Lecture 2 A Simple Kernel

2025.10.29

中国科学院大学
University of Chinese Academy of Sciences

Schedule

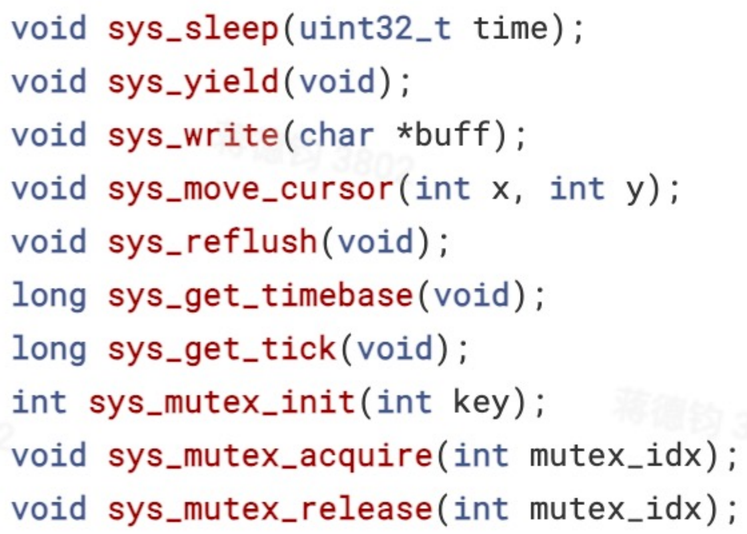
- Project 2 part2 assignment
- Project 2 part1 due

Project 2 – A Simple Kernel

- Requirement II
  - Write a simple kernel (preemptive)
    - Provide preemptive scheduling supporting exception handler, including
      - System call handler
      - Clock interrupt handler
      - Round-robin scheduler
    - A complex scheduler (optional for C-Core)

Project 2 A Simple Kernel

- A set of user process
  - User processes call syscalls to invoke kernel services
  - Please refer to unistd.h(tiny_libc/include) and syscall.c (tiny_libc/)



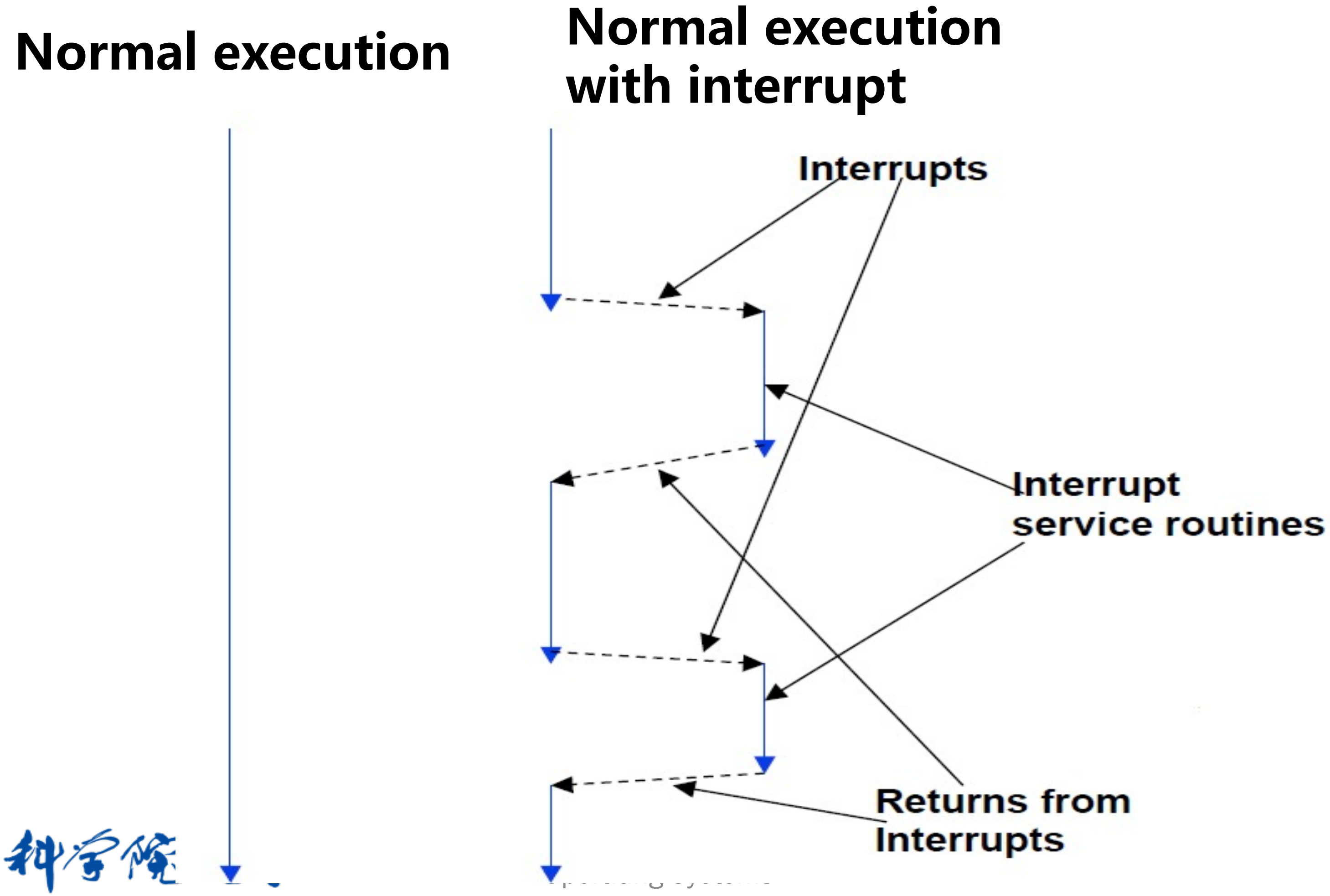
```
void sys_sleep(uint32_t time);
void sys_yield(void);
void sys_write(char *buff);
void sys_move_cursor(int x, int y);
void sys_reflush(void);
long sys_get_timebase(void);
long sys_get_tick(void);
int sys_mutex_init(int key);
void sys_mutex_acquire(int mutex_idx);
void sys_mutex_release(int mutex_idx);
```

