Lab 2: RV64 内核线程调度

姓名：高铭健

学号：3210102322

日期：2023.11.4

一、实验目的和要求

1.1 实验目的：

- 了解线程概念，并学习线程相关结构体，并实现线程的初始化功能。
- 了解如何使用时钟中断来实现线程的调度。
- 了解线程切换原理，并实现线程的切换。
- 掌握简单的线程调度算法，并完成两种简单调度算法的实现。

1.2 实验环境：

- 计算机（Intel Core i5以上，4GB内存以上）系统
- Ubuntu 22.04.2 LTS

二、实验过程和数据记录

2.1 准备工程：

- 从 `repo` 同步以下代码: `rand.h/rand.c`， `string.h/string.c`， `mm.h/mm.c`， `proc.h/proc.c`， `test.h/test_schedule.h`， `schedule_test.c` ，同步后代码结构如下：

```
fengnian@MICROSO-VU1H310:/mnt/d/study/OS/os23fall-stu/src/lab2$ tree
.
├── Makefile
├── arch
│   └── riscv
│       ├── Makefile
│       ├── include
│       │   ├── clock.h
│       │   ├── defs.h
│       │   ├── mm.h
│       │   ├── proc.h
│       │   └── sbi.h
│       └── kernel
│           ├── Makefile
│           ├── clock.c
│           ├── entry.S
│           ├── head.S
│           ├── mm.c
│           ├── proc.c
│           ├── sbi.c
│           ├── trap.c
│           └── vmlinux.lds
├── include
│   ├── print.h
│   ├── printk.h
│   ├── rand.h
│   ├── schedule_test.h
│   ├── stddef.h
│   ├── string.h
│   ├── test.h
│   └── types.h
├── init
│   ├── Makefile
│   ├── main.c
│   └── test.c
├── lib
│   ├── Makefile
│   ├── print.c
│   ├── printk.c
│   ├── rand.c
│   ├── schedule_test.c
│   └── string.c
└── test
    ├── Makefile
    └── schedule_test.c
```

- 在 `_start` 中调用 `mm_init` （`jal mm_init`），来初始化内存管理系统

```
arch > riscv > kernel > head.S
.extern start_kernel

    .section .text.init
    .globl _start
_start:
    #将栈顶指针放入sp
    la sp, boot_stack_top
    jal mm_init
```

- 在初始化时用一些自定义的宏，需修改 `defs.h`，在 `defs.h` 添加如下内容：

```
#define PHY_START 0x0000000080000000
#define PHY_SIZE  128 * 1024 * 1024 // 128MB， QEMU 默认内存大小
#define PHY_END   (PHY_START + PHY_SIZE)

#define PGSIZE 0x1000 // 4KB
#define PGROUNDUP(addr) ((addr + PGSIZE - 1) & (~(PGSIZE - 1)))
#define PGROUNDDOWN(addr) (addr & (~(PGSIZE - 1)))
```

2.2 线程调度功能实现

2.2.1 线程初始化

- 为 `idle` 设置 `task_struct`。并将 `current`, `task[0]` 都指向 `idle`, 代码如下所示:

```c
    // 1. 调用 kalloc() 为 idle 分配一个物理页
    idle = kalloc();
    // 2. 设置 state 为 TASK_RUNNING;
    idle->state = TASK_RUNNING;
    // 3. 由于 idle 不参与调度 可以将其 counter / priority 设置为 0
    idle->counter = 0;
    idle->priority = 0;
    // 4. 设置 idle 的 pid 为 0
    idle->pid = 0;
    // 5. 将 current 和 task[0] 指向 idle
    current = idle;
    task[0] = idle;
```

- 将 `task[1]` ~ `task[NR_TASKS - 1]` 全部初始化
    - 调用 `kalloc()` 为 `task[i]` 分配一个物理页
    - 每个线程的 `state` 为 `TASK_RUNNING`
    - `ra` 设置为 `__dummy` 的地址, `sp` 设置为 该线程申请的物理页的高地址

```c
for(int i = 1 ; i < NR_TASKS ; i++){
        task[i]  = kalloc();
        task[i]->pid = i;
        task[i]->counter = task_test_counter[i];
        task[i]->priority = task_test_priority[i];
        task[i]->state = TASK_RUNNING;
        uint64 ra_address = &__dummy;
        uint64 sp_address = (uint64)task[i] + PGSIZE;
        task[i]->thread.ra = ra_address;
        task[i]->thread.sp = sp_address;
    }
```

- 在 `_start` 中调用 `task_init`

```asm
.extern start_kernel

    .section .text.init
    .globl _start
_start:
    #将栈顶指针放入sp
    la sp, boot_stack_top
    jal mm_init
    jal task_init
```

2.2.2 `__dummy` 与 `dummy`

- 在 `proc.c` 添加 `dummy()`:

```
void dummy() {
    schedule_test();
    uint64 MOD = 1000000007;
    uint64 auto_inc_local_var = 0;
    int last_counter = -1;
    while(1) {
        if ((last_counter == -1 || current->counter != last_counter) && current->counter > 0) {
            if(current->counter == 1){
                --(current->counter);   // forced the counter to be zero if this thread is going to be scheduled
            }                           // in case that the new counter is also 1, leading the information not printed.
            last_counter = current->counter;
            auto_inc_local_var = (auto_inc_local_var + 1) % MOD;
            printk("[PID = %d] is running. auto_inc_local_var = %d\n", current->pid, auto_inc_local_var);
        }
    }
}
```

- 为线程 **第一次调度** 提供返回函数 `__dummy`，在 `entry.S` 添加 `__dummy`
    - 在 `__dummy` 中将 sepc 设置为 `dummy()` 的地址
    - 使用 `sret` 从中断中返回。

```
__dummy:
    # YOUR CODE HERE
    la t0, dummy
    csrw sepc, t0 #在__dummy 中将 sepc 设置为 dummy() 的地址
    sret #从中断中返回
```

2.2.3 实现线程切换

- 判断下一个执行的线程 `next` 与当前的线程 `current` 是否为同一个线程，如果是同一个线程，则无需做任何处理，否则调用 `__switch_to` 进行线程切换。

```c
void switch_to(struct task_struct* next) {
    /* YOUR CODE HERE */
    //判断下一个执行的线程 next 与当前的线程 current 是否为同一个线程，如果是同一个线程，则无需做任何处理，否则调用 __switch_to 进行线程切换。
    if(current->pid != next->pid){
        struct task_struct*tmp = current;
        current = next ;
        __switch_to(tmp, next);
    }
}
```

