实验 6：RV64 内核线程调度

姓名：张云策 学号：3200105787 学院：计算机科学与技术学院

课程名称：计算机系统 II 同组学生姓名：/

实验时间：2021. 实验地点：紫金港机房 指导老师：申文博

一、 实验目的和要求

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。

二、 实验内容和原理

2.1 实验内容

添加 rand.c / rand.h, string.h/string.c, mm.h / mm.c 等函数文件，设计实现 proc.h, proc.c, 完善 entry.S 并且对 defs.h, head.S 以及 makefile 进行修改。

2.2 设计模块

- 修改 defs.h
  将下列宏添加，发现可以继续运行

```
#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)))
```

- 完成线程初始化

```c
#define LAB_TEST_NUM 32;
void task_init() {
    current = (struct task_struct *)TASK_BASE;
    idle = (struct task_struct*)kalloc();
    idle -> state = TASK_RUNNING;
    idle -> counter = 0;
    idle -> priority = 0;
    idle -> pid = 0;
    current = idle;
    task[0] = idle;
    for (int i = 0; i <= LAB_TEST_NUM; i++) {
        unsigned long ret;
    ret = kalloc();
    task[i] = ret;
    task[i] -> state = TASK_RUNNING;
    task[i] -> counter = 0;
    task[i] -> priority = rand();
    task[i] -> pid = i;
    task[i] -> thread.ra = &__dummy;
    task[i] -> thread.sp = ret + PGSIZE;
        if (i != 0) {
            puts("[PID = ");
            puti(task[i]->pid);
            puts("] Process Create Successfully! counter = ");
            puti(task[i]->counter);
#ifdef  PRIORITY
            puts(" priority = ");
            puti(task[i]->priority);
#endif
            puts("\n");
        }
    }
}
```

- 完善 entry.S，添加__dummy 和 dummy 部分，设置 spec

```
__dummy:
    la t0, dummy
    csrw sepc, t0
    ecall
    sret
```

- 利用__switch_to 在 entry.S 中实现线程切换，将当今的 ra/sp/S0~S11 寄存到结构中

```
__switch_to:
    li    a4,  40
    add   a3, a0, a4
    add   a4, a1, a4
    sd ra,0(a3)
    sd sp,8(a3)
    sd s0,16(a3)
    sd s1,24(a3)
    sd s2,32(a3)
    sd s3,40(a3)
    sd s4,48(a3)
    sd s5,56(a3)
    sd s6,64(a3)
    sd s7,72(a3)
    sd s8,80(a3)
    sd s9,88(a3)
    sd s10,96(a3)
    sd s11,104(a3)
    ld ra,0(a4)
    ld sp,8(a4)
    ld s0,16(a4)
    ld s1,24(a4)
    ld s2,32(a4)
    ld s3,40(a4)
    ld s4,48(a4)
    ld s5,56(a4)
    ld s6,64(a4)
    ld s7,72(a4)
    ld s8,80(a4)
    ld s9,88(a4)
    ld s10,96(a4)
    ld s11,104(a4)

    la t0, current
    sd a1,0(t0)

    mv t0, a1
    addi t0, t0, 40
    ld ra, 0(t0)
    ld sp, 8(t0)
    addi t0, t0, 0x10
    ld s0, 0(t0)
    ld s1, 8(t0)
    ld s2, 16(t0)
    ld s3, 24(t0)
    ld s4, 32(t0)
    ld s5, 40(t0)
    ld s6, 48(t0)
    ld s7, 56(t0)
    ld s8, 64(t0)
    ld s9, 72(t0)
    ld s10, 80(t0)
    ld s11, 88(t0)
    ret
```

- 实现 do_timer 函数

```
void do_timer() {
#ifdef  PRIORITY
    puts("[PID = ");
    puti(current->pid);
    puts("] ");
    puts("Context Calculation: ");
    puts("counter = ");
    puti(current->counter);
    puts("\n");

    current->counter--;
    if (current->counter <= 0)
        schedule();

#else
    current->counter--;
    if (current->counter <= 0)
        current->counter = (current->pid == 0) ? 5 : 8 - current->pid;
    schedule();
#endif
}
```

- 实现 sjf 调度

```
void schedule() {
#ifdef SJF
    int i_min_cnt    = LAB_TEST_NUM;
    _Bool all_zeroes = 1;
    for (int i = LAB_TEST_NUM; i > 0; i--)
        if (task[i]->state == TASK_RUNNING) {
            if (task[i]->counter > 0 && task[i]->counter < task[i_min_cnt]->counter ||task[i_min_cnt]->counter == 0)
                i_min_cnt = i;
            if (task[i]->counter > 0)
                all_zeroes = 0;
        }
    if (all_zeroes) {
        for (int i = 1; i <= LAB_TEST_NUM; i++)
            if (task[i]->state == TASK_RUNNING)
            {
                task[i]->counter = rand();

                puts("[PID = ");
                puti(task[i]->pid);
                puts("] Reset counter = ");
                puti(task[i]->counter);
                puts("\n");
            }
        schedule();
    } else {
        puts("[!] Switch from task ");
        puti(current->pid);
        puts(" to task ");
        puti(task[i_min_cnt]->pid);
        puts(", prio: ");
        puti(task[i_min_cnt]->priority);
        puts(", counter: ");
        puti(task[i_min_cnt]->counter);
        puts("\n");

        switch_to(task[i_min_cnt]);
    }
```

- 实现优先级调度

```
#else
    int max_pri = __INT_MAX__, i_min_cnt = 1;
    for (int i = 1; i <= LAB_TEST_NUM; i++)
        if (task[i]->state == TASK_RUNNING)
            if (task[i]->priority < max_pri) {
                i_min_cnt = i;
                max_pri   = task[i]->priority;
            } else if (task[i]->priority == max_pri &&
                       task[i]->counter < task[i_min_cnt]->counter && task[i]->counter > 0)
                i_min_cnt = i;

    puts("[!] Switch from task ");
    puti(current->pid);
    puts(" to task ");
    puti(task[i_min_cnt]->pid);
    puts(", prio: ");
    puti(task[i_min_cnt]->priority);
    puts(", counter: ");
    puti(task[i_min_cnt]->counter);
    puts("\n");

    for (int i = 1; i <= LAB_TEST_NUM; i++)
        if (task[i]->state == TASK_RUNNING)
            task[i]->priority = rand();

    puts("tasks' priority changed\n");
    for (int i = 1; i <= LAB_TEST_NUM; i++)
        if (task[i]->state == TASK_RUNNING) {
            puts("[PID = ");
            puti(task[i]->pid);
            puts("] ");
            puts("counter = ");
            puti(task[i]->counter);
            puts(" priority = ");
            puti(task[i]->priority);
            puts("\n");
        }
    switch_to(task[i_min_cnt]);
#endif
```

