# 操作系统研讨课

蒋德钧

Fall Term 2017-2018

email: jiangdejun@ict.ac.cn
office phone: 62601007

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

# Lecture 3 Preemptive Kernel

2017.10.18

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

# Schedule

- Project 2 due
- Project 3 assignment

# Project 2 Due

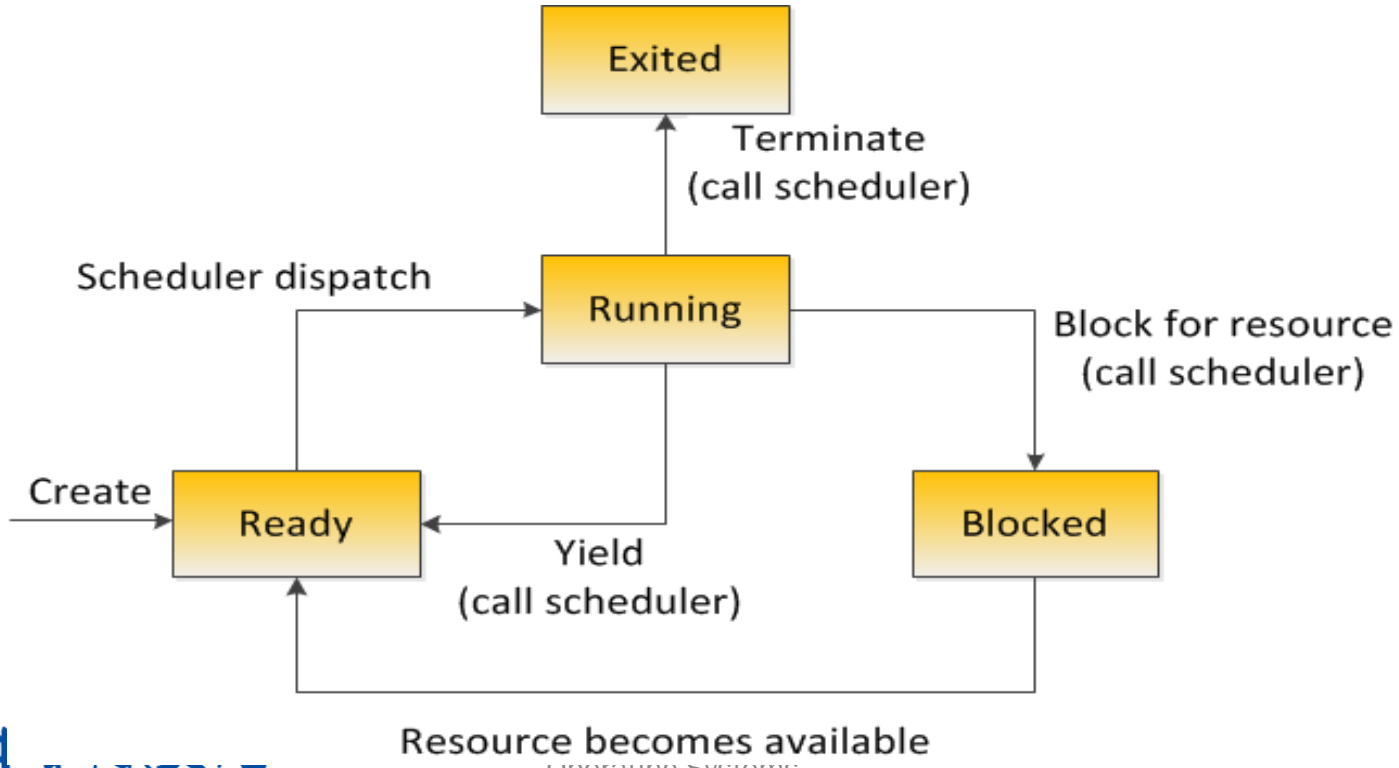
- P2 review
  - We test mutual lock, context switch cost measurement, and all tests in tasks.c for P2 due
  - Please compile your code, running the code on your board, and show the results to TA
  - Answer any questions we may ask

# Project 3 Preemptive Kernel

- Requirement
  - Support preemptive kernel
    - Write a timer interrupt handler
    - Write functions to support block sleeping
    - Write a scheduler to support both round robin and priority based scheduling

# Project 3 Preemptive Kernel

- Non-Preemptive kernel
  - How about the throughput of the operating system?