- 在 `entry.S` 中实现线程上下文切换 `__switch_to`:
  - `__switch_to` 接受两个 `task_struct` 指针作为参数，分别在 `a0` 和 `a1` 中
  - 保存当前线程的 `ra`， `sp`， `s0~s11` 到当前线程的 `thread_struct` （偏移量为 `40`，因为 `thread_struct` 有5个成员变量）
  - 将下一个线程的 `thread_struct` 中的相关数据载入到 `ra`， `sp`， `s0~s11` 中

```asm
__switch_to:
    # save state to prev process
    # YOUR CODE HERE
    addi t0, a0, 40
    sd ra, 8(t0)
    sd sp, 16(t0)
    sd s0, 24(t0)
    sd s1, 32(t0)
    sd s2, 40(t0)
    sd s3, 48(t0)
    sd s4, 56(t0)
    sd s5, 64(t0)
    sd s6, 72(t0)
    sd s7, 80(t0)
    sd s8, 88(t0)
    sd s9, 96(t0)
    sd s10, 104(t0)
    sd s11, 112(t0)
    # restore state from next process
    # YOUR CODE HERE
    addi t0, a1, 40
    ld ra, 8(t0)
    ld sp, 16(t0)
    ld s0, 24(t0)
    ld s1, 32(t0)
    ld s2, 40(t0)
    ld s3, 48(t0)
    ld s4, 56(t0)
    ld s5, 64(t0)
    ld s6, 72(t0)
    ld s7, 80(t0)
    ld s8, 88(t0)
```

```
    ld s9, 96(t0)
    ld s10, 104(t0)
    ld s11, 112(t0)

    ret
```

2.2.4 实现调度入口函数

实现 `do_timer()`，并在 `时钟中断处理函数` 中调用。

- 如果当前线程是 idle 线程 直接进行调度
- 如果当前线程不是 idle 对当前线程的运行剩余时间减1 若剩余时间仍然大于0 则直接返回 否则进行调度

```c
void do_timer(void) {
    if(current == idle){
        schedule();
    }
    else{
        current->counter--;
        if(current->counter){
            return;
        }
        else{
            schedule();
        }
    }
}
```

```c
void trap_handler(unsigned long scause, unsigned long sepc) {
    // 通过 `scause` 判断trap类型
    // 如果是interrupt 判断是否是timer interrupt
    // 如果是timer interrupt 则打印输出相关信息, 并通过 `clock_set_next_event()` 设置下一次时钟中断
    // `clock_set_next_event()` 见 4.3.4 节
    // 其他interrupt / exception 可以直接忽略
    if(scause & 0x8000000000000000){//trap 类型为interrupt
        if(scause == 0x8000000000000005){//timer interrupt
            printk("[S] Supervisor Mode Timer Interrupt\n");
            clock_set_next_event();
            do_timer();
        }
    }
    else{

    }
}
```

2.2.5 实现线程调度

2.2.5.1 短作业优先调度算法

- 遍历线程指针数组`task`（不包括`idle`，即`task[0]`），在所有运行状态（`TASK_RUNNING`）下的线程运行剩余时间最少的线程作为下一个执行的线程。
- 如果所有运行状态下的线程运行剩余时间都为0，则对`task[1]` ~ `task[NR_TASKS-1]`的运行剩余时间重新赋值（使用`rand()`），之后再重新进行调度。
- 打印设置和switch是的`pid`和`counter`信息

```
    int judge_all_0 = 1;
    int select_task = 0;
    for(int i = 1 ; i < NR_TASKS ; i++){
        if(task[i]->counter){
            judge_all_0 = 0;
        }
        if((task[i]->counter < task[select_task]->counter || select_task == 0) && task[i]->state == TASK_RUNNING && task[i]->counter){
            select_task = i;
        }
    }
    if(judge_all_0 == 1){
        for(int i = 0 ; i < NR_TASKS ; i++){
            task[i]->counter = rand();
            printk("SET [PID = %d COUNTER = %d]\n",task[i]->pid,task[i]->counter);
        }
        schedule();
    }
    else{
        printk("\nswitch to [PID = %d COUNTER = %d]\n",task[select_task]->pid,task[select_task]->counter);
        switch_to(task[select_task]);
    }
```

2.2.5.2 优先级调度算法

- 遍历线程指针数组`task`（不包括`idle`，即`task[0]`），第一轮在所有运行状态（`TASK_RUNNING`）下的线程运行剩余时间最多的线程作为下一个执行的线程，如果时间相同，选择优先级更高的线程
- 第二轮所有运行状态下的线程运行剩余时间都为0，将`(task[i]->counter >> 1) +task[i]->priority`的结果作为新的时间片，也就是`priority`与`counter`一致，再选择剩余时间最多的线程作为下一个执行的线程就相当于按优先级调度
- 打印设置和switch是的`pid`和`counter`信息

```
 int judge_all_0 = 1;
    int select_task = 0;
    for( ; ; ){
        for(int i = NR_TASKS-1 ; i >=1  ; i--){
```

```
            if(task[i]->counter){
                judge_all_0 = 0;
            }
            if((task[i]->counter > task[select_task]->counter || select_task == 0) && task[i]->state == TASK_RUNNING && task[i]->counter){
                select_task = i;
            }
        }
        if(select_task){
            break;
        }
        for(int i = NR_TASKS-1 ; i >=1  ; i--){
            if (task[i]){
                task[i]->counter = (task[i]->counter >> 1) +task[i]->priority;
            }
        }
    }
    if(judge_all_0 == 1){
        for(int i = 0 ; i < NR_TASKS ; i++){
            task[i]->counter = rand();
            printk("SET [PID = %d COUNTER = %d]\n",task[i]->pid,task[i]->counter);
        }
        schedule();
    }
    else{
        printk("\nswitch to [PID = %d COUNTER = %d PRIORITY = %d]\n",task[select_task]->pid,task[select_task]->counter,task[select_task]->priority);
        switch_to(task[select_task]);
    }
```