三、 主要仪器设备

Dockers in Lab3

四、 实验结果与分析

```
oslab@e3ecaba9cde0:~/lab3$ make run
qemu-system-riscv64: warning: No -bios option specified. Not loading a firmware.
qemu-system-riscv64: warning: This default will change in a future QEMU release. Please use the -
bios option to avoid breakages when this happens.
qemu-system-riscv64: warning: See QEMU's deprecation documentation for details.
ZJU-system lab6
[PID = 1] Process Create Successfully! counter = 7 priority = 5
[PID = 2] Process Create Successfully! counter = 6 priority = 5
[PID = 3] Process Create Successfully! counter = 5 priority = 5
[PID = 4] Process Create Successfully! counter = 4 priority = 5
[!] Switch from task 0 [task struct: 0xffffffe00001c000, sp: 0xffffffe00001d000] to task 4 [task
struct: 0xffffffe000020000, sp: 0xffffffe000021000], prio: 5, counter: 4
tasks' priority changed
[PID = 1] counter = 7 priority = 1
[PID = 2] counter = 6 priority = 4
[PID = 3] counter = 5 priority = 5
[PID = 4] counter = 4 priority = 4
[!] Switch from task 4 [task struct: 0xffffffe000020000, sp: 0xffffffe000021000] to task 1 [task
struct: 0xffffffe00001d000, sp: 0xffffffe00001e000], prio: 1, counter: 7
tasks' priority changed
[PID = 1] counter = 7 priority = 5
[PID = 2] counter = 6 priority = 5
[PID = 3] counter = 5 priority = 5
[PID = 4] counter = 3 priority = 2
[!] Switch from task 1 [task struct: 0xffffffe00001d000, sp: 0xffffffe00001e000] to task 4 [task
struct: 0xffffffe000020000, sp: 0xffffffe000020eb0], prio: 2, counter: 3
tasks' priority changed
[PID = 1] counter = 6 priority = 4
[PID = 2] counter = 6 priority = 4
[PID = 3] counter = 5 priority = 4
[PID = 4] counter = 3 priority = 5
[!] Switch from task 4 [task struct: 0xffffffe000020000, sp: 0xffffffe000020eb0] to task 3 [task
struct: 0xffffffe00001f000, sp: 0xffffffe000020000], prio: 4, counter: 5
tasks' priority changed
[PID = 1] counter = 6 priority = 5
[PID = 2] counter = 6 priority = 5
[PID = 3] counter = 5 priority = 4
[PID = 4] counter = 2 priority = 2
[!] Switch from task 3 [task struct: 0xffffffe00001f000, sp: 0xffffffe000020000] to task 4 [task
struct: 0xffffffe000020000, sp: 0xffffffe000020eb0], prio: 2, counter: 2
tasks' priority changed
[PID = 1] counter = 6 priority = 5
[PID = 2] counter = 6 priority = 3
[PID = 3] counter = 4 priority = 3
[PID = 4] counter = 2 priority = 4
```

```
[PID = 1] Context Calculation: counter = 1
[PID = 1] Process Create Successfully! counter = 1
[PID = 2] Process Create Successfully! counter = 4
[PID = 3] Process Create Successfully! counter = 5
[PID = 4] Process Create Successfully! counter = 4
[PID = 0] Context Calculation: counter = 0
[!] Switch from task 0 [task struct: 0xffffffe000020000, sp: 0xffffffe000021000] to task 1 [task
struct: 0xffffffe000021000, sp: 0xffffffe000022000], prio: 5, counter: 1
[PID = 1] Context Calculation: counter = 1
[!] Switch from task 1 [task struct: 0xffffffe000021000, sp: 0xffffffe000022000] to task 4 [task
struct: 0xffffffe000024000, sp: 0xffffffe000025000], prio: 5, counter: 4
[PID = 4] Context Calculation: counter = 4
[PID = 4] Context Calculation: counter = 3
[PID = 4] Context Calculation: counter = 2
[PID = 4] Context Calculation: counter = 1
[!] Switch from task 4 [task struct: 0xffffffe000024000, sp: 0xffffffe000025000] to task 2 [task
struct: 0xffffffe000022000, sp: 0xffffffe000023000], prio: 5, counter: 4
[PID = 2] Context Calculation: counter = 4
[PID = 2] Context Calculation: counter = 3
[PID = 2] Context Calculation: counter = 2
[PID = 2] Context Calculation: counter = 1
[!] Switch from task 2 [task struct: 0xffffffe000022000, sp: 0xffffffe000023000] to task 3 [task
struct: 0xffffffe000023000, sp: 0xffffffe000024000], prio: 5, counter: 5
[PID = 3] Context Calculation: counter = 5
[PID = 3] Context Calculation: counter = 4
[PID = 3] Context Calculation: counter = 3
[PID = 3] Context Calculation: counter = 2
[PID = 3] Context Calculation: counter = 1
[PID = 1] Reset counter = 5
[PID = 2] Reset counter = 5
[PID = 3] Reset counter = 5
[PID = 4] Reset counter = 2
[!] Switch from task 3 [task struct: 0xffffffe000023000, sp: 0xffffffe000024000] to task 4 [task
struct: 0xffffffe000024000, sp: 0xffffffe000024f00], prio: 5, counter: 2
[PID = 4] Context Calculation: counter = 2
[PID = 4] Context Calculation: counter = 1
[!] Switch from task 4 [task struct: 0xffffffe000024000, sp: 0xffffffe000024f00] to task 3 [task
struct: 0xffffffe000023000, sp: 0xffffffe000023f00], prio: 5, counter: 5
[PID = 3] Context Calculation: counter = 5
[PID = 3] Context Calculation: counter = 4
[PID = 3] Context Calculation: counter = 3
[PID = 3] Context Calculation: counter = 2
[PID = 3] Context Calculation: counter = 1
[!] Switch from task 3 [task struct: 0xffffffe000023000, sp: 0xffffffe000023f00] to task 2 [task
```