Project 2 A Simple Kernel

- Interrupt
  - A signal to the processor emitted by hardware indicating an event requiring immediate process
  - The processor responds by suspending its current running task, saving its state, and executing interrupt handler
  - After the interrupt handler finishes, the processor resumes normal activities

Project 2 A Simple Kernel

- Interrupt



Normal execution

Normal execution with interrupt

Interrupts

Interrupt service routines

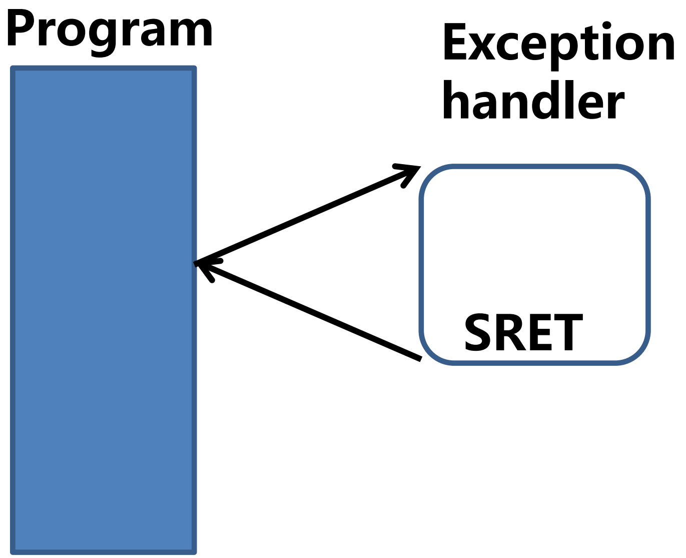
Returns from Interrupts

Project 2 A Simple Kernel

- Hardware interrupt
  - A change in execution caused by a hardware event
    - Timer interrupt for timesharing
    - I/O device interrupt: disk, network, keyboard etc.
- We do not handle other interrupts here
  - Page fault, TLB miss etc.
- Trap
  - System calls

Project 2 A Simple Kernel

- Handling exception
  - Save context
  - Determine what causes exception
  - Invoke specific routine based on type of exception
  - Restore context
  - Return from exception