2.3 编译及测试

- 在`proc.c`中使用`#ifdef`，`#endif`来控制代码，用`#ifdef SJF`，`#else`实现编译时的代码选择，完整的`shedule`函数如下：

```
void schedule(void) {
#ifdef SJF
    int judge_all_0 = 1;
    int select_task = 0;
    for(int i = 1 ; i < NR_TASKS ; i++){
        if(task[i]->counter){
            judge_all_0 = 0;
        }
        if((task[i]->counter < task[select_task]->counter || select_task == 0) && task[i]->state == TASK_RUNNING && task[i]->counter){
            select_task = i;
        }
```

```
    }
    if(judge_all_0 == 1){
        for(int i = 0 ; i < NR_TASKS ; i++){
            task[i]->counter = rand();
            printk("SET [PID = %d COUNTER = %d]\n",task[i]->pid,task[i]->counter);
        }
        schedule();
    }
    else{
        printk("\nswitch to [PID = %d COUNTER = %d]\n",task[select_task]->pid,task[select_task]->counter);
        switch_to(task[select_task]);
    }
#else
    int judge_all_0 = 1;
    int select_task = 0;
    for(int i = 1 ; i < NR_TASKS ; i++){
        if(task[i]->counter){
            judge_all_0 = 0;
        }
        if((task[i]->counter > task[select_task]->counter || select_task == 0) && task[i]->state == TASK_RUNNING && task[i]->counter){
            select_task = i;
        }
        if(task[i]->counter == task[select_task]->counter && task[i]->priority > task[select_task]->priority){
            select_task = i;
        }
    }
    if(judge_all_0 == 1){
        for(int i = 0 ; i < NR_TASKS ; i++){
            task[i]->counter = rand();
            printk("SET [PID = %d COUNTER = %d]\n",task[i]->pid,task[i]->counter);
        }
        schedule();
    }
    else{
        printk("\nswitch to [PID = %d COUNTER = %d PRIORITY = %d]\n",task[select_task]->pid,task[select_task]->counter,task[select_task]->priority);
        switch_to(task[select_task]);
    }
#endif
}
```

- `NR_TASKS = 4` 情况下短作业优先调度算法的测试结果：

  先修改`makefile`：

```
CFLAG = ${CF} ${INCLUDE} -D SJF
```

```
Boot HART MIDELEG          : 0x0000000000000222
Boot HART MEDELEG          : 0x000000000000b109
...mm_init done!
...proc_init done!
2022 Hello RISC-V
...proc_init done!
[S] Supervisor Mode Timer Interrupt

switch to [PID = 1 COUNTER = 4]
B
B[S] Supervisor Mode Timer Interrupt
B
BB[S] Supervisor Mode Timer Interrupt
B
BBB[S] Supervisor Mode Timer Interrupt
B
BBBB[S] Supervisor Mode Timer Interrupt

switch to [PID = 3 COUNTER = 8]
D
BBBBD[S] Supervisor Mode Timer Interrupt
D
BBBBDD[S] Supervisor Mode Timer Interrupt
D
BBBBDDD[S] Supervisor Mode Timer Interrupt
D
BBBBDDDD[S] Supervisor Mode Timer Interrupt
D
BBBBDDDDD[S] Supervisor Mode Timer Interrupt
D
BBBBDDDDDD[S] Supervisor Mode Timer Interrupt
D
BBBBDDDDDDD[S] Supervisor Mode Timer Interrupt
```

```
D
BBBBDDDDDDD[S] Supervisor Mode Timer Interrupt
D
BBBBDDDDDDDD[S] Supervisor Mode Timer Interrupt
D
BBBBDDDDDDDDD[S] Supervisor Mode Timer Interrupt

switch to [PID = 2 COUNTER = 9]
C
BBBBDDDDDDDDDC[S] Supervisor Mode Timer Interrupt
C
BBBBDDDDDDDDDCC[S] Supervisor Mode Timer Interrupt
C
BBBBDDDDDDDDDCCC[S] Supervisor Mode Timer Interrupt
C
BBBBDDDDDDDDDCCCC[S] Supervisor Mode Timer Interrupt
C
BBBBDDDDDDDDDCCCCC[S] Supervisor Mode Timer Interrupt
C
BBBBDDDDDDDDDCCCCCC[S] Supervisor Mode Timer Interrupt
C
BBBBDDDDDDDDDCCCCCCC[S] Supervisor Mode Timer Interrupt
C
BBBBDDDDDDDDDCCCCCCCC[S] Supervisor Mode Timer Interrupt
C
BBBBDDDDDDDDDCCCCCCCCC
NR_TASKS = 4, SJF test passed!

[S] Supervisor Mode Timer Interrupt
SET [PID = 0 COUNTER = 1]
SET [PID = 1 COUNTER = 4]
SET [PID = 2 COUNTER = 10]
SET [PID = 3 COUNTER = 4]
```

测试通过

- `NR_TASKS = 4` 情况下优先级调度算法的测试结果

```
Boot HART MIDELEG         : 0x0000000000000222
Boot HART MEDELEG         : 0x000000000000b109
...mm_init done!
...proc_init done!
2022 Hello RISC-V
...proc_init done!
[S] Supervisor Mode Timer Interrupt

switch to [PID = 2 COUNTER = 9 PRIORITY = 88]
C
C[S] Supervisor Mode Timer Interrupt
C
CC[S] Supervisor Mode Timer Interrupt
C
CCC[S] Supervisor Mode Timer Interrupt
C
CCCC[S] Supervisor Mode Timer Interrupt
C
CCCCC[S] Supervisor Mode Timer Interrupt
C
CCCCCC[S] Supervisor Mode Timer Interrupt
C
CCCCCCC[S] Supervisor Mode Timer Interrupt
C
CCCCCCCC[S] Supervisor Mode Timer Interrupt
C
CCCCCCCCC[S] Supervisor Mode Timer Interrupt

switch to [PID = 3 COUNTER = 8 PRIORITY = 52]
D
CCCCCCCCCD[S] Supervisor Mode Timer Interrupt
D
```

```
CCCCCCCCCD[S] Supervisor Mode Timer Interrupt
D
CCCCCCCCCDD[S] Supervisor Mode Timer Interrupt
D
CCCCCCCCCDDD[S] Supervisor Mode Timer Interrupt
D
CCCCCCCCCDDDD[S] Supervisor Mode Timer Interrupt
D
CCCCCCCCCDDDDD[S] Supervisor Mode Timer Interrupt
D
CCCCCCCCCDDDDDD[S] Supervisor Mode Timer Interrupt
D
CCCCCCCCCDDDDDDD[S] Supervisor Mode Timer Interrupt
D
CCCCCCCCCDDDDDDDD[S] Supervisor Mode Timer Interrupt

switch to [PID = 1 COUNTER = 4 PRIORITY = 37]
B
CCCCCCCCCDDDDDDDDB[S] Supervisor Mode Timer Interrupt
B
CCCCCCCCCDDDDDDDDBB[S] Supervisor Mode Timer Interrupt
B
CCCCCCCCCDDDDDDDDBBB[S] Supervisor Mode Timer Interrupt
B
CCCCCCCCCDDDDDDDDBBBB
NR_TASKS = 4, PRIORITY test passed!

[S] Supervisor Mode Timer Interrupt
SET [PID = 0 COUNTER = 1]
SET [PID = 1 COUNTER = 4]
SET [PID = 2 COUNTER = 10]
SET [PID = 3 COUNTER = 4]
```