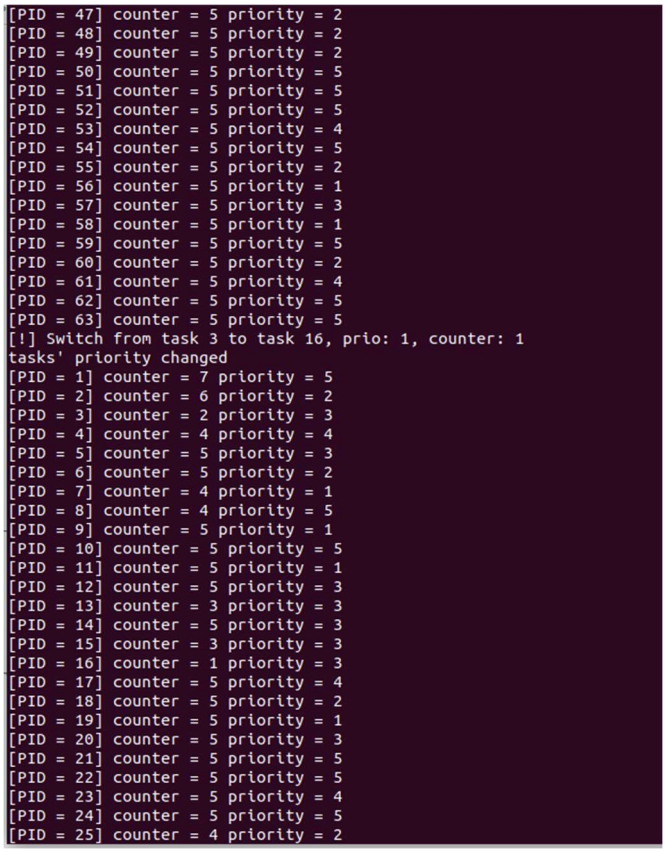
在 SJF 模式下：

```
[PID = 31] Process Create Successfully! counter = 4
[PID = 32] Process Create Successfully! counter = 2
[PID = 33] Process Create Successfully! counter = 2
[PID = 34] Process Create Successfully! counter = 2
[PID = 35] Process Create Successfully! counter = 5
[PID = 36] Process Create Successfully! counter = 5
[PID = 37] Process Create Successfully! counter = 5
[PID = 38] Process Create Successfully! counter = 4
[PID = 39] Process Create Successfully! counter = 5
[PID = 40] Process Create Successfully! counter = 2
[PID = 41] Process Create Successfully! counter = 1
[PID = 42] Process Create Successfully! counter = 3
[PID = 43] Process Create Successfully! counter = 1
[PID = 44] Process Create Successfully! counter = 5
[PID = 45] Process Create Successfully! counter = 2
[PID = 46] Process Create Successfully! counter = 4
[PID = 47] Process Create Successfully! counter = 5
[PID = 48] Process Create Successfully! counter = 5
[PID = 49] Process Create Successfully! counter = 5
[PID = 50] Process Create Successfully! counter = 2
[PID = 51] Process Create Successfully! counter = 3
[PID = 52] Process Create Successfully! counter = 4
[PID = 53] Process Create Successfully! counter = 3
[PID = 54] Process Create Successfully! counter = 2
[PID = 55] Process Create Successfully! counter = 1
[PID = 56] Process Create Successfully! counter = 5
[PID = 57] Process Create Successfully! counter = 1
[PID = 58] Process Create Successfully! counter = 5
[PID = 59] Process Create Successfully! counter = 1
[PID = 60] Process Create Successfully! counter = 3
[PID = 61] Process Create Successfully! counter = 3
[PID = 62] Process Create Successfully! counter = 3
[PID = 63] Process Create Successfully! counter = 3
[PID = 0] Context Calculation: counter = 0
[!] Switch from task 0 to task 59, prio: 5, counter: 1
[PID = 59] Context Calculation: counter = 1
[!] Switch from task 59 to task 57, prio: 5, counter: 1
[PID = 57] Context Calculation: counter = 1
[!] Switch from task 57 to task 55, prio: 5, counter: 1
[PID = 55] Context Calculation: counter = 1
[!] Switch from task 55 to task 43, prio: 5, counter: 1
[PID = 43] Context Calculation: counter = 1
[!] Switch from task 43 to task 41, prio: 5, counter: 1
```

```
[PID = 59] Context Calculation: counter = 1
[!] Switch from task 59 to task 57, prio: 5, counter: 1
[PID = 57] Context Calculation: counter = 1
[!] Switch from task 57 to task 55, prio: 5, counter: 1
[PID = 55] Context Calculation: counter = 1
[!] Switch from task 55 to task 43, prio: 5, counter: 1
[PID = 43] Context Calculation: counter = 1
[!] Switch from task 43 to task 41, prio: 5, counter: 1
[PID = 41] Context Calculation: counter = 1
[!] Switch from task 41 to task 27, prio: 5, counter: 1
[PID = 27] Context Calculation: counter = 1
[!] Switch from task 27 to task 21, prio: 5, counter: 1
[PID = 21] Context Calculation: counter = 1
[!] Switch from task 21 to task 1, prio: 5, counter: 1
[PID = 1] Context Calculation: counter = 1
[!] Switch from task 1 to task 54, prio: 5, counter: 2
[PID = 54] Context Calculation: counter = 2
[PID = 54] Context Calculation: counter = 1
[!] Switch from task 54 to task 50, prio: 5, counter: 2
[PID = 50] Context Calculation: counter = 2
[PID = 50] Context Calculation: counter = 1
[!] Switch from task 50 to task 45, prio: 5, counter: 2
[PID = 45] Context Calculation: counter = 2
[PID = 45] Context Calculation: counter = 1
[!] Switch from task 45 to task 40, prio: 5, counter: 2
[PID = 40] Context Calculation: counter = 2
[PID = 40] Context Calculation: counter = 1
[!] Switch from task 40 to task 34, prio: 5, counter: 2
[PID = 34] Context Calculation: counter = 2
[PID = 34] Context Calculation: counter = 1
[!] Switch from task 34 to task 33, prio: 5, counter: 2
[PID = 33] Context Calculation: counter = 2
[PID = 33] Context Calculation: counter = 1
[!] Switch from task 33 to task 32, prio: 5, counter: 2
[PID = 32] Context Calculation: counter = 2
[PID = 32] Context Calculation: counter = 1
[!] Switch from task 32 to task 16, prio: 5, counter: 2
[PID = 16] Context Calculation: counter = 2
[PID = 16] Context Calculation: counter = 1
[!] Switch from task 16 to task 8, prio: 5, counter: 2
[PID = 8] Context Calculation: counter = 2
[PID = 8] Context Calculation: counter = 1
[!] Switch from task 8 to task 63, prio: 5, counter: 3
[PID = 63] Context Calculation: counter = 3
```

