。