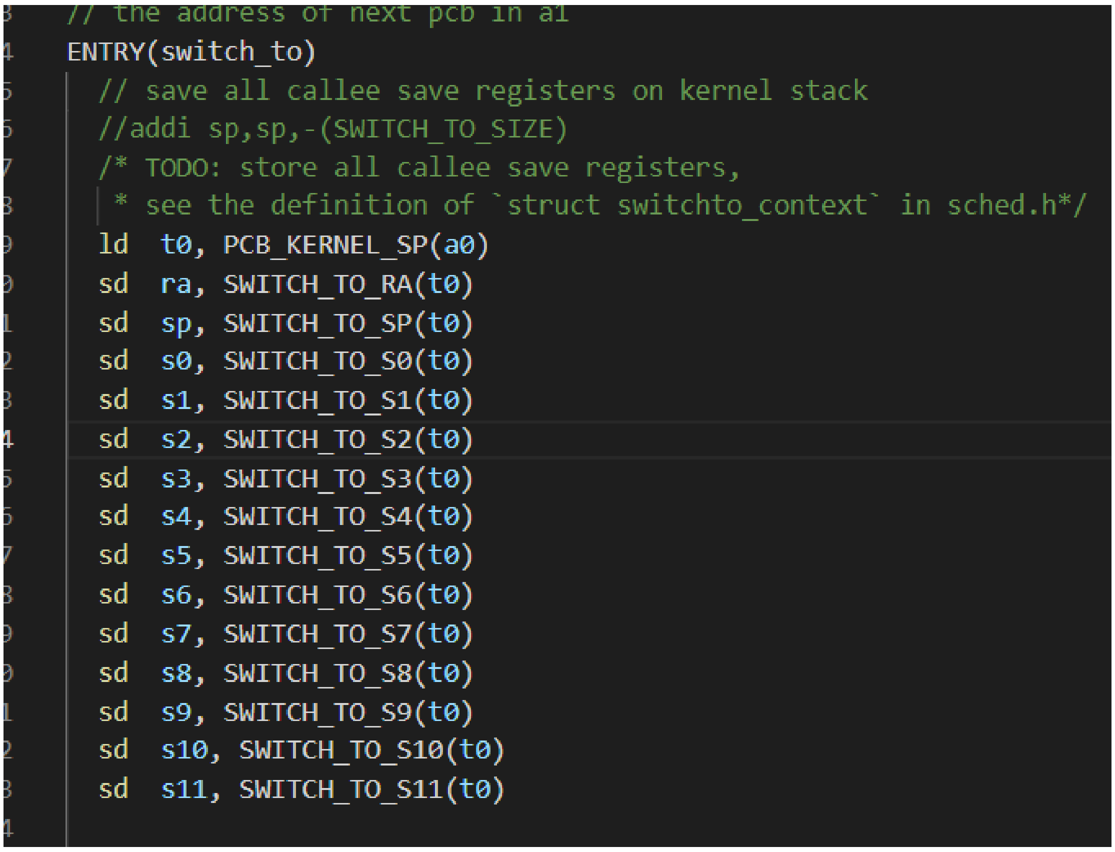
操作系统 Project2——A simple kernel

胡力杭 2019k8009926002

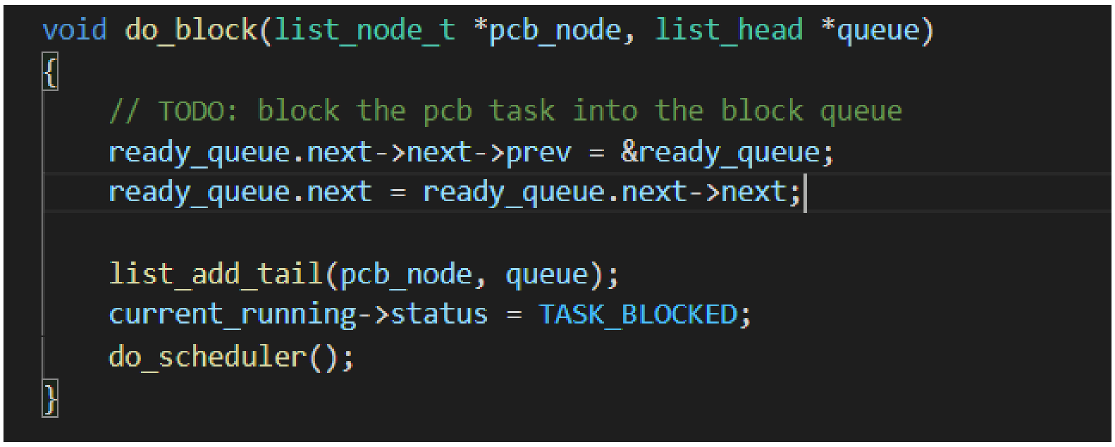
1、truly illegal instruiton

这次实验因为需要不断地保存寄存器、恢复寄存器，所以 load 指令、store 指令会比较多。如果在某一些部分没有写正确或者是读正确，那很有可能会跳转到一些莫名其妙的位置，这也就是为什么会产生相当多的 truly illegal 指令。



```
// the address of next pcb in a1
ENTRY(switch_to)
  // save all callee save registers on kernel stack
  //addi sp,sp,-(SWITCH_TO_SIZE)
  /* TODO: store all callee save registers,
   * see the definition of `struct switchto_context` in sched.h*/
  ld  t0, PCB_KERNEL_SP(a0)
  sd  ra, SWITCH_TO_RA(t0)
  sd  sp, SWITCH_TO_SP(t0)
  sd  s0, SWITCH_TO_S0(t0)
  sd  s1, SWITCH_TO_S1(t0)
  sd  s2, SWITCH_TO_S2(t0)
  sd  s3, SWITCH_TO_S3(t0)
  sd  s4, SWITCH_TO_S4(t0)