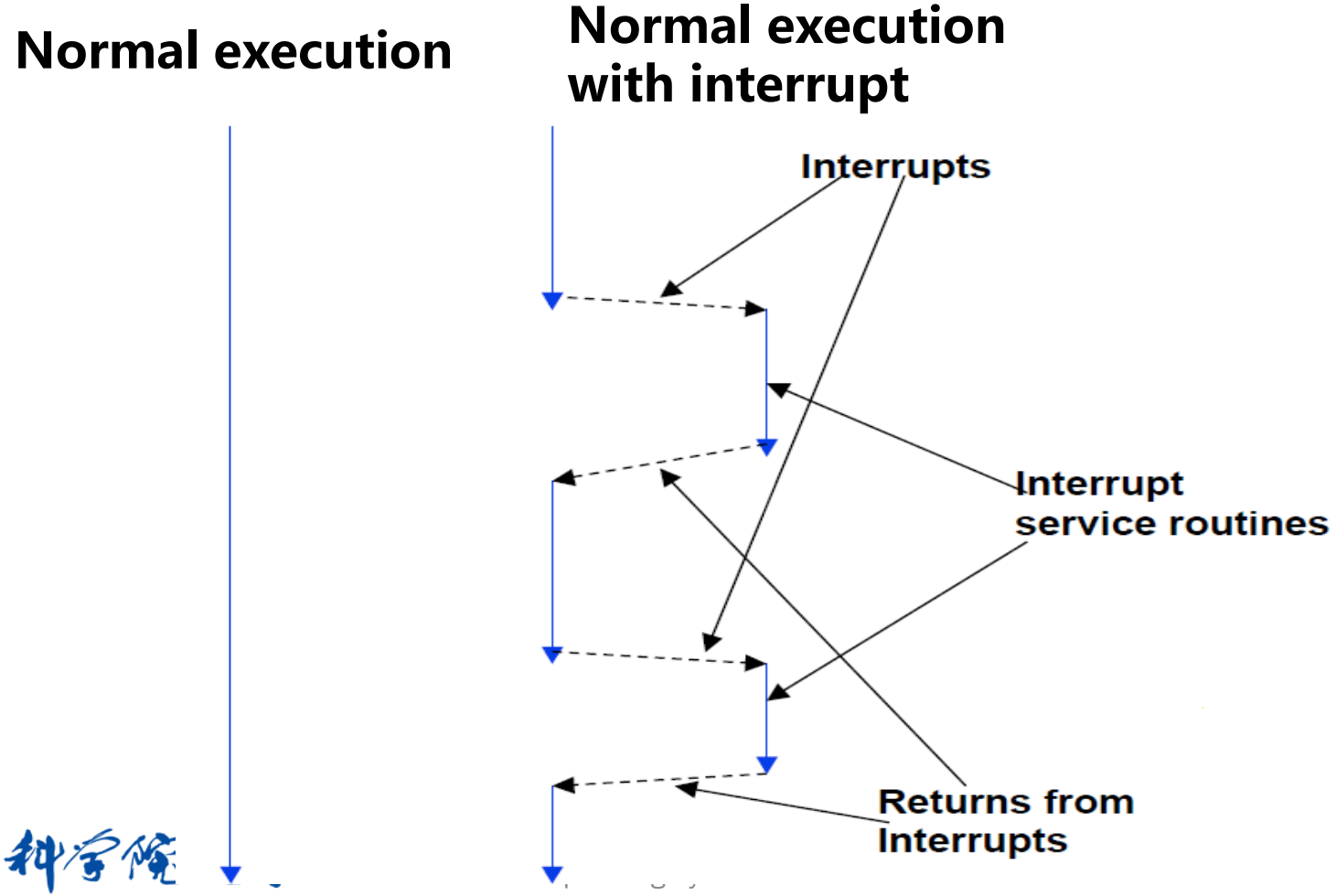


# Project 3 Preemptive Kernel

- Interrupt & Exception & Trap
  - A signal to the processor emitted by HW or SW indicating an event requiring immediate process
  - The processor responds by suspending its current running task, saving its state, and executing interrupt/exception handler
  - After the interrupt/exception handler finishes, the processor resumes normal activities

