# lab1

## 实现功能总结

在`TaskControlBlock`中

1. 添加`task_stime`记录任务第一次开始执行时间
2. 添加`task_syscall_times`记录当前任务各个系统调用次数

在`task`模块中

1. 添加`task_syscall_times`函数，用于统计不同系统调用的调用次数
2. 添加`current_task_info`函数，用于返回当前任务系统调用次数和开始时间

之后在`syscall`模块中调用`task_syscall_times`方法统计系统调用次数，在task模块中的任务切换函数中，记录任务第一次开始调用时间。

完成任务信息记录之后，在`sys_task_info`函数中调用`current_task_info`方法，返回任务信息，完成任务信息的系统调用。

## 简答作业

### 第一题

> 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。

> RustSBI version 0.4.0-alpha.1

```
[kernel] Loading app_0
[kernel] PageFault in application, kernel killed it.
[kernel] Loading app_1
[kernel] IllegalInstruction in application, kernel killed it.
[kernel] Loading app_2
[kernel] IllegalInstruction in application, kernel killed it.
```

由于存在非法操作，qemu报出异常，进入异常处理函数`trap_handler`，由于是非法异常，无法处理，所以输出异常信息，退出该线程，继续进入下一线程执行。

### 第二题

> 深入理解 trap.S 中两个函数`__alltraps`和`__restore`的作用，并回答如下问题:

## 第一问

> L40：刚进入 `__restore` 时，`a0` 代表了什么值。请指出 `__restore` 的两种使用情景。

`a0` 代表 `trap_handler` 函数的返回值

1. 线程切换，恢复下一线程的上下文，并从s态退出，进入下一线程开始执行
2. 从系统调用中恢复，恢复系统调用之前的上下文，并从s态退出，完成异常处理，继续执行

## 第二问

> 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。

```
1 ld t0, 32*8(sp)
2 ld t1, 33*8(sp)
3 ld t2, 2*8(sp)
4 csrw sstatus, t0
5 csrw sepc, t1
6 csrw sscratch, t2
```

**`sstatus` 寄存器**

保存和控制cpu当前的状态的CSR寄存器，主要用于保存中断和特权模式的状态和使能中断

**`sepc` 寄存器**

保存异常发生地址，用于异常处理完成后退出

**`sscratch` 寄存器**

主要用于在异常处理中保存临时数据，在rcore中，它被用于保存栈指针地址

## 第三问

> L50-L56：为何跳过了 `x2` 和 `x4`？

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

`x2` 寄存器是 `sp` 寄存器，目前还指向用户态的上下文，不能被恢复，之后会被恢复

`x4` 寄存器是 `tp` 寄存器，之前没有被保存，因此不会被恢复

### 第四问

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

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

执行该指令， `sp` 寄存器和 `sscratch` 寄存器交换，交换之后， `sp` 保存进入异常处理之前的用户栈指针地址， `sscratch` 保存内核栈保存上下文栈指针地址

### 第五问

> `__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？

发生在 `sret` 指令，执行该指令，会跳转到 `sepc` 寄存器保存的 `pc` 地址，并且恢复 `sstatus` 寄存器到中断发生前的状态（用户态）

### 第六问

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

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

执行这条指令， `sp` 寄存器和 `sscratch` 寄存器中的值会被交换，交换之后， `sscratch` 保存用户栈的指针地址， `sp` 保存栈中上下文的指针地址

### 第七问

> 从 U 态进入 S 态是哪一条指令发生的？

`ecall` 指令

# 荣誉准则

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：

>

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：

>

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。