测试通过

- `NR_TASKS = 16` 情况下短作业优先调度算法的测试结果：

```
Boot HART MIDELEG         : 0x0000000000000222
Boot HART MEDELEG         : 0x000000000000b109
...mm_init done!
...proc_init done!
2022 Hello RISC-V
...proc_init done!
[S] Supervisor Mode Timer Interrupt

switch to [PID = 8 COUNTER = 1]
I
I[S] Supervisor Mode Timer Interrupt

switch to [PID = 7 COUNTER = 2]
H
IH[S] Supervisor Mode Timer Interrupt
H
IHH[S] Supervisor Mode Timer Interrupt

switch to [PID = 14 COUNTER = 2]
O
IHHO[S] Supervisor Mode Timer Interrupt
O
IHHOO[S] Supervisor Mode Timer Interrupt

switch to [PID = 11 COUNTER = 3]
L
IHHOOL[S] Supervisor Mode Timer Interrupt
L
IHHOOLL[S] Supervisor Mode Timer Interrupt
```

```
L
IHHOOLL[S] Supervisor Mode Timer Interrupt
L
IHHOOLLL[S] Supervisor Mode Timer Interrupt

switch to [PID = 13 COUNTER = 3]
N
IHHOOLLLN[S] Supervisor Mode Timer Interrupt
N
IHHOOLLLNN[S] Supervisor Mode Timer Interrupt
N
IHHOOLLLNNN[S] Supervisor Mode Timer Interrupt

switch to [PID = 1 COUNTER = 4]
B
IHHOOLLLNNNB[S] Supervisor Mode Timer Interrupt
B
IHHOOLLLNNNBB[S] Supervisor Mode Timer Interrupt
B
IHHOOLLLNNNBBB[S] Supervisor Mode Timer Interrupt
B
IHHOOLLLNNNBBBB[S] Supervisor Mode Timer Interrupt
```

```
Boot HART MIDELEG         : 0x0000000000000222
Boot HART MEDELEG         : 0x000000000000b109
...mm_init done!
...proc_init done!
2022 Hello RISC-V
...proc_init done!
[S] Supervisor Mode Timer Interrupt

switch to [PID = 8 COUNTER = 1]
I
I[S] Supervisor Mode Timer Interrupt

switch to [PID = 7 COUNTER = 2]
H
IH[S] Supervisor Mode Timer Interrupt
H
IHH[S] Supervisor Mode Timer Interrupt

switch to [PID = 14 COUNTER = 2]
O
IHHO[S] Supervisor Mode Timer Interrupt
O
IHHOO[S] Supervisor Mode Timer Interrupt

switch to [PID = 11 COUNTER = 3]
L
IHHOOL[S] Supervisor Mode Timer Interrupt
L
IHHOOLL[S] Supervisor Mode Timer Interrupt
```

```
L
IHHOOLL[S] Supervisor Mode Timer Interrupt
L
IHHOOLLL[S] Supervisor Mode Timer Interrupt

switch to [PID = 13 COUNTER = 3]
N
IHHOOLLLN[S] Supervisor Mode Timer Interrupt
N
IHHOOLLLNN[S] Supervisor Mode Timer Interrupt
N
IHHOOLLLNNN[S] Supervisor Mode Timer Interrupt

switch to [PID = 1 COUNTER = 4]
B
IHHOOLLLNNNB[S] Supervisor Mode Timer Interrupt
B
IHHOOLLLNNNBB[S] Supervisor Mode Timer Interrupt
B
IHHOOLLLNNNBBB[S] Supervisor Mode Timer Interrupt
B
IHHOOLLLNNNBBBB[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 12 COUNTER = 4]
M
IHHOOLLLNNNBBBBM[S] Supervisor Mode Timer Interrupt
M
IHHOOLLLNNNBBBBMM[S] Supervisor Mode Timer Interrupt
M
IHHOOLLLNNNBBBBMMM[S] Supervisor Mode Timer Interrupt
M
IHHOOLLLNNNBBBBMMMM[S] Supervisor Mode Timer Interrupt

switch to [PID = 4 COUNTER = 5]
E
IHHOOLLLNNNBBBBMMMME[S] Supervisor Mode Timer Interrupt
E
IHHOOLLLNNNBBBBMMMMEE[S] Supervisor Mode Timer Interrupt
E
IHHOOLLLNNNBBBBMMMMEEE[S] Supervisor Mode Timer Interrupt
E
IHHOOLLLNNNBBBBMMMMEEEE[S] Supervisor Mode Timer Interrupt
E
IHHOOLLLNNNBBBBMMMMEEEEE[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 15 COUNTER = 6]
P
IHHOOLLLNNNBBBBMMMMEEEEEP[S] Supervisor Mode Timer Interrupt
P
IHHOOLLLNNNBBBBMMMMEEEEEPP[S] Supervisor Mode Timer Interrupt
P
IHHOOLLLNNNBBBBMMMMEEEEEPPP[S] Supervisor Mode Timer Interrupt
P
IHHOOLLLNNNBBBBMMMMEEEEEPPPP[S] Supervisor Mode Timer Interrupt
P
IHHOOLLLNNNBBBBMMMMEEEEEPPPPP[S] Supervisor Mode Timer Interrupt
P
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPP[S] Supervisor Mode Timer Interrupt

switch to [PID = 9 COUNTER = 7]
J
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJ[S] Supervisor Mode Timer Interrupt
J
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJ[S] Supervisor Mode Timer Interrupt
J
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJ[S] Supervisor Mode Timer Interrupt
J
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJ[S] Supervisor Mode Timer Interrupt
J
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJ[S] Supervisor Mode Timer Interrupt
J
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJ[S] Supervisor Mode Timer Interrupt
J
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJ[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 3 COUNTER = 8]
D
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJD[S] Supervisor Mode Timer Interrupt
D
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDD[S] Supervisor Mode Timer Interrupt
D
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDD[S] Supervisor Mode Timer Interrupt
D
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDD[S] Supervisor Mode Timer Interrupt