  sd  s5, SWITCH_TO_S5(t0)
  sd  s6, SWITCH_TO_S6(t0)
  sd  s7, SWITCH_TO_S7(t0)
  sd  s8, SWITCH_TO_S8(t0)
  sd  s9, SWITCH_TO_S9(t0)
  sd  s10, SWITCH_TO_S10(t0)
  sd  s11, SWITCH_TO_S11(t0)
```

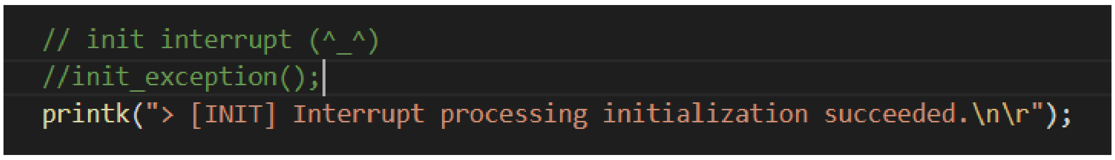
使用 t0 而不是 sp



```
void do_block(list_node_t *pcb_node, list_head *queue)
{
    // TODO: block the pcb task into the block queue
    ready_queue.next->next->prev = &ready_queue;
    ready_queue.next = ready_queue.next->next;

    list_add_tail(pcb_node, queue);
    current_running->status = TASK_BLOCKED;
    do_scheduler();
}
```

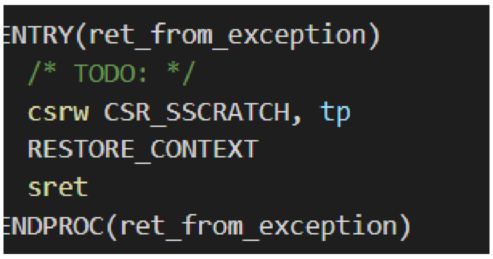
未将 block 的进程移出队列



```
// init interrupt (^_^)
//init_exception();
printk("> [INIT] Interrupt processing initialization succeeded.\n\r");
```