# Project 3 Preemptive Kernel

- Interrupt & Exception & Trap

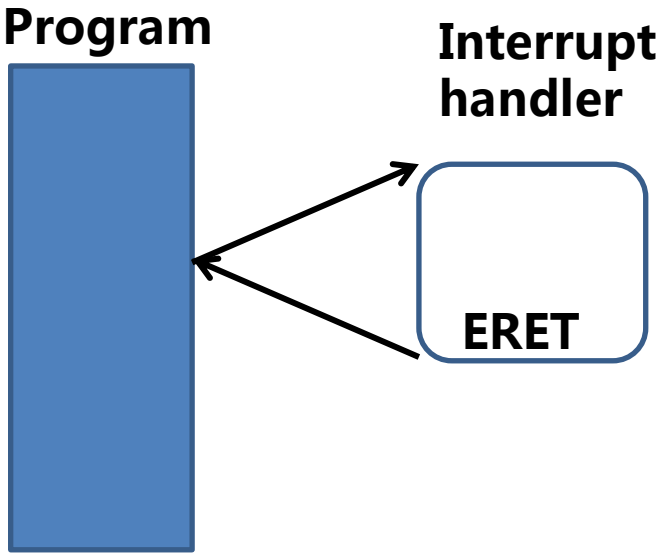


# Project 3 Preemptive Kernel

- Interrupt
  - A change in execution caused by an external event outside the processor
    - Clock for timesharing
    - I/O devices: disk, network, keyboard etc.
- Exception
  - A change in execution caused by an internal event within the processor
    - Segment fault, Overflow, Page fault etc.
- Trap
  - A user-requested exception
    - Syscall

# Project 3 Preemptive Kernel

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

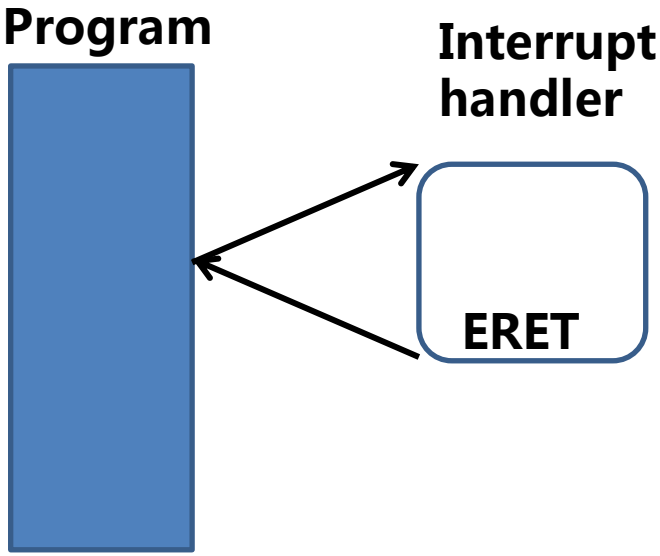


# Project 3 Preemptive Kernel

- Handling interrupt
  - What if interrupt occurs while in interrupt handler?
    - ENTER_CRITICAL
      - Disable interrupt before actually handling the interrupt
      - Setting a label to indicate the interrupt is disabled
    - LEAVE_CRITICAL
      - Do not forget to re-enable interrupt after handling the interrupt

# Project 3 Preemptive Kernel

- Handling interrupt
  - ENTER_CRITICAL
  - Save context
  - Determine what caused interrupt
  - Invoke specific routine based on type of interrupt
  - LEAVE_CRITICAL
  - Restore context
  - Return from interrupt



# Project 3 Preemptive Kernel

- Tips on implementing interrupt handler
  - Coprocessor 0 (CP0) registers
    - Status register
    - IE bit: 1 enable interrupt; 0 disable
    - IM7 ~ IM0: control enable/disable different interrupts
    - IM7 bit: 1: enable Clock interrupt; 0: disable