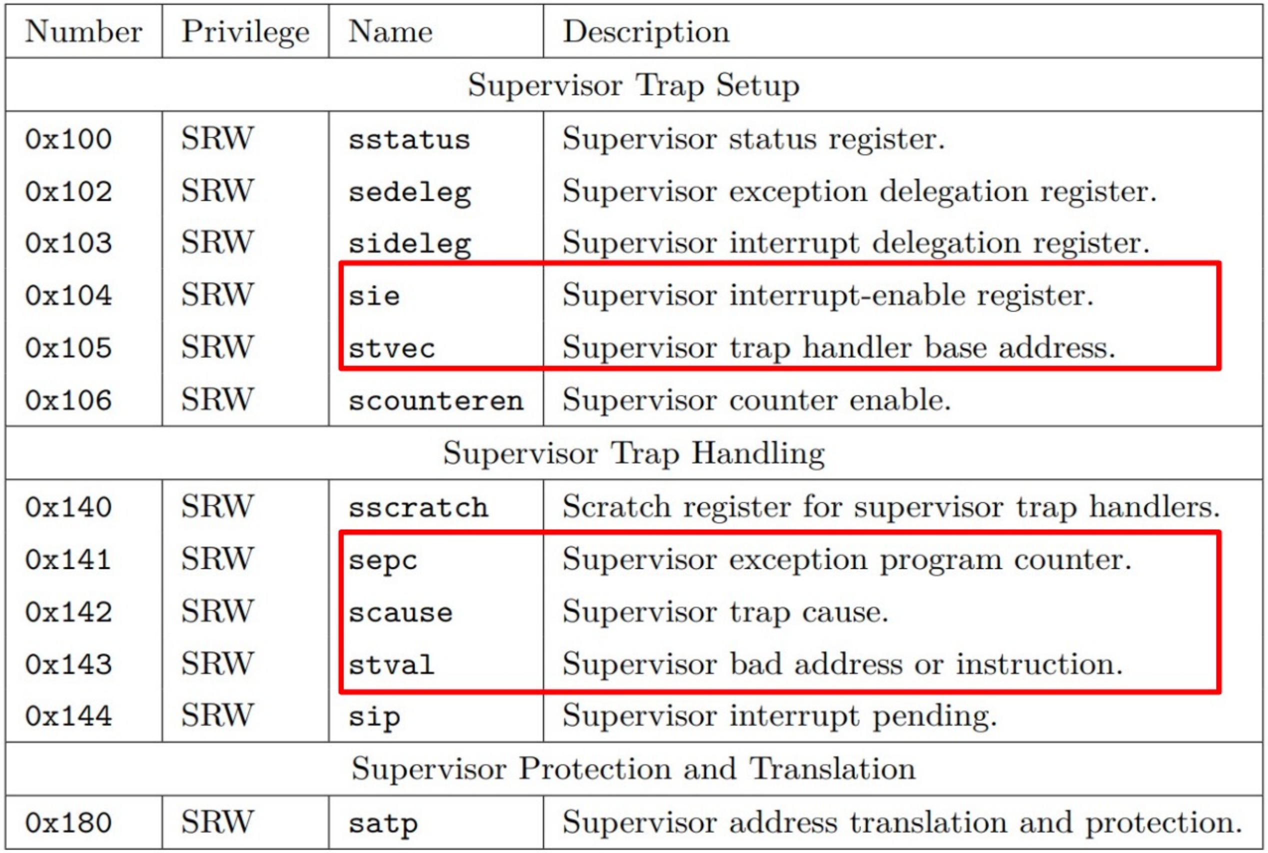


Project 2 A Simple Kernel

- Control Status Registers(CSRs)
  - We need to manipulate CSRs when implementing interrupt handler
- Two types of CSRs
  - One type is for setting up interrupt
  - The other is for handling interrupt
  - Pls. refer to Table P2-2 in guidebook

Project 2 A Simple Kernel

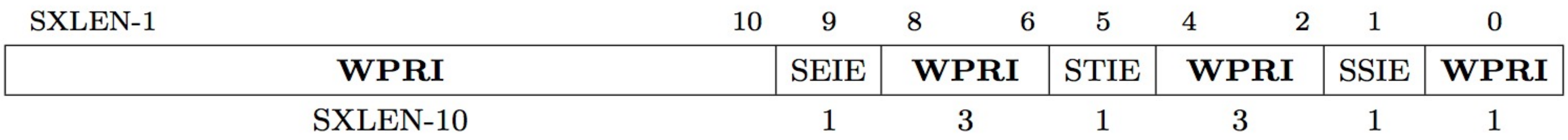
- Control Status Registers(CSRs)