D
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDD[S] Supervisor Mode Timer Interrupt
D
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDD[S] Supervisor Mode Timer Interrupt
D
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDD[S] Supervisor Mode Timer Interrupt
D
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDD[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 2 COUNTER = 9]
C
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDC[S] Supervisor Mode Timer Interrupt
C
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCC[S] Supervisor Mode Timer Interrupt
C
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCC[S] Supervisor Mode Timer Interrupt
C
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCC[S] Supervisor Mode Timer Interrupt
C
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCC[S] Supervisor Mode Timer Interrupt
C
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCC[S] Supervisor Mode Timer Interrupt
C
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCC[S] Supervisor Mode Timer Interrupt
C
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCC[S] Supervisor Mode Timer Interrupt
C
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCC[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 6 COUNTER = 10]
G
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCG[S] Supervisor Mode Timer Interrupt
G
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGG[S] Supervisor Mode Timer Interrupt
G
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGG[S] Supervisor Mode Timer Interrupt
G
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGG[S] Supervisor Mode Timer Interrupt
G
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGG[S] Supervisor Mode Timer Interrupt
G
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGG[S] Supervisor Mode Timer Interrupt
G
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGG[S] Supervisor Mode Timer Interrupt
G
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGG[S] Supervisor Mode Timer Interrupt
G
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGG[S] Supervisor Mode Timer Interrupt
G
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGG[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 10 COUNTER = 11]
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGK[S] Supervisor Mode Timer Interrupt
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKK[S] Supervisor Mode Timer Interrupt
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKK[S] Supervisor Mode Timer Interrupt
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKK[S] Supervisor Mode Timer Interrupt
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKK[S] Supervisor Mode Timer Interrupt
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKK[S] Supervisor Mode Timer Interrupt
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKK[S] Supervisor Mode Timer Interrupt
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKK[S] Supervisor Mode Timer Interrupt
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKK[S] Supervisor Mode Timer Interrupt
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKK[S] Supervisor Mode Timer Interrupt
K
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKK[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 5 COUNTER = 12]
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFFFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFFFFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFFFFFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFFFFFFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFFFFFFFF[S] Supervisor Mode Timer Interrupt
F
```

```
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFFFFFFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFFFFFFFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFFFFFFFFF[S] Supervisor Mode Timer Interrupt
F
IHHOOLLLNNNBBBBMMMMEEEEEPPPPPPJJJJJJJDDDDDDDDCCCCCCCCCGGGGGGGGGGKKKKKKKKKKKFFFFFFFFFFFF
NR_TASKS = 16, SJF test passed!

[S] Supervisor Mode Timer Interrupt
SET [PID = 0 COUNTER = 1]
SET [PID = 1 COUNTER = 4]
SET [PID = 2 COUNTER = 10]
SET [PID = 3 COUNTER = 4]
SET [PID = 4 COUNTER = 10]
SET [PID = 5 COUNTER = 10]
SET [PID = 6 COUNTER = 5]
SET [PID = 7 COUNTER = 2]
SET [PID = 8 COUNTER = 9]
SET [PID = 9 COUNTER = 4]
SET [PID = 10 COUNTER = 4]
SET [PID = 11 COUNTER = 10]
SET [PID = 12 COUNTER = 5]
SET [PID = 13 COUNTER = 10]
SET [PID = 14 COUNTER = 4]
SET [PID = 15 COUNTER = 7]
```