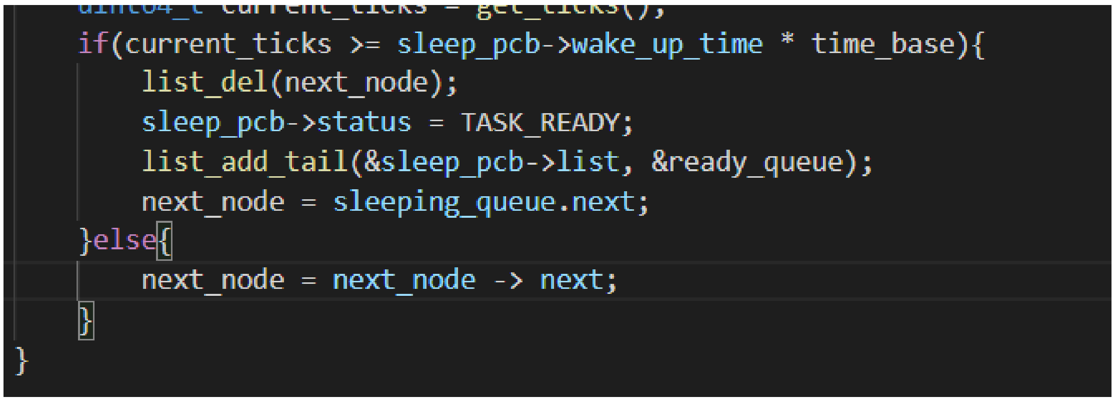
该函数未完成，需要注释

Part2：



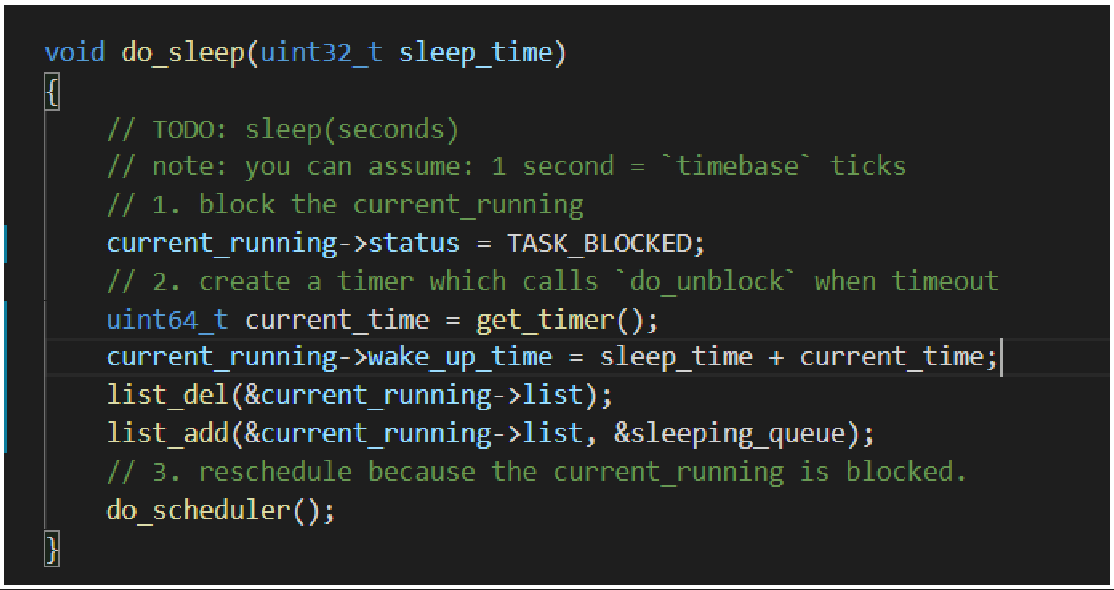
```
NTRY(ret_from_exception)
  /* TODO: */
  csrw CSR_SSCRATCH, tp
  RESTORE_CONTEXT
  sret
NDPROC(ret_from_exception)
```