Number	Privilege	Name	Description
Supervisor Trap Setup			
0x100	SRW	`sstatus`	Supervisor status register.
0x102	SRW	`sedeleg`	Supervisor exception delegation register.
0x103	SRW	`sideleg`	Supervisor interrupt delegation register.
0x104	SRW	`sie`	Supervisor interrupt-enable register.
0x105	SRW	`stvec`	Supervisor trap handler base address.
0x106	SRW	`scounteren`	Supervisor counter enable.
Supervisor Trap Handling			
0x140	SRW	`sscratch`	Scratch register for supervisor trap handlers.
0x141	SRW	`sepc`	Supervisor exception program counter.
0x142	SRW	`scause`	Supervisor trap cause.
0x143	SRW	`stval`	Supervisor bad address or instruction.
0x144	SRW	`sip`	Supervisor interrupt pending.
Supervisor Protection and Translation			
0x180	SRW	`satp`	Supervisor address translation and protection.

Project 2 A Simple Kernel

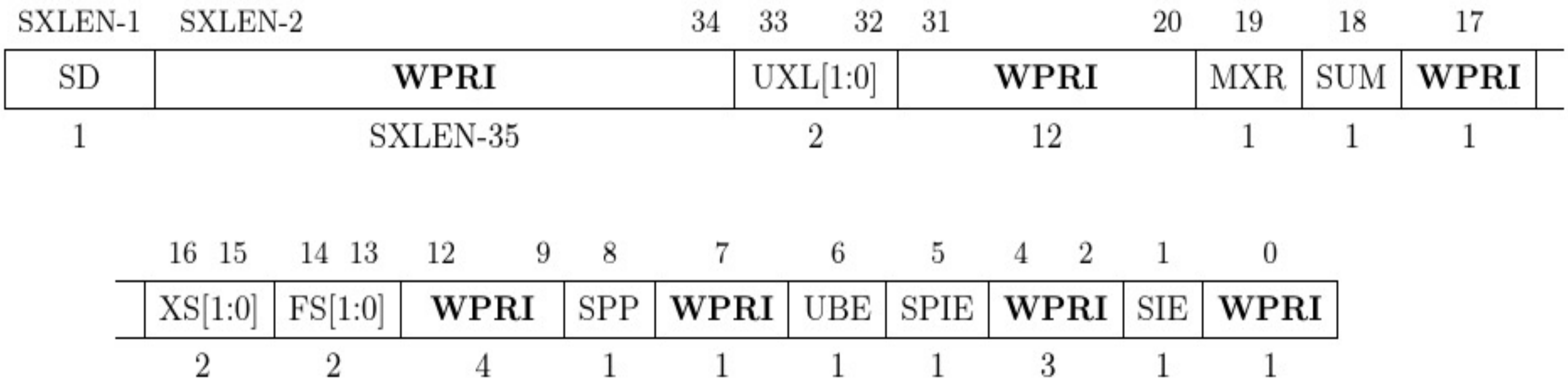
- Setup interrupt
  - sie register
  - STIE(Supervisor-level timer interrupt enable)
  - Use enable_preempt (arch/riscv/kernel/entry.S) to set up timer interrupt



SXLEN-1 10	9	8 6	5	4 2	1	0
WPRI	SEIE	**WPRI**	STIE	**WPRI**	SSIE	**WPRI**
SXLEN-10	1	3	1	3	1	1

Project 2 A Simple Kernel

- Setup interrupt
  - sstatus register



SXLEN-1	SXLEN-2 34	33 32	31 20	19	18	17
SD	**WPRI**	UXL[1:0]	**WPRI**	MXR	SUM	**WPRI**
1	SXLEN-35	2	12	1	1	1

16 15	14 13	12 9	8	7	6	5	4 2	1	0
XS[1:0]	FS[1:0]	**WPRI**	SPP	**WPRI**	UBE	SPIE	**WPRI**	SIE	**WPRI**
2	2	4	1	1	1	1	3	1	1

Project 2 A Simple Kernel

- Setup interrupt
  - SIE (Supervisor Interrupt Enable) bit enables or disables all interrupts in supervisor mode
  - SPIE (Supervisor Previous Interrupt Enable) bit indicates whether supervisor interrupts were enabled prior to trapping into supervisor mode.
  - SPP (Supervisor Previous Privilege Mode) bit indicates the privilege mode before the trapping
  - You need to correctly initialize SPIE(1), SPP(0) in sstatus register

