操作系统研讨课

蒋德钧

Fall Term 2019-2020

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

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

Lecture 2 A Simple Kernel
(Part I)

2019.09.16

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

Schedule

- Project 1 due
- Project 2 part I assignment

Project 1 Due

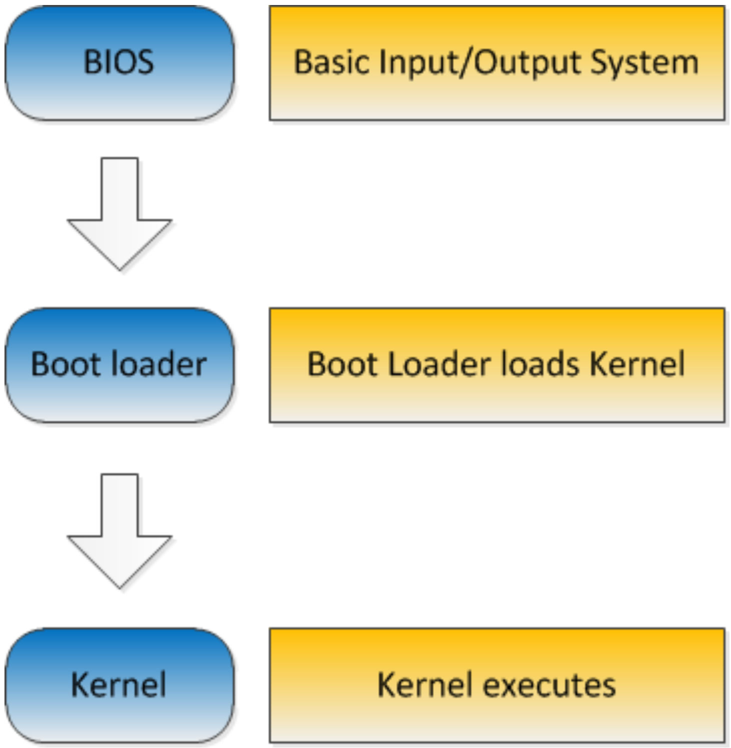
- P1 due
  - We test Task 3 for P1 due
  - Download the large kernel and test your createimage
    - 课程网站 – 资源 – 测试
  - Please compile your code, running the code on your board, and show the results to TA
  - Answer any questions we may ask

Project 1 Due

- P1 submission
  - Submit a compressed package named as “StudentNo.-YourName-P1”
  - Please includes
    - Source code
    - README to simply describe your code, e.g. which file is your work or how to run your code
    - Design document covering the questions in the design document template
  - Do not forget to submit before 23:55 TONIGHT

Project 2 – A Simple Kernel

- Booting procedure



BIOS

Basic Input/Output System

Boot loader

Boot Loader loads Kernel

Kernel

Kernel executes

Project 2 – A Simple Kernel

- Requirements
  - Write a simple kernel (non-preemptive)
    - Start a set of user processes and kernel threads
    - Perform context switches between processes and threads
    - Provide non-preemptive kernel support with context switch
    - Support basic mutex to allow BLOCK state of processes/threads

Project 2 – A Simple Kernel

- A set of multiple tasks
  - Program codes under the *test* directory in start-code
  - Please refer to *test.c* for different groups of tasks
  - Fixed number of tasks for each test group
  - Allocate per-task state statically in main.c
  - STRONGLY suggest to first read the codes of different tasks to understand what they do

Project 2 – A Simple Kernel

- Process Control Block (PCB/TCB)
  - A data structure in OS kernel containing the information to manage a particular process/thread
  - Normally, kept in an area of memory protected from normal user access

Project 2 – A Simple Kernel

- Process Control Block (PCB/TCB)
  - Process status
    - Status of a process when it is suspended
    - Contents of registers, stack pointers etc.
  - Process scheduling info
    - e.g. priority