测试通过

- `NR_TASKS = 16` 情况下优先级调度算法的测试结果：

```
[S] Supervisor Mode Timer Interrupt

switch to [PID = 5 COUNTER = 12 PRIORITY = 39]
F
F[S] Supervisor Mode Timer Interrupt
F
FF[S] Supervisor Mode Timer Interrupt
F
FFF[S] Supervisor Mode Timer Interrupt
F
FFFF[S] Supervisor Mode Timer Interrupt
F
FFFFF[S] Supervisor Mode Timer Interrupt
F
FFFFFF[S] Supervisor Mode Timer Interrupt
F
FFFFFFF[S] Supervisor Mode Timer Interrupt
F
FFFFFFFF[S] Supervisor Mode Timer Interrupt
F
FFFFFFFFF[S] Supervisor Mode Timer Interrupt
F
FFFFFFFFFF[S] Supervisor Mode Timer Interrupt
F
FFFFFFFFFFF[S] Supervisor Mode Timer Interrupt
F
FFFFFFFFFFFF[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 10 COUNTER = 11 PRIORITY = 43]
K
FFFFFFFFFFFFK[S] Supervisor Mode Timer Interrupt
K
FFFFFFFFFFFFKK[S] Supervisor Mode Timer Interrupt
K
FFFFFFFFFFFFKKK[S] Supervisor Mode Timer Interrupt
K
FFFFFFFFFFFFKKKK[S] Supervisor Mode Timer Interrupt
K
FFFFFFFFFFFFKKKKK[S] Supervisor Mode Timer Interrupt
K
FFFFFFFFFFFFKKKKKK[S] Supervisor Mode Timer Interrupt
K
FFFFFFFFFFFFKKKKKKK[S] Supervisor Mode Timer Interrupt
K
FFFFFFFFFFFFKKKKKKKK[S] Supervisor Mode Timer Interrupt
K
FFFFFFFFFFFFKKKKKKKKK[S] Supervisor Mode Timer Interrupt
K
FFFFFFFFFFFFKKKKKKKKKK[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 6 COUNTER = 10 PRIORITY = 1]
G
FFFFFFFFFFFFKKKKKKKKKKG[S] Supervisor Mode Timer Interrupt
G
FFFFFFFFFFFFKKKKKKKKKKGG[S] Supervisor Mode Timer Interrupt
G
FFFFFFFFFFFFKKKKKKKKKKGGG[S] Supervisor Mode Timer Interrupt
G
FFFFFFFFFFFFKKKKKKKKKKGGGG[S] Supervisor Mode Timer Interrupt
G
FFFFFFFFFFFFKKKKKKKKKKGGGGG[S] Supervisor Mode Timer Interrupt
G
FFFFFFFFFFFFKKKKKKKKKKGGGGGG[S] Supervisor Mode Timer Interrupt
G
FFFFFFFFFFFFKKKKKKKKKKGGGGGGG[S] Supervisor Mode Timer Interrupt
G
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGG[S] Supervisor Mode Timer Interrupt
G
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGG[S] Supervisor Mode Timer Interrupt
G
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGG[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 2 COUNTER = 9 PRIORITY = 88]
C
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGC[S] Supervisor Mode Timer Interrupt
C
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCC[S] Supervisor Mode Timer Interrupt
C
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCC[S] Supervisor Mode Timer Interrupt
C
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCC[S] Supervisor Mode Timer Interrupt
C
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCC[S] Supervisor Mode Timer Interrupt
C
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCC[S] Supervisor Mode Timer Interrupt
C
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCC[S] Supervisor Mode Timer Interrupt
C
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCC[S] Supervisor Mode Timer Interrupt
C
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCC[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 3 COUNTER = 8 PRIORITY = 52]
D
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCD[S] Supervisor Mode Timer Interrupt
D
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDD[S] Supervisor Mode Timer Interrupt
D
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDD[S] Supervisor Mode Timer Interrupt
D
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDD[S] Supervisor Mode Timer Interrupt
D
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDD[S] Supervisor Mode Timer Interrupt
D
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDD[S] Supervisor Mode Timer Interrupt
D
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDD[S] Supervisor Mode Timer Interrupt
D
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDD[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 9 COUNTER = 7 PRIORITY = 28]
J
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJ[S] Supervisor Mode Timer Interrupt
J
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJ[S] Supervisor Mode Timer Interrupt
J
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJ[S] Supervisor Mode Timer Interrupt
J
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJ[S] Supervisor Mode Timer Interrupt
J
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJ[S] Supervisor Mode Timer Interrupt
J
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJ[S] Supervisor Mode Timer Interrupt
J
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJ[S] Supervisor Mode Timer Interrupt
```

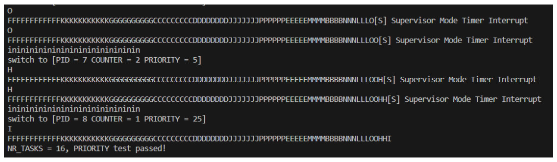
```
switch to [PID = 15 COUNTER = 6 PRIORITY = 30]
P
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJP[S] Supervisor Mode Timer Interrupt
P
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPP[S] Supervisor Mode Timer Interrupt
P
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPP[S] Supervisor Mode Timer Interrupt
P
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPP[S] Supervisor Mode Timer Interrupt
P
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPP[S] Supervisor Mode Timer Interrupt
P
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPP[S] Supervisor Mode Timer Interrupt

switch to [PID = 4 COUNTER = 5 PRIORITY = 66]
E
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPE[S] Supervisor Mode Timer Interrupt
E
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEE[S] Supervisor Mode Timer Interrupt
E
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEE[S] Supervisor Mode Timer Interrupt
E
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEE[S] Supervisor Mode Timer Interrupt
E
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEE[S] Supervisor Mode Timer Interrupt
```

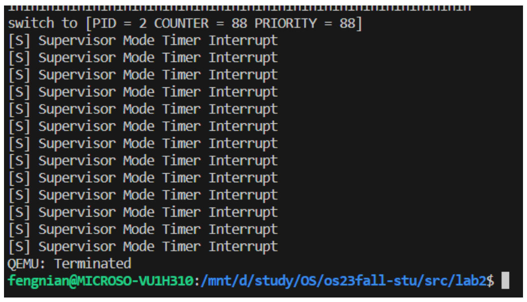
```
switch to [PID = 12 COUNTER = 4 PRIORITY = 8]
M
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEM[S] Supervisor Mode Timer Interrupt
M
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMM[S] Supervisor Mode Timer Interrupt
M
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMM[S] Supervisor Mode Timer Interrupt
M
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMM[S] Supervisor Mode Timer Interrupt
```

```
switch to [PID = 1 COUNTER = 4 PRIORITY = 37]
B
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMMB[S] Supervisor Mode Timer Interrupt
B
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMMBB[S] Supervisor Mode Timer Interrupt
B
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMMBBB[S] Supervisor Mode Timer Interrupt
B
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMMBBBB[S] Supervisor Mode Timer Interrupt
```

```
N
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMMBBBBN[S] Supervisor Mode Timer Interrupt
N
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMMBBBBNN[S] Supervisor Mode Timer Interrupt
N
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMMBBBBNNN[S] Supervisor Mode Timer Interrupt
ininininininininininininininin
switch to [PID = 11 COUNTER = 3 PRIORITY = 21]
L
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMMBBBBNNNL[S] Supervisor Mode Timer Interrupt
L
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMMBBBBNNNLL[S] Supervisor Mode Timer Interrupt
L
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJJPPPPPPEEEEEMMMMBBBBNNNLLL[S] Supervisor Mode Timer Interrupt
```



```
O
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJPPPPPPEEEEEMMMBBBBNNNLLLO[S] Supervisor Mode Timer Interrupt
O
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJPPPPPPEEEEEMMMBBBBNNNLLLOO[S] Supervisor Mode Timer Interrupt
inininininininininininininin
switch to [PID = 7 COUNTER = 2 PRIORITY = 5]
H
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJPPPPPPEEEEEMMMBBBBNNNLLLOOH[S] Supervisor Mode Timer Interrupt
H
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJPPPPPPEEEEEMMMBBBBNNNLLLOOHH[S] Supervisor Mode Timer Interrupt
inininininininininininininin
switch to [PID = 8 COUNTER = 1 PRIORITY = 25]
I
FFFFFFFFFFFFKKKKKKKKKKGGGGGGGGGGCCCCCCCCCDDDDDDDDJJJJJJPPPPPPEEEEEMMMBBBBNNNLLLOOHHI
NR_TASKS = 16, PRIORITY test passed!
```



```
switch to [PID = 2 COUNTER = 88 PRIORITY = 88]
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
[S] Supervisor Mode Timer Interrupt
QEMU: Terminated
fengnian@MICROSO-VU1H310:/mnt/d/study/OS/os23fall-stu/src/lab2$
```