Project 2 A Simple Kernel

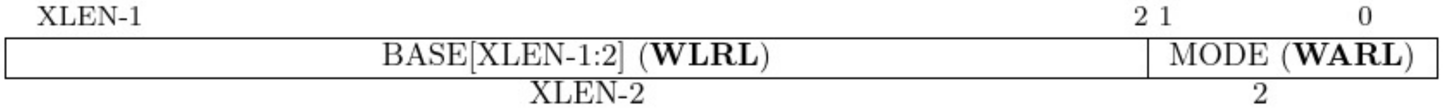
- Manipulate CSR registers
  - csrr: read csr
    - Loads data from a CSR register into a CPU register
  - csrw: write csr
    - Stores data into a CSR register
  - csrc/csrs：clear/set a CSR register’s corresponding bits
- Examples
  - csrr a0, sip
  - csrw stvec, t0

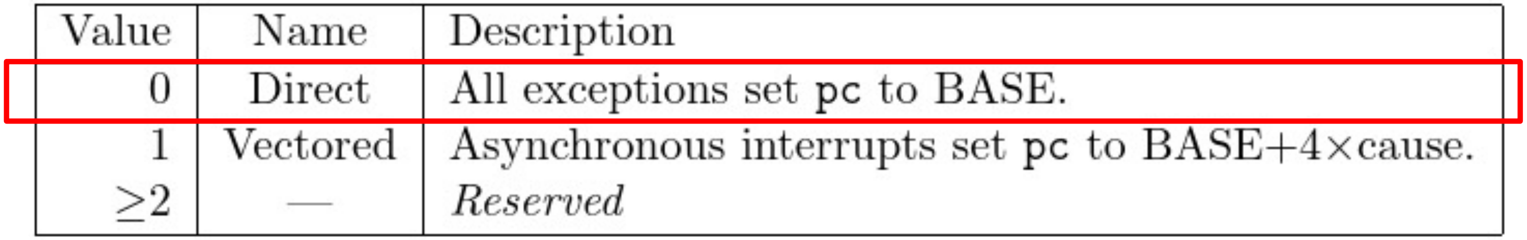
Project 2 A Simple Kernel

- What if interrupt occurs while in interrupt handler?
  - The processor automatically disables all interrupts when an interrupt occurs
  - The processor automatically re-enables all interrupts after sret is called

Project 2 A Simple Kernel

- Entry to handle exception
  - stvec register
  - BASE field hold the exception handler’ s address
  - exception_handler_entry in entry.S is the entry point for handling exceptions

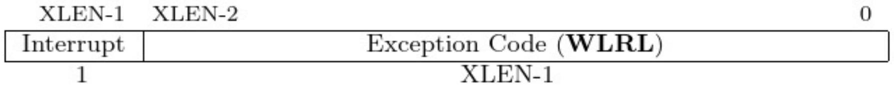




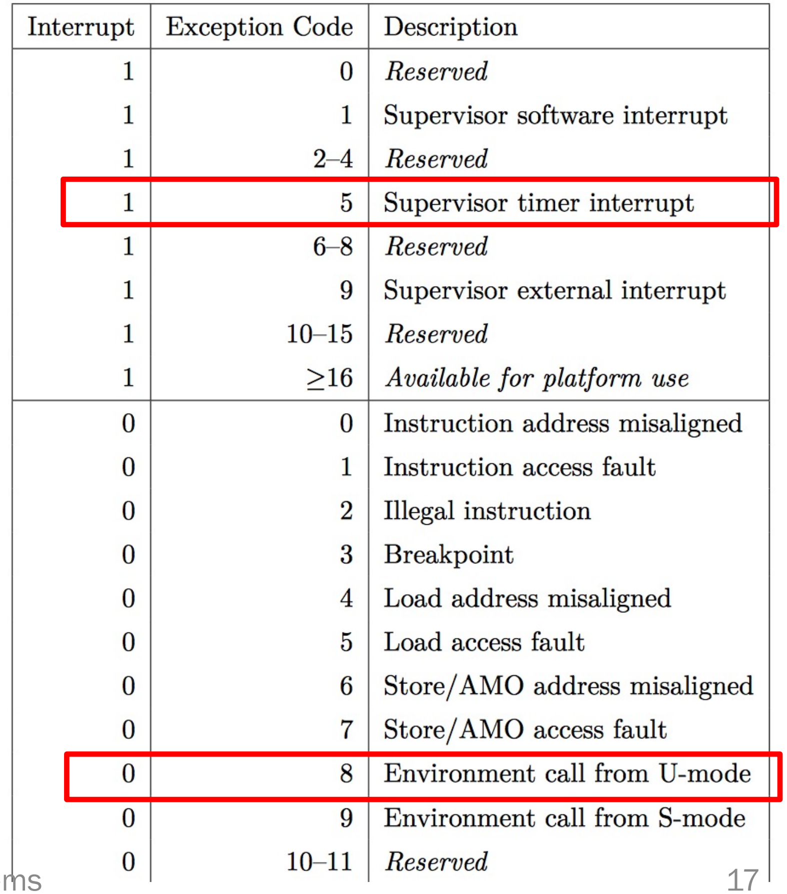
Value	Name	Description
0	Direct	All exceptions set pc to BASE.
1	Vectored	Asynchronous interrupts set pc to BASE+4×cause.
≥2	—	*Reserved*