在 Priority 模式下：

```
File  Edit  View  Search  Terminal  Help
[PID = 51] counter = 5 priority = 4
[PID = 52] counter = 5 priority = 4
[PID = 53] counter = 5 priority = 4
[PID = 54] counter = 5 priority = 3
[PID = 55] counter = 5 priority = 2
[PID = 56] counter = 5 priority = 1
[PID = 57] counter = 5 priority = 5
[PID = 58] counter = 5 priority = 1
[PID = 59] counter = 5 priority = 5
[PID = 60] counter = 5 priority = 1
[PID = 61] counter = 5 priority = 1
[PID = 62] counter = 5 priority = 4
[PID = 63] counter = 5 priority = 4
[!] Switch from task 16 to task 21, prio: 1, counter: 4
tasks' priority changed
[PID = 1] counter = 7 priority = 4
[PID = 2] counter = 6 priority = 3
[PID = 3] counter = 5 priority = 2
[PID = 4] counter = 1 priority = 4
[PID = 5] counter = 5 priority = 5
[PID = 6] counter = 5 priority = 2
[PID = 7] counter = 5 priority = 4
[PID = 8] counter = 5 priority = 4
[PID = 9] counter = 4 priority = 2
[PID = 10] counter = 5 priority = 2
[PID = 11] counter = 5 priority = 5
[PID = 12] counter = 5 priority = 5
[PID = 13] counter = 5 priority = 1
[PID = 14] counter = 5 priority = 1
[PID = 15] counter = 4 priority = 3
[PID = 16] counter = 4 priority = 3
[PID = 17] counter = 5 priority = 4
[PID = 18] counter = 5 priority = 5
[PID = 19] counter = 5 priority = 1
[PID = 20] counter = 5 priority = 1
[PID = 21] counter = 4 priority = 2
[PID = 22] counter = 5 priority = 2
[PID = 23] counter = 5 priority = 1
[PID = 24] counter = 5 priority = 2
[PID = 25] counter = 5 priority = 1
[PID = 26] counter = 5 priority = 1
[PID = 27] counter = 5 priority = 5
[PID = 28] counter = 5 priority = 5
[PID = 29] counter = 5 priority = 1
```



五、 讨论、心得

思考题：

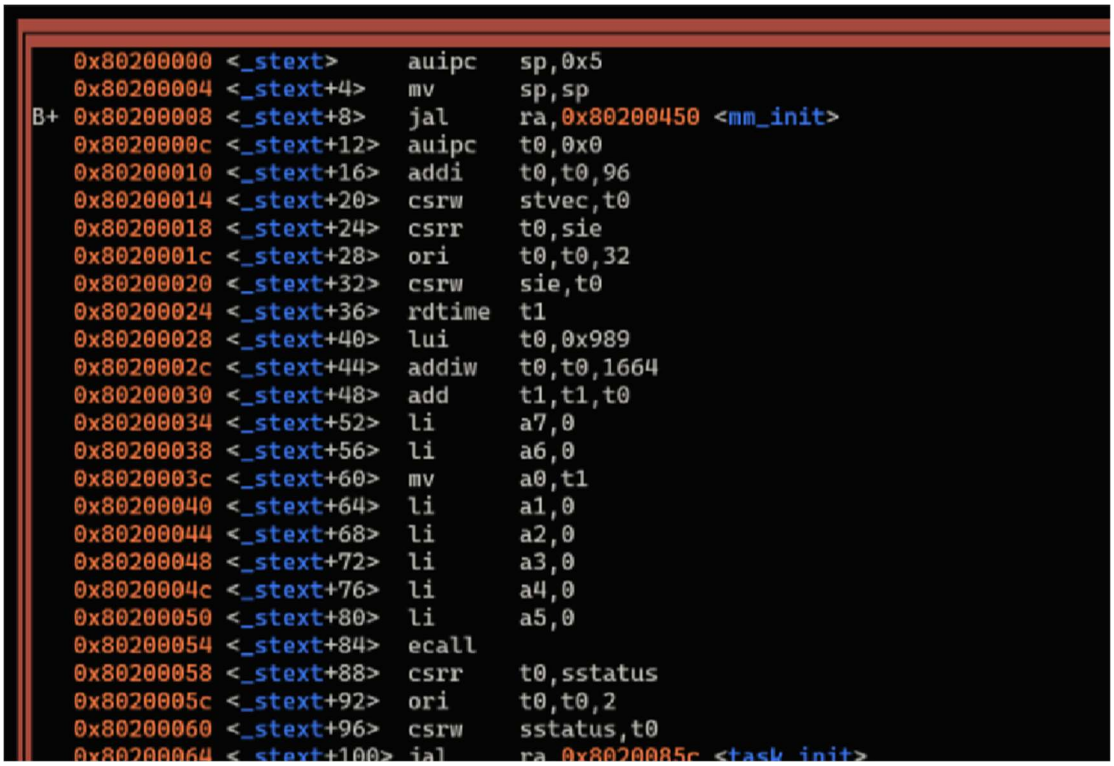
1. context_switch 中一共存在着 ra、sp、s0~s11 这十四个寄存器，ra 和 sp 寄存器主要用来记录切换对象的栈位置以及程序下一步执行的目的代码，所以需要在进程切换中保存，然后才可以恢复进程的时候程序可以继续运行。