Csrw 和 restore 写反了。按照讲义，我们在恢复现场之前需要将用户态的 tp 存到 sscratch 中，然后将 sscratch 中的值放至存 tp 的栈中。



```
    uint64_t current_ticks = get_ticks();
    if(current_ticks >= sleep_pcb->wake_up_time * time_base){
        list_del(next_node);
        sleep_pcb->status = TASK_READY;
        list_add_tail(&sleep_pcb->list, &ready_queue);
        next_node = sleeping_queue.next;
    }else{
        next_node = next_node -> next;
    }
}
```

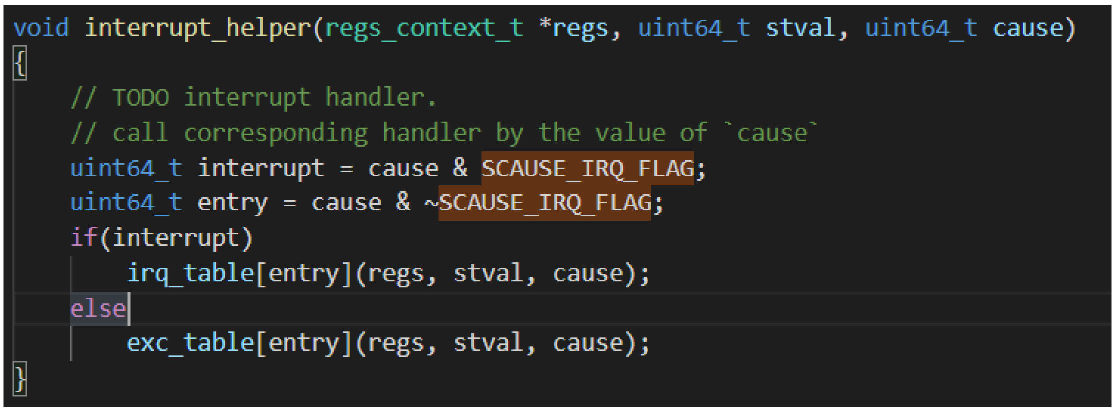
在书写 check_sleep 的时候对 next_node 的处理不正确。开始写的时候，if 第一种情况下也是 next_node = next_node -> next，但是此时 next_node 已经被移出了队列，会读取到一个空指针，造成了错误。



```
void do_sleep(uint32_t sleep_time)
{
    // TODO: sleep(seconds)
    // note: you can assume: 1 second = `timebase` ticks
    // 1. block the current_running
    current_running->status = TASK_BLOCKED;
    // 2. create a timer which calls `do_unblock` when timeout
    uint64_t current_time = get_timer();
    current_running->wake_up_time = sleep_time + current_time;
    list_del(&current_running->list);
    list_add(&current_running->list, &sleeping_queue);
    // 3. reschedule because the current_running is blocked.
    do_scheduler();
}
```

do_sleep 的时候将 `current_running->status = TASK_BLOCKED;`

```
写成了 current_running = TASK_BLOCKED;
```



```c
void interrupt_helper(regs_context_t *regs, uint64_t stval, uint64_t cause)
{
    // TODO interrupt handler.
    // call corresponding handler by the value of `cause`
    uint64_t interrupt = cause & SCAUSE_IRQ_FLAG;
    uint64_t entry = cause & ~SCAUSE_IRQ_FLAG;
    if(interrupt)
        irq_table[entry](regs, stval, cause);
    else
        exc_table[entry](regs, stval, cause);
}
```

Interrupt 变量使用了 32 位的 int，导致了位宽不够。如果是时钟中断的话，原本 interrupt 变量的值应该为 0x80000000，但是因为是 int 类型，值变为 0，产生了错误。应该使用 64 位的 uint64_t