Project 2 A Simple Kernel

- Identifying exception type
  - scause register
  - pls. refer to Table P2-3
  - When exception occurs, uses scause register to identify exception type



XLEN-1	XLEN-2 ... 0
Interrupt	Exception Code (**WLRL**)
1	XLEN-1



Interrupt	Exception Code	Description
1	0	*Reserved*
1	1	Supervisor software interrupt
1	2–4	*Reserved*
1	5	Supervisor timer interrupt
1	6–8	*Reserved*
1	9	Supervisor external interrupt
1	10–15	*Reserved*
1	≥16	*Available for platform use*
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10–11	*Reserved*

Project 2 A Simple Kernel

- Return from exception
  - sepc register
  - The PC address when exception occurs is stored in sepc
  - sret can return to the address in sepc
  - Please pay attention to the sepc address when handling syscall
    - sret should return to sepc+4

Project 2 A Simple Kernel

- Support process sleep
  - Blocking sleep
    - Block the task when it calls sleep()
    - Use a separate queue to keep sleeping tasks
  - Wake up the task
    - When the timing reaches sleeping threshold of the task
  - About timing
    - We provide timing related functions in time.c
    - Use these functions to decide whether sleeping ends

Project 2 A Simple Kernel

- Implement timer interrupt handler
  - Deal with normal tasks
    - Schedule based on your scheduling policy
    - Round Robin policy is OK
  - Deal with sleeping tasks
    - Check whether waking up the task
  - Reset timer using bios_set_timer (include/os/bios.h)
  - CPU frequency is obtained using bios_read_fdt in main.c

Project 2 A Simple Kernel

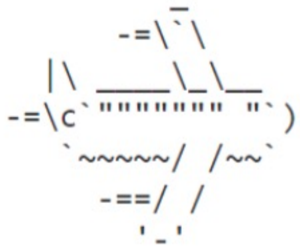
- Step by step – Task3
  - Implement syscalls
  - Support do_sleep
  - In this task, we do not have timer interrupt, you need to consider when to check sleeping time

Project 2 A Simple Kernel

- Step by step – Task 4
  - Implement timer interrupt handler
  - Implement round robin scheduler

Project 2 A Simple Kernel

- Step by step – Task 5
  - There exist five airplanes with different moving speed
  - Please design a scheduling policy to allow the five airplanes reach the checkpoint simultaneously as well as reach the end simultaneously
    - e.g. the slow airplane gets longer time slice than the fast one



```
         _
     -=\`\
 |\ ____\_\__
-=\c`""""""" "`)
   `~~~~~/ /~~`
     -==/ /
       '-'
```

A note to “loadbootd”

- Open CPU in DASIC mode
  - Prevent user code jump/call kernel code
  - Prevent user code access kernel data
  - Provide memory protection before virtual memory is available
  - Run both in QEMU and Pynq
- A/C core should pass loadbootd check

Project 2 A Simple Kernel

- Requirement for design review
  - 请描述syscall的执行流程，以及开发syscall时，如何实现参数传递？
  - 在初始化中断处理时，需要设置哪些寄存器，具体初始化为什么值？
  - 当进程因异常进入内核时，需要进行哪些上下文保存？
  - 开发任务3时，init_pcb_stack函数需要完成哪些初始化操作？
  - 任务3和4中，分别何时唤醒睡眠的进程？
  - Any questions you are not clear enough

Project 2 – A Simple Kernel

- Requirement for S/A/C-Core

Core type	Task requirements
S-Core	Tasks 1, 2,3-（只用打印例外报错信息即可）
A-Core	Tasks 1,2,3,4
C-Core	Tasks 1,2,3,4,5

Project 2 – A Simple Kernel

- P2-2 schedule
  - 5th Nov.
    - P2 part2 design review
  - 12th Nov.
    - P2 part2 due