而寄存器 s0~s11 中，s0 是帧指针，s1 的作用是“保存寄存器”，而寄存器 s2-s11，也是在程序运行中需要保存的寄存器，因此在上下文切换中需要保存。

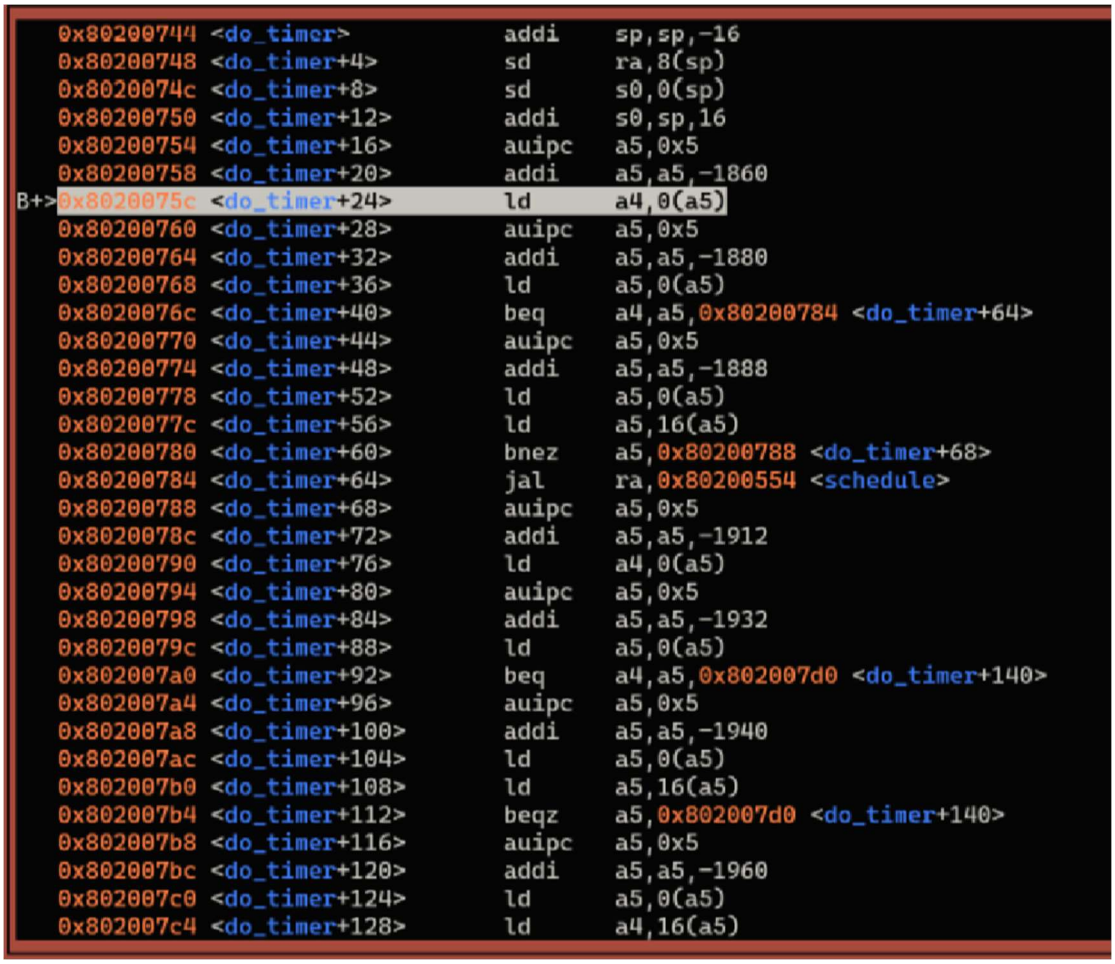
而其他寄存器的值都是临时变量的值，没有必要保存。

2. 进入_traps(1)，随后进入 schedule 函数(2)，进入 switch_to 函数，可以得出结论：ra 的值就是 schedule 的地址(3)，再次进入_switch_to 函数，ra 变成了调用_switch_to 的 switch_to 地址(4)，完成上下文切换后，ra 变成了__dummy 的地址（预设地址）(5)，在切换完成后，返回地址写入前，ra 为 switch_to 的地址(6)，写入后，仍为 switch_to 的地址(7)，我们可以得出结论 ra 存储的返回地址和先前函数的调用者地址相同

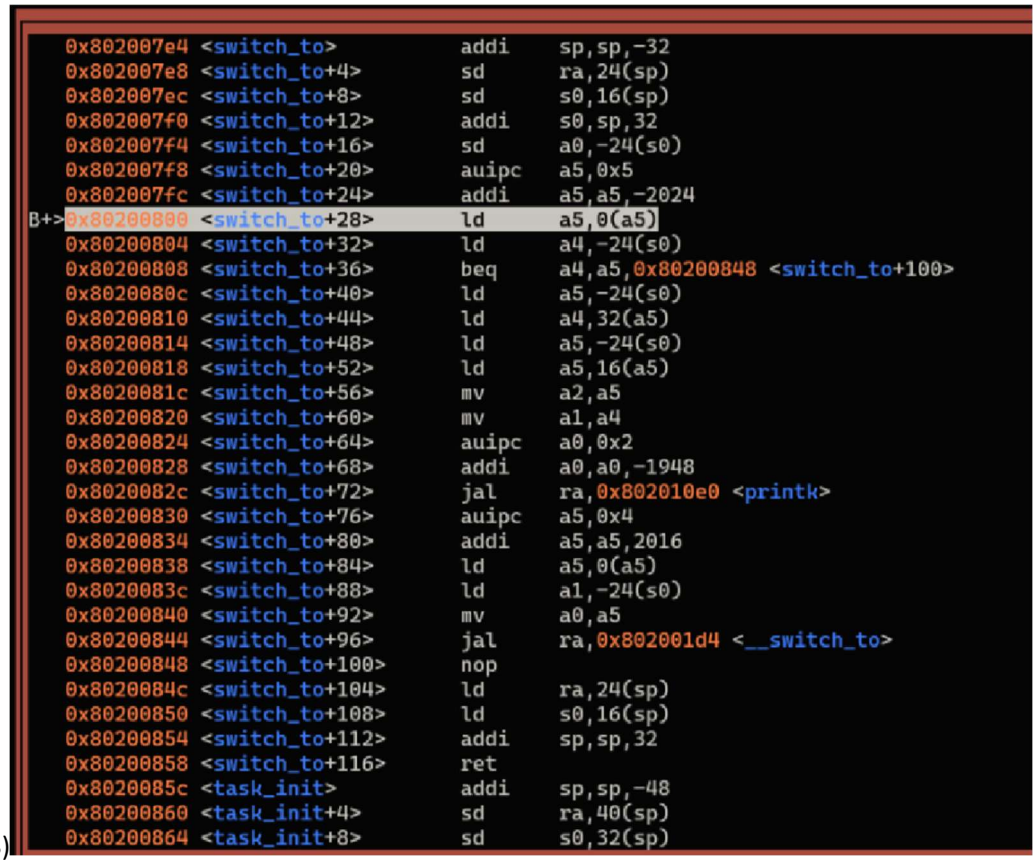
(1)