Project 2 – A Simple Kernel

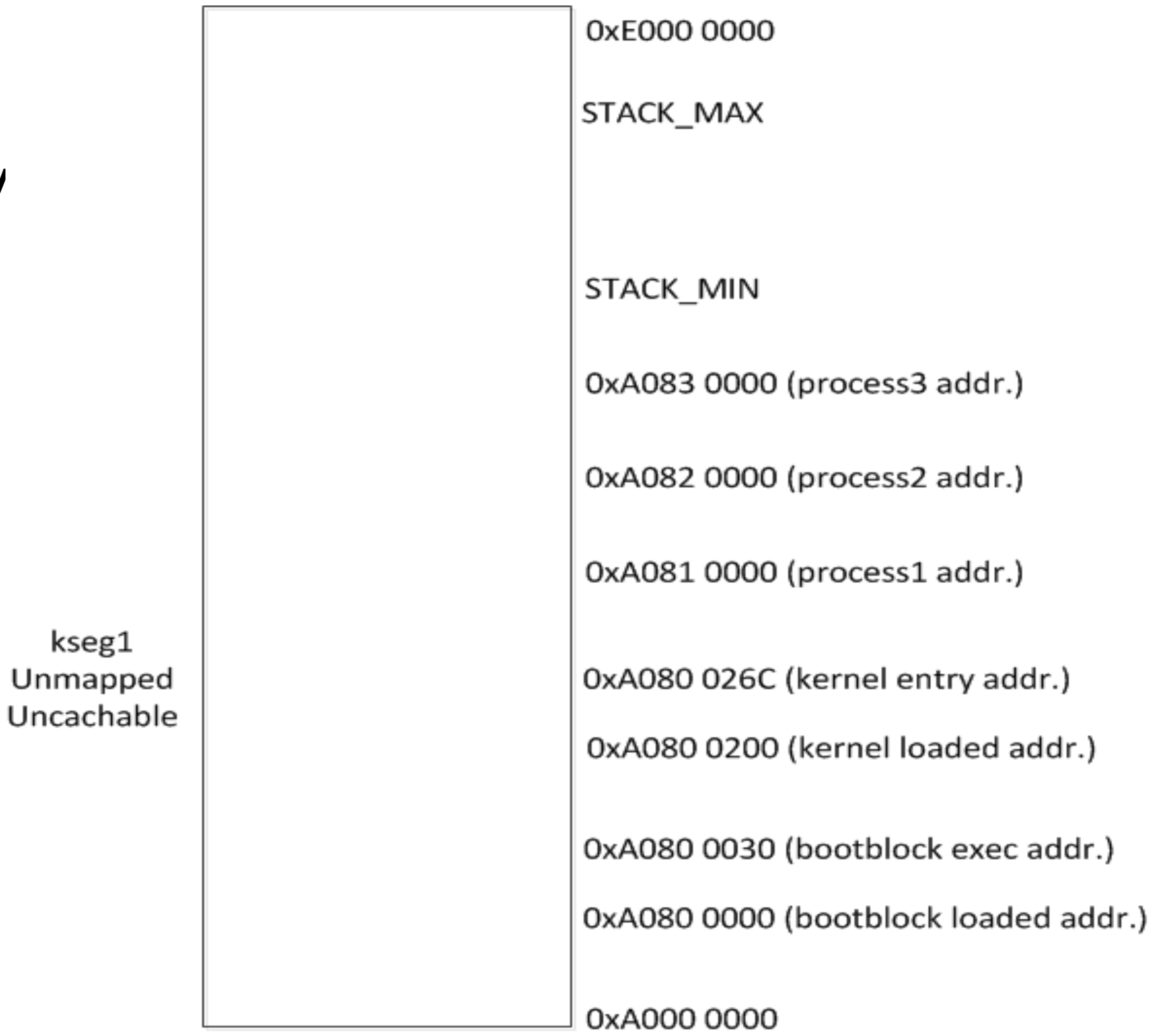
- Start a task(process/thread)
  - Task type
    - User Process
      - Use faked SYSCALL to call kernel functions
      - But, in this project compiled with the kernel and share the same address with the kernel
    - Kernel thread
  - Task entry point
    - Function addresses
  - Please refer to *task_info* structure in start-code

Project 2 – A Simple Kernel

- Start a task (process/thread)
  - – Each task is associated with a PCB
  - – Initialize PCB
    - Which registers should be set?
    - Where is the task located?
    - How to setup stack? Stack size?
    - Where is the PCB located?

Project 2 – A Simple Kernel

- Start a task
  - Possible memory layout
  - Decide your own STACK_MIN and STACK_MAX


0xE000 0000
STACK_MAX
STACK_MIN
0xA083 0000 (process3 addr.)
0xA082 0000 (process2 addr.)
0xA081 0000 (process1 addr.)
kseg1
Unmapped
Uncachable
0xA080 026C (kernel entry addr.)
0xA080 0200 (kernel loaded addr.)
0xA080 0030 (bootblock exec addr.)
0xA080 0000 (bootblock loaded addr.)
0xA000 0000