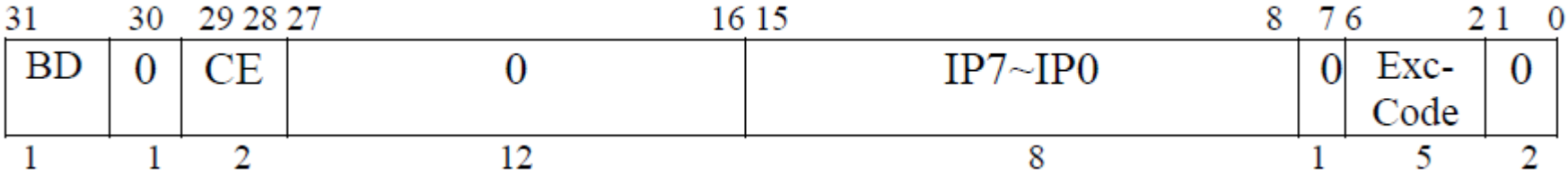
| 31 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 16 | 15 8 | 7 5 | 4 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CU (cu3:cu0) | 0 | FR | 0 | NO-FDIV | NO-FSQR | BEV | 0 | SR | 0 | IM7-IM0 | 0 | KSU | ERL | EXL | IE |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 4 | 8 | 3 | 2 | 1 | 1 | 1 |

# Project 3 Preemptive Kernel

- Tips on implementing interrupt handler
  - Coprocessor 0 (CP0) registers
    - Status register
    - Refer to the initialization of interrupts in start code
    - Pay attention to the initialization of the register in YOUR PCB
    - We only handle clock interrupt in this project

# Project 3 Preemptive Kernel

- Tips on implementing interrupt handler
  - Coprocessor 0 (CP0) registers
    - Cause register
    - IP7 ~ IP0: indicating the interrupt type
    - IP7 bit: 1: Clock interrupt
    - EXCCODE: Exception type



# Project 3 Preemptive Kernel

- Tips on implementing interrupt handler
  - Coprocessor 0 (CP0) registers
    - BadVAddr register
    - EPC register
    - TagHi/TagLo registers
  - Please keep these registers in your PCB

# Project 3 Preemptive Kernel

- Tips on implementing interrupt handler
  - Operate CP0 registers
    - mfc0: move from  Coprocessor 0
      - Loads data from a CP0 register into a CPU register
    - mtc0: move to Coprocessor 0
      - Stores data into a CP0 register
    - mfhi/mthi
    - mflo/mtlo
  - Use MIPS registers
    - $k0, $k1

# Project 3 Preemptive Kernel

- Tips on implementing interrupt handler
  - Coprocessor 0 (CP0): read-modify-write
    - Read the cp0 register into a CPU register
    - Modify the contents of the CPU register
    - Write the modified value back to the cp0 register
    - Please refer MACRO STI and CLI in start code
    - E.g. mfc0 k0, CP0_STATUS
      mtc0 k0, CP0_STATUS

# Project 3 Preemptive Kernel

- Tips on implementing interrupt handler
  - What do you do in the clock interrupt handler?
    - How to deal with normal tasks with yielding?
    - How to deal with sleeping tasks?
  - Different operations for user process and kernel threads
    - nested_count field
    - Yield if in user mode
    - Continue running if in kernel mode

# Project 3 Preemptive Kernel

- Tips on disabling/enabling interrupt
  - Get to understand the following macros
    - STI, CLI
    - ENTER_CRITICAL, LEAVE_CRITICAL
  - Get to understand the following leaf functions
    - enter_critical()
    - leave_critical()

# Project 3 Preemptive Kernel

- 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
    - Use the number of ticks
    - Time slice increases in each clock interrupt

# Project 3 Preemptive Kernel

- Scheduler
  - Round robin
  - Priority based
  - Fairness
  - Think about what kind of information should be included in PCB if you want to do the above scheduler

# Project 3 Preemptive Kernel

- Step by step
  - Task 1: implement a clock interrupt handler, blocking sleep, and round-robin scheduler
  - Task 2: implement a priority based scheduler

# Project 3 Preemptive Kernel

- Requirements for design review (40 points)
  - What is the workflow for handling interrupt?
  - What do you do in the clock interrupt handler?
  - How do you implement blocking sleep? When to wake up the task?
  - How do you implement the priority based scheduler?

# Project 3 Preemptive Kernel

- Requirements of developing (60 points)
  - Implement clock interrupt handler with round robin scheduler (30)
  - Implement blocking sleep (15)
  - Implement a priority-based scheduler (15)

# Project 3 Preemptive Kernel

- P3 schedule
  - P3 design review: 25th Oct.
  - No class on 1st Nov.
  - P3 due: 8th Nov.