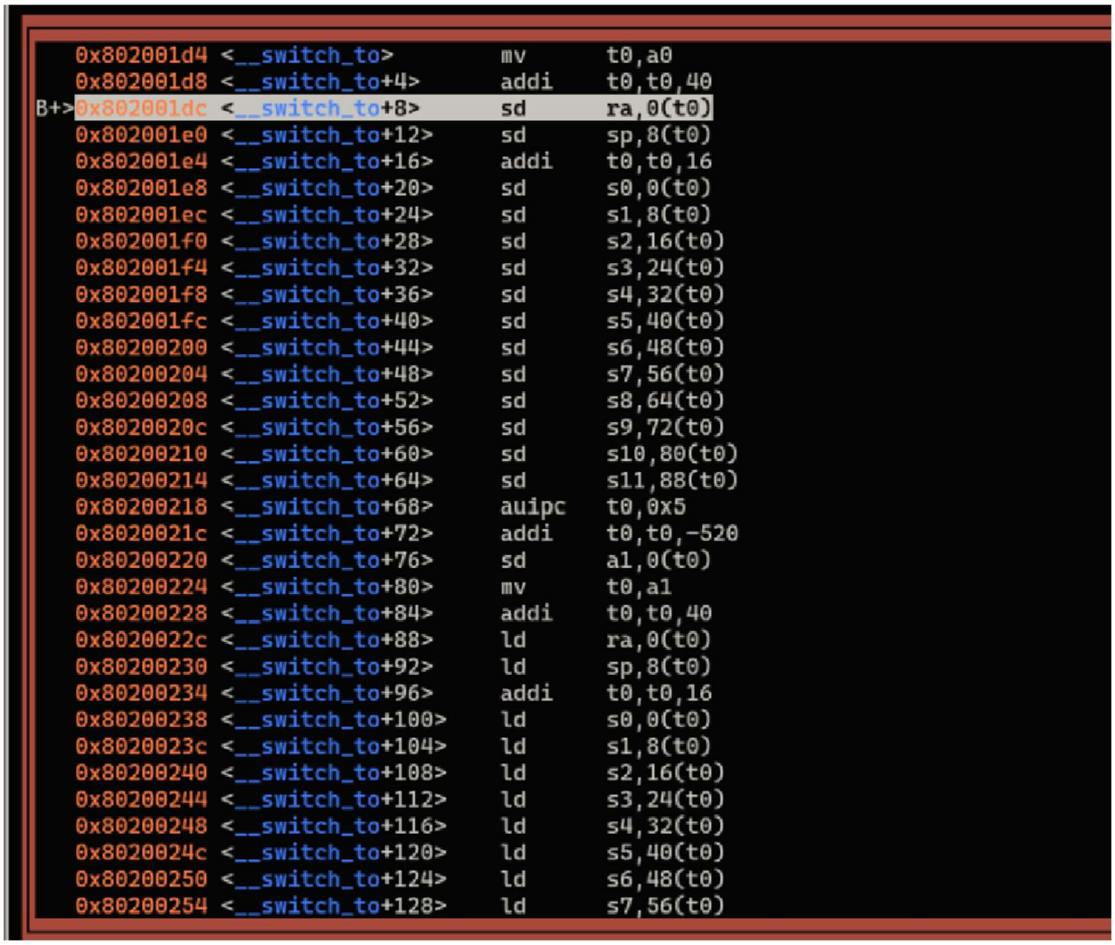

0x80200000 <_stext> auipc sp,0x5
0x80200004 <_stext+4> mv sp,sp
B+ 0x80200008 <_stext+8> jal ra,0x80200450 <mm_init>
0x8020000c <_stext+12> auipc t0,0x0
0x80200010 <_stext+16> addi t0,t0,96
0x80200014 <_stext+20> csrw stvec,t0
0x80200018 <_stext+24> csrr t0,sie
0x8020001c <_stext+28> ori t0,t0,32
0x80200020 <_stext+32> csrw sie,t0
0x80200024 <_stext+36> rdtime t1
0x80200028 <_stext+40> lui t0,0x989
0x8020002c <_stext+44> addiw t0,t0,1664
0x80200030 <_stext+48> add t1,t1,t0
0x80200034 <_stext+52> li a7,0
0x80200038 <_stext+56> li a6,0
0x8020003c <_stext+60> mv a0,t1
0x80200040 <_stext+64> li a1,0
0x80200044 <_stext+68> li a2,0
0x80200048 <_stext+72> li a3,0
0x8020004c <_stext+76> li a4,0
0x80200050 <_stext+80> li a5,0
0x80200054 <_stext+84> ecall
0x80200058 <_stext+88> csrr t0,sstatus
0x8020005c <_stext+92> ori t0,t0,2
0x80200060 <_stext+96> csrw sstatus,t0

(2)


0x80200744 <do_timer> addi sp,sp,-16
0x80200748 <do_timer+4> sd ra,8(sp)
0x8020074c <do_timer+8> sd s0,0(sp)
0x80200750 <do_timer+12> addi s0,sp,16
0x80200754 <do_timer+16> auipc a5,0x5
0x80200758 <do_timer+20> addi a5,a5,-1860
B+>0x8020075c <do_timer+24> ld a4,0(a5)
0x80200760 <do_timer+28> auipc a5,0x5
0x80200764 <do_timer+32> addi a5,a5,-1880
0x80200768 <do_timer+36> ld a5,0(a5)
0x8020076c <do_timer+40> beq a4,a5,0x80200784 <do_timer+64>
0x80200770 <do_timer+44> auipc a5,0x5
0x80200774 <do_timer+48> addi a5,a5,-1888
0x80200778 <do_timer+52> ld a5,0(a5)
0x8020077c <do_timer+56> ld a5,16(a5)
0x80200780 <do_timer+60> bnez a5,0x80200788 <do_timer+68>
0x80200784 <do_timer+64> jal ra,0x80200554 <schedule>
0x80200788 <do_timer+68> auipc a5,0x5
0x8020078c <do_timer+72> addi a5,a5,-1912
0x80200790 <do_timer+76> ld a4,0(a5)
0x80200794 <do_timer+80> auipc a5,0x5
0x80200798 <do_timer+84> addi a5,a5,-1932
0x8020079c <do_timer+88> ld a5,0(a5)
0x802007a0 <do_timer+92> beq a4,a5,0x802007d0 <do_timer+140>
0x802007a4 <do_timer+96> auipc a5,0x5
0x802007a8 <do_timer+100> addi a5,a5,-1940
0x802007ac <do_timer+104> ld a5,0(a5)
0x802007b0 <do_timer+108> ld a5,16(a5)
0x802007b4 <do_timer+112> beqz a5,0x802007d0 <do_timer+140>
0x802007b8 <do_timer+116> auipc a5,0x5
0x802007bc <do_timer+120> addi a5,a5,-1960
0x802007c0 <do_timer+124> ld a5,0(a5)
0x802007c4 <do_timer+128> ld a4,16(a5)

(3)