Project 2 – A Simple Kernel

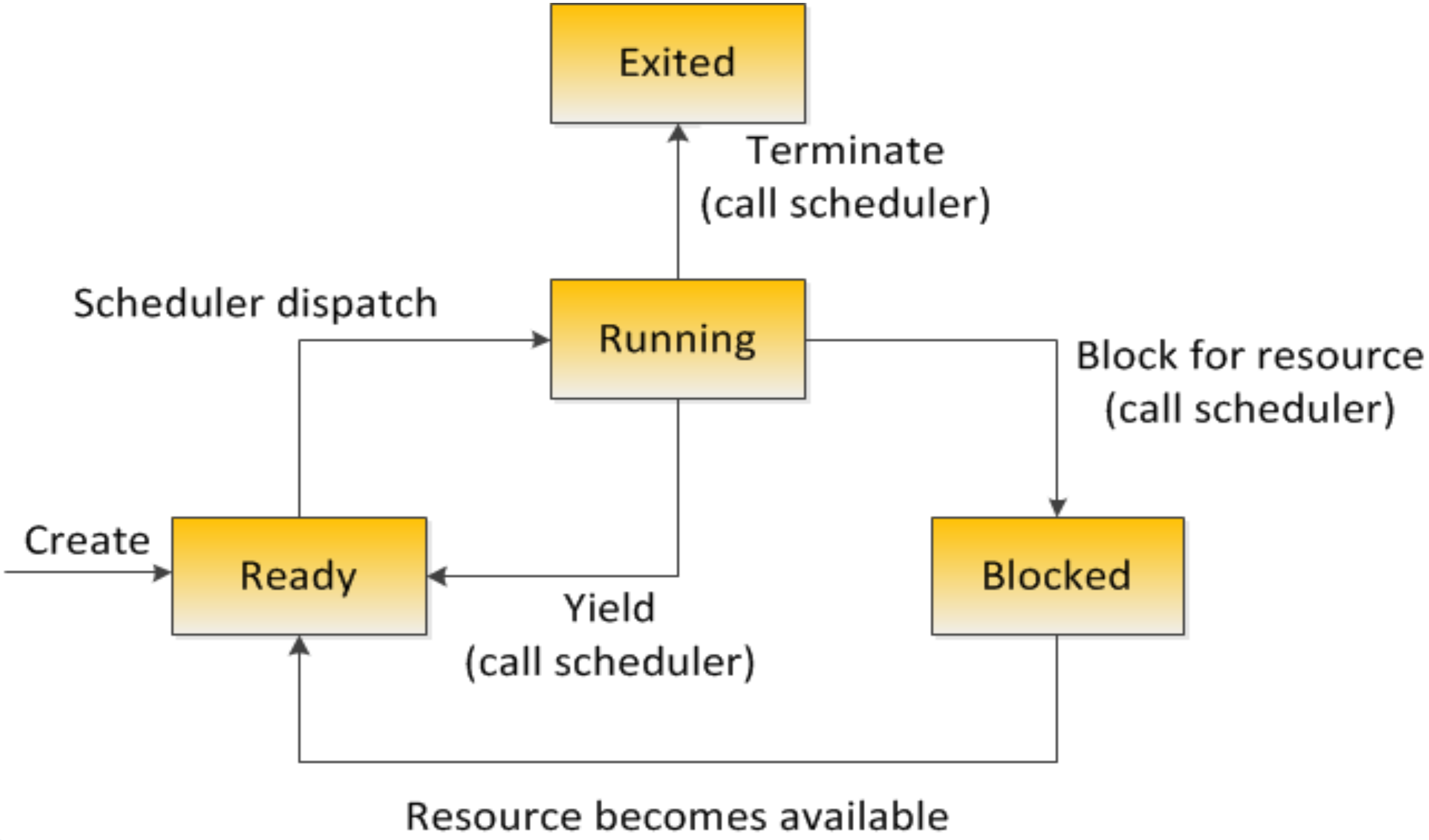
- Start a task
  - Scheduler: single task vs. multi-tasks
    - How to organize tasks being scheduled?
      - Use a queue
    - How to select the next task?
      - FIFO

Project 2 – A Simple Kernel

- Start a task
  - Scheduler: first task vs. the following ones
    - do_scheduler()
      - Locate the PCB of the first task
      - Restore PCB

Project 2 – A Simple Kernel

- Scheduler (non-preemptive kernel)



Project 2 – A Simple Kernel

- Yield
  - An action to force a processor to release control of the current running thread
  - Place the current running thread to the end of the running queue
  - In this project, we call do_scheduler() to execute yield