测试通过（从第二轮开始，完全按`priority`由高到低调度，此时`counter`与`priority`相同，所以从`priority`最大的线程2开始）

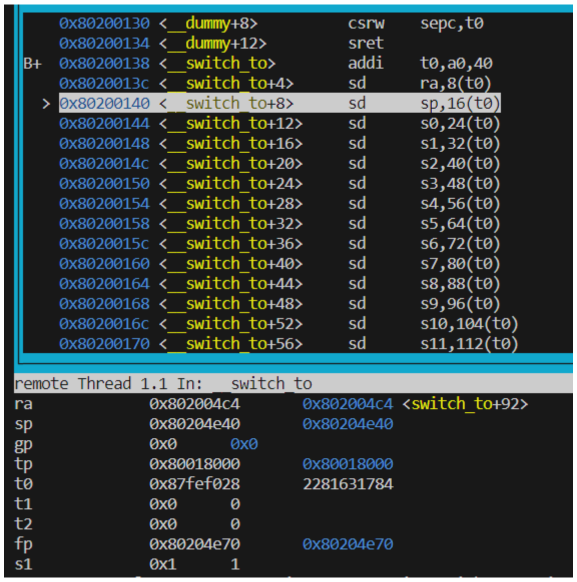
三、思考题

> 1. **在 RV64 中一共用 32 个通用寄存器，为什么 `context_switch` 中只保存了14个?**

答：因为`switch`函数调用时只需要保存`Callee Saved Register`和`ra`、`sp`，`Caller Saved Register`会被编译器自动保存在当前的栈上，也就是只需要保存`ra、sp、s0~s11`寄存器，

> 2. 当线程第一次调用时，其 `ra` 所代表的返回点是 `__dummy`。那么在之后的线程调用中 `context_switch` 中，`ra` 保存/恢复的函数返回点是什么呢? 请同学用 gdb 尝试追踪一次完整的线程切换流程，并关注每一次 `ra` 的变换 (需要截图)。

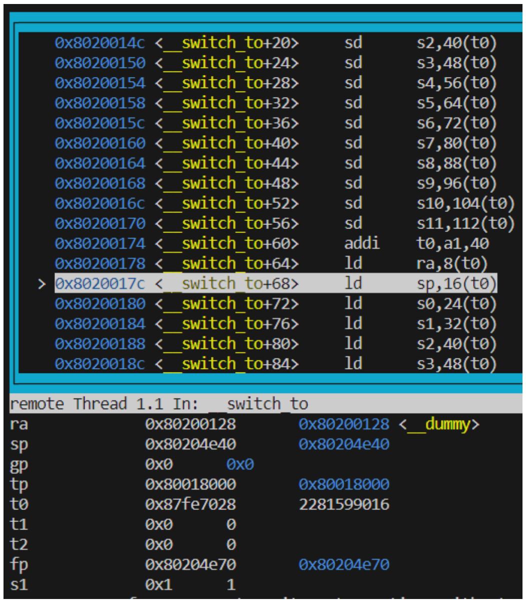
- 运行到第一次保存`ra`的地方，查看此时`ra`的值为0x802004c4（switch+92）



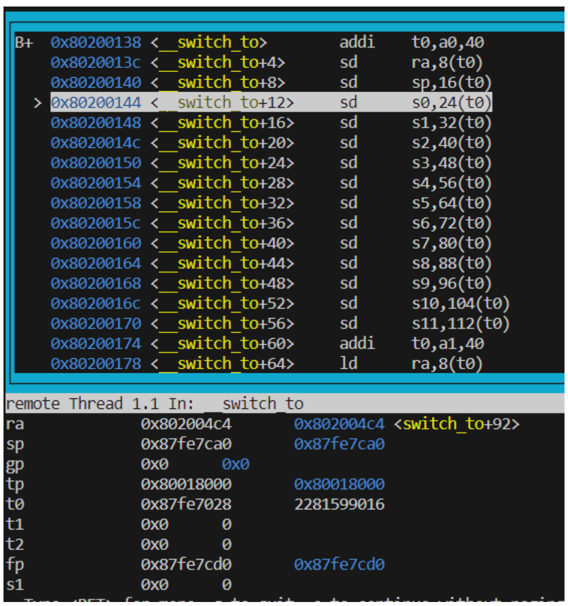
```
   0x80200130 <__dummy+8>      csrw   sepc,t0
   0x80200134 <__dummy+12>     sret
B+ 0x80200138 <__switch_to>    addi   t0,a0,40
   0x8020013c <__switch_to+4>  sd     ra,8(t0)
 > 0x80200140 <__switch_to+8>  sd     sp,16(t0)
   0x80200144 <__switch_to+12> sd     s0,24(t0)
   0x80200148 <__switch_to+16> sd     s1,32(t0)
   0x8020014c <__switch_to+20> sd     s2,40(t0)
   0x80200150 <__switch_to+24> sd     s3,48(t0)
   0x80200154 <__switch_to+28> sd     s4,56(t0)
   0x80200158 <__switch_to+32> sd     s5,64(t0)
   0x8020015c <__switch_to+36> sd     s6,72(t0)
   0x80200160 <__switch_to+40> sd     s7,80(t0)
   0x80200164 <__switch_to+44> sd     s8,88(t0)
   0x80200168 <__switch_to+48> sd     s9,96(t0)
   0x8020016c <__switch_to+52> sd     s10,104(t0)
   0x80200170 <__switch_to+56> sd     s11,112(t0)

remote Thread 1.1 In: __switch_to
ra             0x802004c4       0x802004c4 <switch_to+92>
sp             0x80204e40       0x80204e40
gp             0x0      0x0
tp             0x80018000       0x80018000
t0             0x87fef028       2281631784
t1             0x0      0
t2             0x0      0
fp             0x80204e70       0x80204e70
s1             0x1      1
```