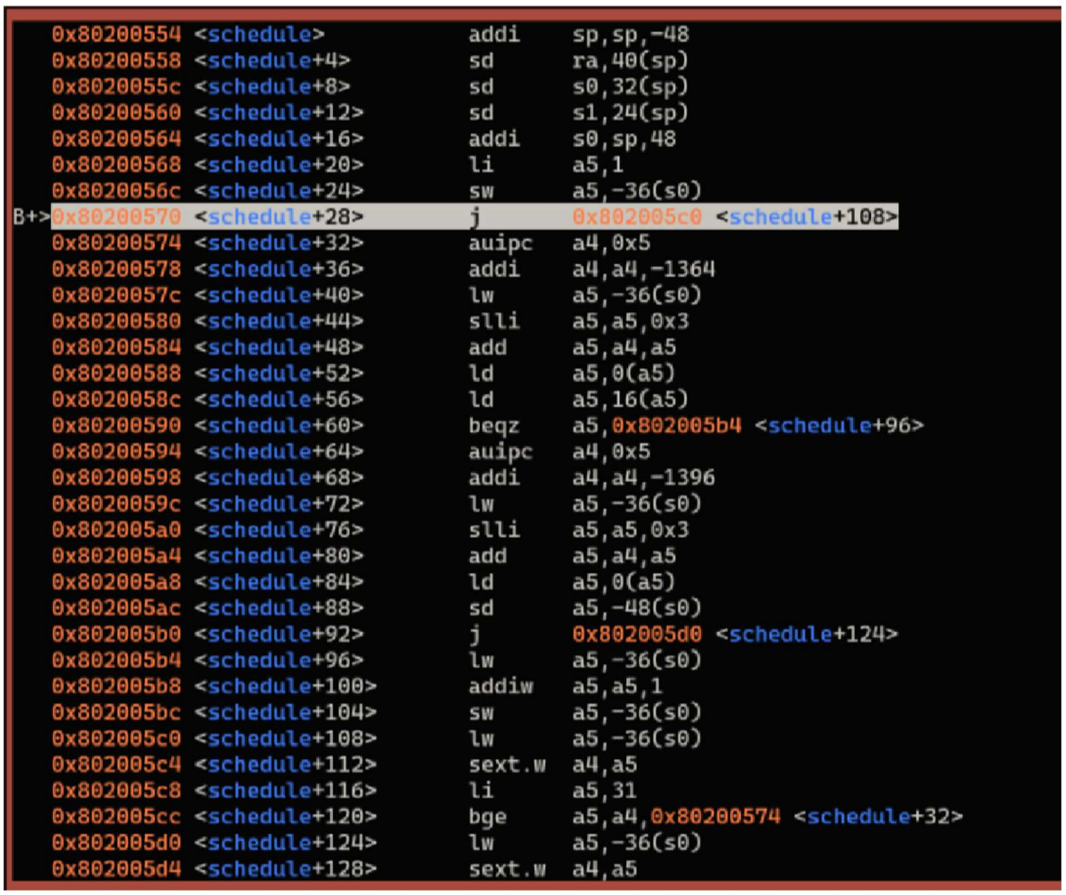

0x802007e4 <switch_to> addi sp,sp,-32
0x802007e8 <switch_to+4> sd ra,24(sp)
0x802007ec <switch_to+8> sd s0,16(sp)
0x802007f0 <switch_to+12> addi s0,sp,32
0x802007f4 <switch_to+16> sd a0,-24(s0)
0x802007f8 <switch_to+20> auipc a5,0x5
0x802007fc <switch_to+24> addi a5,a5,-2024
B+> 0x80200800 <switch_to+28> ld a5,0(a5)
0x80200804 <switch_to+32> ld a4,-24(s0)
0x80200808 <switch_to+36> beq a4,a5,0x80200848 <switch_to+100>
0x8020080c <switch_to+40> ld a5,-24(s0)
0x80200810 <switch_to+44> ld a4,32(a5)
0x80200814 <switch_to+48> ld a5,-24(s0)
0x80200818 <switch_to+52> ld a5,16(a5)
0x8020081c <switch_to+56> mv a2,a5
0x80200820 <switch_to+60> mv a1,a4
0x80200824 <switch_to+64> auipc a0,0x2
0x80200828 <switch_to+68> addi a0,a0,-1948
0x8020082c <switch_to+72> jal ra,0x802010e0 <printk>
0x80200830 <switch_to+76> auipc a5,0x4
0x80200834 <switch_to+80> addi a5,a5,2016
0x80200838 <switch_to+84> ld a5,0(a5)
0x8020083c <switch_to+88> ld a1,-24(s0)
0x80200840 <switch_to+92> mv a0,a5
0x80200844 <switch_to+96> jal ra,0x802001d4 <__switch_to>
0x80200848 <switch_to+100> nop
0x8020084c <switch_to+104> ld ra,24(sp)
0x80200850 <switch_to+108> ld s0,16(sp)
0x80200854 <switch_to+112> addi sp,sp,32
0x80200858 <switch_to+116> ret
0x8020085c <task_init> addi sp,sp,-48
0x80200860 <task_init+4> sd ra,40(sp)
0x80200864 <task_init+8> sd s0,32(sp)


0x802001d4 <__switch_to> mv t0,a0
0x802001d8 <__switch_to+4> addi t0,t0,40
B+>0x802001dc <__switch_to+8> sd ra,0(t0)
0x802001e0 <__switch_to+12> sd sp,8(t0)
0x802001e4 <__switch_to+16> addi t0,t0,16
0x802001e8 <__switch_to+20> sd s0,0(t0)
0x802001ec <__switch_to+24> sd s1,8(t0)
0x802001f0 <__switch_to+28> sd s2,16(t0)
0x802001f4 <__switch_to+32> sd s3,24(t0)
0x802001f8 <__switch_to+36> sd s4,32(t0)
0x802001fc <__switch_to+40> sd s5,40(t0)
0x80200200 <__switch_to+44> sd s6,48(t0)
0x80200204 <__switch_to+48> sd s7,56(t0)
0x80200208 <__switch_to+52> sd s8,64(t0)
0x8020020c <__switch_to+56> sd s9,72(t0)
0x80200210 <__switch_to+60> sd s10,80(t0)
0x80200214 <__switch_to+64> sd s11,88(t0)
0x80200218 <__switch_to+68> auipc t0,0x5
0x8020021c <__switch_to+72> addi t0,t0,-520
0x80200220 <__switch_to+76> sd a1,0(t0)
0x80200224 <__switch_to+80> mv t0,a1
0x80200228 <__switch_to+84> addi t0,t0,40
0x8020022c <__switch_to+88> ld ra,0(t0)
0x80200230 <__switch_to+92> ld sp,8(t0)
0x80200234 <__switch_to+96> addi t0,t0,16
0x80200238 <__switch_to+100> ld s0,0(t0)
0x8020023c <__switch_to+104> ld s1,8(t0)
0x80200240 <__switch_to+108> ld s2,16(t0)
0x80200244 <__switch_to+112> ld s3,24(t0)
0x80200248 <__switch_to+116> ld s4,32(t0)
0x8020024c <__switch_to+120> ld s5,40(t0)
0x80200250 <__switch_to+124> ld s6,48(t0)
0x80200254 <__switch_to+128> ld s7,56(t0)

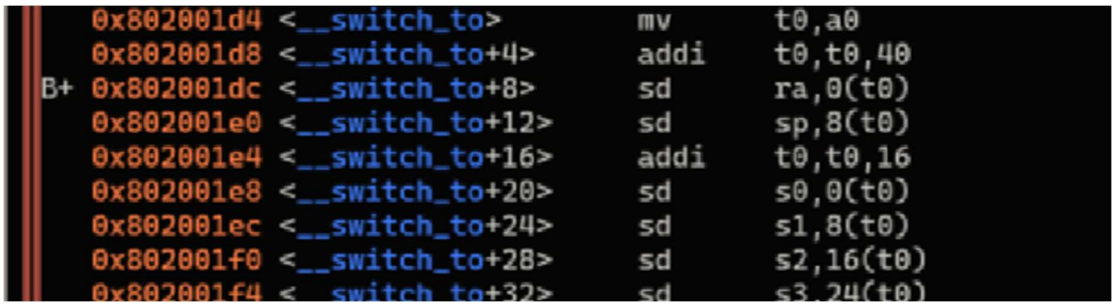
(4)