Project 2 – A Simple Kernel

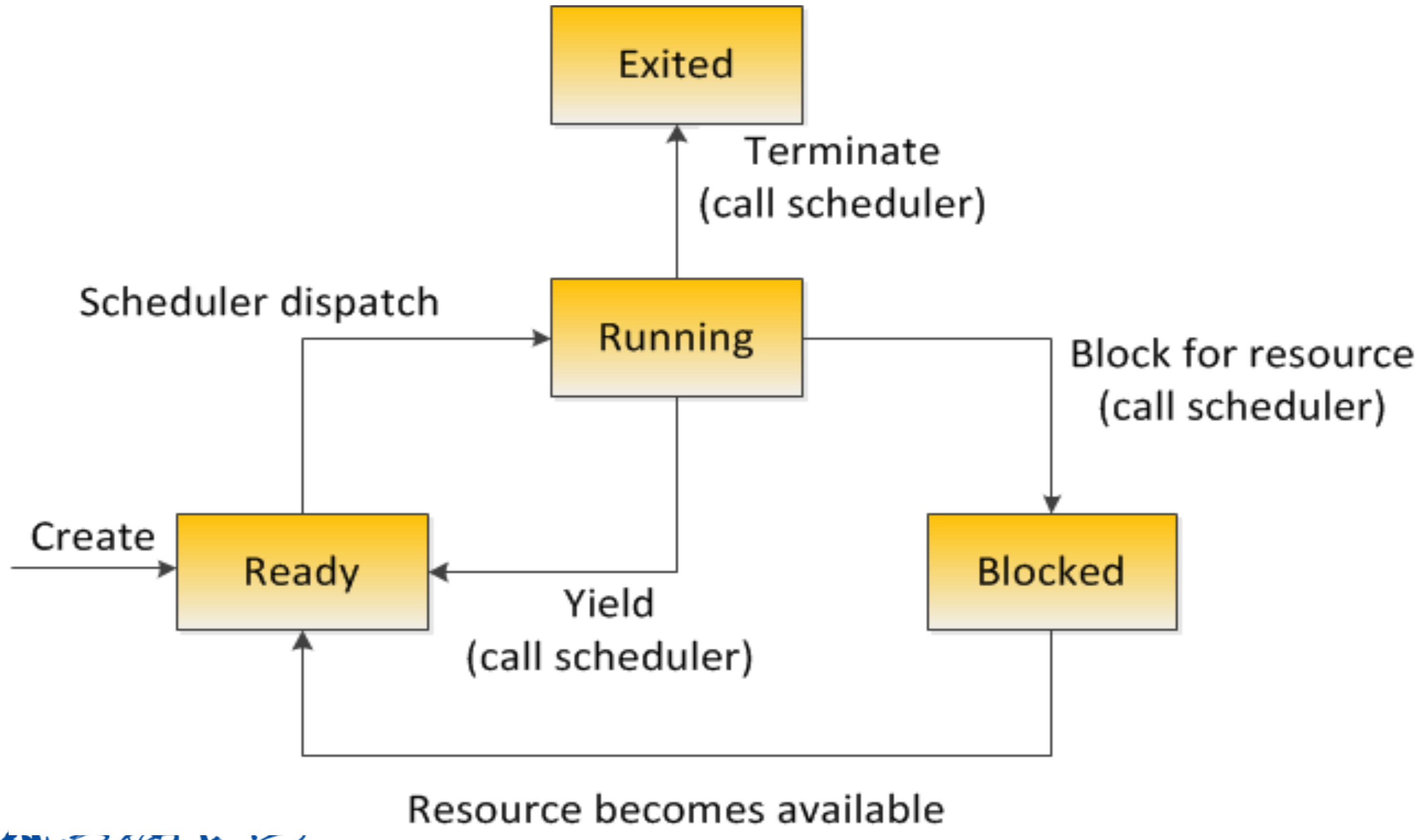
- Context switch
  - Save PCB
    - What kind of things to save
    - Registers → Memory
  - Restore PCB
    - Memory → Registers
    - do_scheduler()

Project 2 – A Simple Kernel

- Create machine image
  - Combine binary of kernel, tasks together into a machine image using createimage in your P1

Project 2 – A Simple Kernel

- Mutex lock



Project 2 – A Simple Kernel

- Mutex lock
  - What if no thread currently holds the lock?
    - Acquire the lock
  - What if the lock is currently held?
    - Wait
  - Implement lock-related functions
  - Manage tasks that do not acquire the lock
    - Same queue vs. different queues?

Project 2 A Simple Kernel

- Step by step
  - Task 0: PLEASE read the guiding book and start code CAREFULLY
  - Task 1: start a set of kernel threads and support context switch as a non-preemptive kernel
  - Task 2: implement mutex lock to support BLOCK state

Project 2 A Simple Kernel

- Requirement for design review (40 points)
  - Show the example code of your PCB
  - Provide the workflow or pseudo code for the task initialization
  - When is context switching in this project? Provide the workflow or pseudo code of the context switching?
  - When a task is blocked, how does the kernel handle the blocked task?

Project 2 – A Simple Kernel

- Requirement for developing (60 points)
  - Start tasks and set PCBs: 10
  - Execute context switch without errors: 30
  - Implement the mutex lock: 20

Project 2 – A Simple Kernel

- P2 schedule
  - 23rd Sep.
    - P2 part I design review:
    - P2 part II assignment
  - 30th Sep.
    - P2 part I due
    - P2 part II design review
  - 9th Oct. (optional)
    - P2 part II design review
  - 14th Oct.
    - P2 part II due

Project 2 – A Simple Kernel

- Final reminder
  - 重要的事情说三遍
  - Start early
  - Start early
  - Start early