- 运行到第一次恢复`ra`的地方，此时`ra`为`dummy`的地址



- 运行到第二次保存`ra`的地方，查看此时`ra`的值还是0x802004c4（switch+92）



- 之后都是第一次被调度，`ra`的变化一样
- 直到进程第二次被调度，此时恢复`ra`的时候，`ra`不再是为`dummy`的地址，而是0x802004c4（switch+92）

```
   0x80200154 <__switch_to+28>    sd      s4,56(t0)
   0x80200158 <__switch_to+32>    sd      s5,64(t0)
   0x8020015c <__switch_to+36>    sd      s6,72(t0)
   0x80200160 <__switch_to+40>    sd      s7,80(t0)
   0x80200164 <__switch_to+44>    sd      s8,88(t0)
   0x80200168 <__switch_to+48>    sd      s9,96(t0)
   0x8020016c <__switch_to+52>    sd      s10,104(t0)
   0x80200170 <__switch_to+56>    sd      s11,112(t0)
   0x80200174 <__switch_to+60>    addi    t0,a1,40
   0x80200178 <__switch_to+64>    ld      ra,8(t0)
 > 0x8020017c <__switch_to+68>    ld      sp,16(t0)
   0x80200180 <__switch_to+72>    ld      s0,24(t0)
   0x80200184 <__switch_to+76>    ld      s1,32(t0)
   0x80200188 <__switch_to+80>    ld      s2,40(t0)
   0x8020018c <__switch_to+84>    ld      s3,48(t0)
   0x80200190 <__switch_to+88>    ld      s4,56(t0)
   0x80200194 <__switch_to+92>    ld      s5,64(t0)

remote Thread 1.1 In: __switch_to
ra             0x802004c4        0x802004c4 <switch_to+92>
sp             0x87ff9c70        0x87ff9c70
gp             0x0      0x0
tp             0x80018000        0x80018000
t0             0x87ffa028        2281676840
t1             0x0      0
t2             0x0      0
fp             0x87ff9ca0        0x87ff9ca0
s1             0x87ff8000        2281668608
```

- 之后无论保存还是恢复，`ra`的值均为0x802004c4（switch+92），不再变化

```
B+ 0x80200138 <__switch_to>       addi    t0,a0,40
   0x8020013c <__switch_to+4>     sd      ra,8(t0)
 > 0x80200140 <__switch_to+8>     sd      sp,16(t0)
   0x80200144 <__switch_to+12>    sd      s0,24(t0)
   0x80200148 <__switch_to+16>    sd      s1,32(t0)
   0x8020014c <__switch_to+20>    sd      s2,40(t0)
   0x80200150 <__switch_to+24>    sd      s3,48(t0)
   0x80200154 <__switch_to+28>    sd      s4,56(t0)
   0x80200158 <__switch_to+32>    sd      s5,64(t0)
   0x8020015c <__switch_to+36>    sd      s6,72(t0)
   0x80200160 <__switch_to+40>    sd      s7,80(t0)
   0x80200164 <__switch_to+44>    sd      s8,88(t0)
   0x80200168 <__switch_to+48>    sd      s9,96(t0)
   0x8020016c <__switch_to+52>    sd      s10,104(t0)
   0x80200170 <__switch_to+56>    sd      s11,112(t0)
   0x80200174 <__switch_to+60>    addi    t0,a1,40
   0x80200178 <__switch_to+64>    ld      ra,8(t0)

remote Thread 1.1 In: __switch_to
ra             0x802004c4        0x802004c4 <switch_to+92>
sp             0x87ff8ca0        0x87ff8ca0
gp             0x0      0x0
tp             0x80018000        0x80018000
t0             0x87ff8028        2281668648
t1             0x0      0
t2             0x0      0
fp             0x87ff8cd0        0x87ff8cd0
s1             0x0      0
```

这个返回的地址（proc.c:31）就是调用switch_to的下一行

```
   0x802004a4 <switch_to+60>      sd      a5,-24(s0)
   0x802004a8 <switch_to+64>      auipc   a5,0x5
   0x802004ac <switch_to+68>      addi    a5,a5,-1176
   0x802004b0 <switch_to+72>      ld      a4,-40(s0)
   0x802004b4 <switch_to+76>      sd      a4,0(a5)
   0x802004b8 <switch_to+80>      ld      a1,-40(s0)
   0x802004bc <switch_to+84>      ld      a0,-24(s0)
   0x802004c0 <switch_to+88>      jal     ra,0x80200138 <__switch_to>
 > 0x802004c4 <switch_to+92>      nop
   0x802004c8 <switch_to+96>      ld      ra,40(sp)
   0x802004cc <switch_to+100>     ld      s0,32(sp)
   0x802004d0 <switch_to+104>     addi    sp,sp,48
   0x802004d4 <switch_to+108>     ret
   0x802004d8 <task_init>         addi    sp,sp,-48
   0x802004dc <task_init+4>       sd      ra,40(sp)
   0x802004e0 <task_init+8>       sd      s0,32(sp)
   0x802004e4 <task_init+12>      addi    s0,sp,48

remote Thread 1.1 In: switch_to
(gdb) n
(gdb) n
(gdb) n
(gdb) n
(gdb) n
(gdb) n
(gdb) n
(gdb) n
switch_to (next=0x87ffa000) at proc.c:31
```

```c
void switch_to(struct task_struct* next) {
    /* YOUR CODE HERE */
    //判断下一个执行的线程 next 与当前的线程 current 是否为同一个线程，如果是同一个线程，则无需做任何处理，否则调用 __switch_to 进行线程切换。
    if(current->pid != next->pid){
        struct task_struct*tmp = current;
        current = next ;
        __switch_to(tmp, next);
    }
}
```

四、遇到的问题与心得

通过操作系统的第一次实验，我了解了线程的概念，并学习了线程相关结构体，最后实现了线程的调度，在实验过程中，由于对相关知识的不了解，遇到了一些问题，我在搜索了解相关知识以及自己尝试后终于解决了如下问题：

1. 没有在_start里调用初始化函数导致一直卡住，在反复调试后还是不知道出现什么问题，最后仔细看实验指导才发现问题
2. 对于优先级调度算法一开始理解错误，以为是先考虑优先级再考虑时间，导致结果错误