(5)

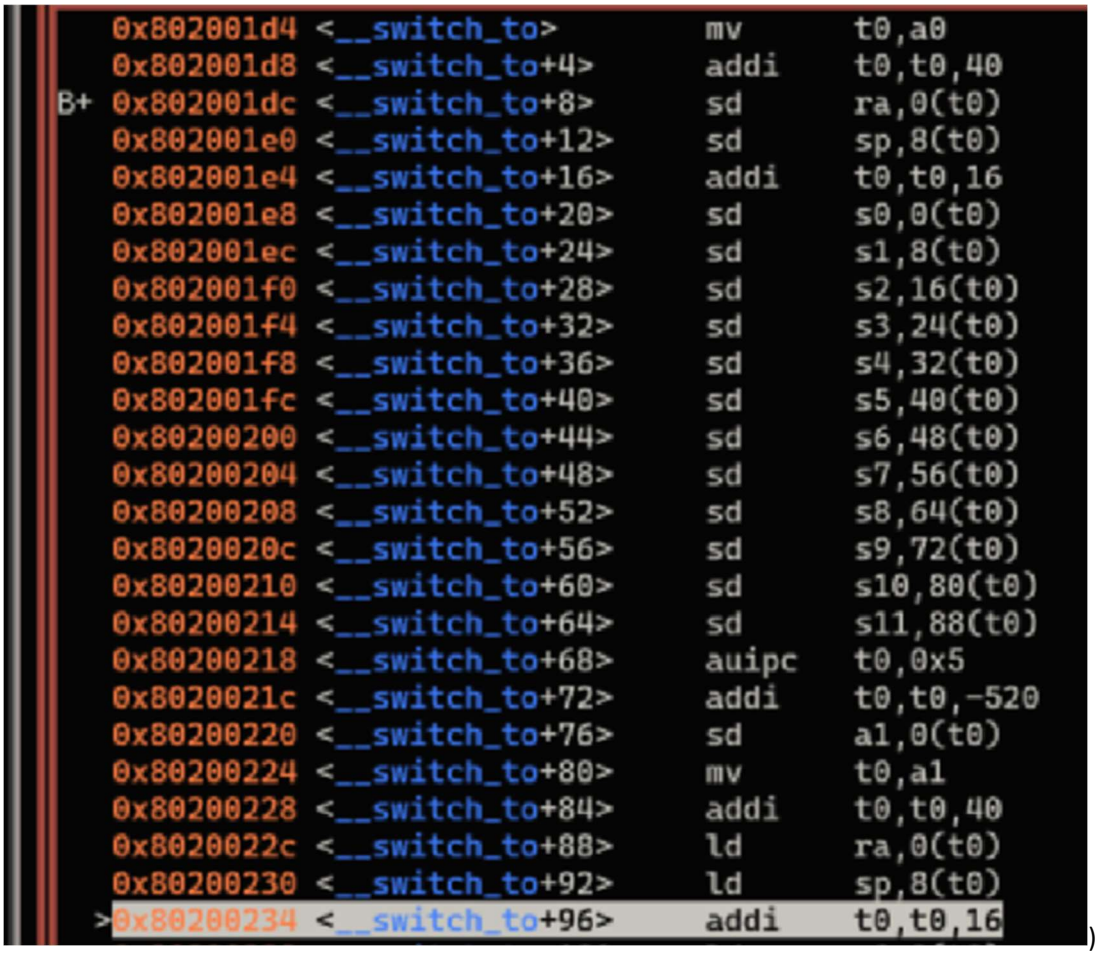
(6)


0x80200554 <schedule> addi sp,sp,-48
0x80200558 <schedule+4> sd ra,40(sp)
0x8020055c <schedule+8> sd s0,32(sp)
0x80200560 <schedule+12> sd s1,24(sp)
0x80200564 <schedule+16> addi s0,sp,48
0x80200568 <schedule+20> li a5,1
0x8020056c <schedule+24> sw a5,-36(s0)
B+> 0x80200570 <schedule+28> j 0x802005c0 <schedule+108>
0x80200574 <schedule+32> auipc a4,0x5
0x80200578 <schedule+36> addi a4,a4,-1364
0x8020057c <schedule+40> lw a5,-36(s0)
0x80200580 <schedule+44> slli a5,a5,0x3
0x80200584 <schedule+48> add a5,a4,a5
0x80200588 <schedule+52> ld a5,0(a5)
0x8020058c <schedule+56> ld a5,16(a5)
0x80200590 <schedule+60> beqz a5,0x802005b4 <schedule+96>
0x80200594 <schedule+64> auipc a4,0x5
0x80200598 <schedule+68> addi a4,a4,-1396
0x8020059c <schedule+72> lw a5,-36(s0)
0x802005a0 <schedule+76> slli a5,a5,0x3
0x802005a4 <schedule+80> add a5,a4,a5
0x802005a8 <schedule+84> ld a5,0(a5)
0x802005ac <schedule+88> sd a5,-48(s0)
0x802005b0 <schedule+92> j 0x802005d0 <schedule+124>
0x802005b4 <schedule+96> lw a5,-36(s0)
0x802005b8 <schedule+100> addiw a5,a5,1
0x802005bc <schedule+104> sw a5,-36(s0)
0x802005c0 <schedule+108> lw a5,-36(s0)
0x802005c4 <schedule+112> sext.w a4,a5
0x802005c8 <schedule+116> li a5,31
0x802005cc <schedule+120> bge a5,a4,0x80200574 <schedule+32>
0x802005d0 <schedule+124> lw a5,-36(s0)
0x802005d4 <schedule+128> sext.w a4,a5

(7)


0x802001d4 <__switch_to> mv t0,a0
0x802001d8 <__switch_to+4> addi t0,t0,40
B+ 0x802001dc <__switch_to+8> sd ra,0(t0)
0x802001e0 <__switch_to+12> sd sp,8(t0)
0x802001e4 <__switch_to+16> addi t0,t0,16
0x802001e8 <__switch_to+20> sd s0,0(t0)
0x802001ec <__switch_to+24> sd s1,8(t0)
0x802001f0 <__switch_to+28> sd s2,16(t0)

(8



)

心得：

本次实验的难度跟上几次感觉根本不是一个等级，中间多次出现 bug 和未知情况，并且需要参展 stackoverflow 和 github 上的资料进行修改，而且对于 entry.S 的修改也是很困难的，一开始根本没有修改思路，虽然这次仅仅需要编写两个文件，但是这两个文件需要的精力需要之前二十个的精力，对于 sp/ra/epc 等寄存器值的处理很麻烦。