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 2 Non-Preemptive Kernel

2016.09.21

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

Schedule

- Project 1 due
- Project 2 assignment

Project 1 Due

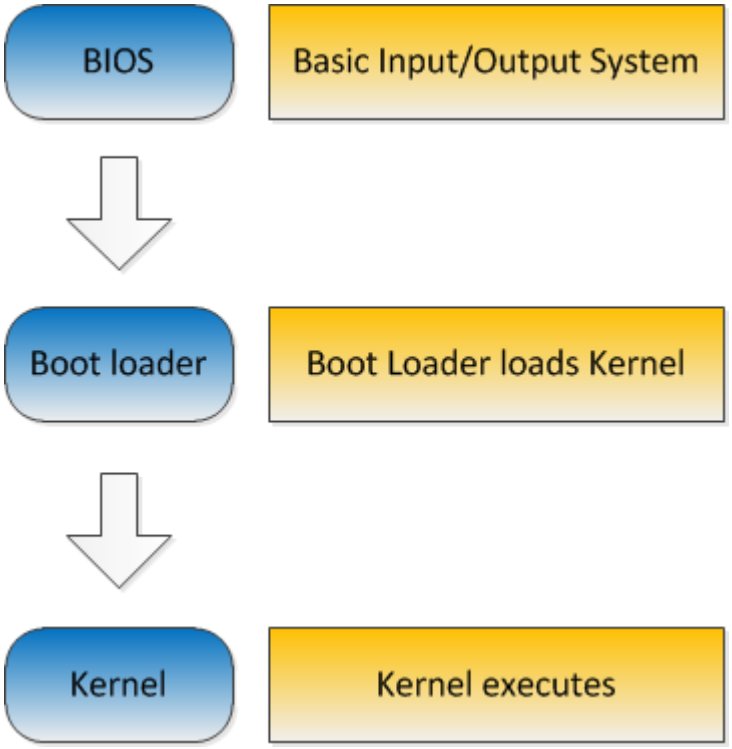
- P1 due
  - We test Tasks 3 and 4 for P1 due
  - Please compile your code, running the code on your board, and show the results to TA
  - Answer any questions we may ask

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
    - Design document covering the questions in the design document template
  - Do not forget to submit before 23:59 tonight

Project 2 – Non-Preemptive Kernel

- Booting procedure



Project 2 – Non-Preemptive Kernel

- Requirements
  - Write a non-preemptive kernel
    - Start a set of processes/threads
    - Perform context switches between processes and threads
    - Measure the cost of context switch
    - Implement basic mutual exclusion
    - Package the binary of kernel and processes/threads into machine image

Project 2 – Non-Preemptive Kernel

- Assumption about multi-tasks
  - A set of tasks
    - Program codes in processX.c and thX.c
    - Test defined in tasks.c
    - Fixed number of tasks for each test
    - Allocate per-task state statically in kernel.c
  - Non-preemptive tasks
    - Run until the task voluntarily yield or exit

Project 2 – Non-Preemptive Kernel

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

Project 2 – Non-Preemptive Kernel

- Process Control Block (PCB, TCB)
  - Process identification data
    - process_type
  - Process state data
    - Status of a process when it is suspended
    - Contents of registers, stack pointers etc.
  - Process control data
    - Process scheduling state
      - process_state

Project 2 – Non-Preemptive Kernel

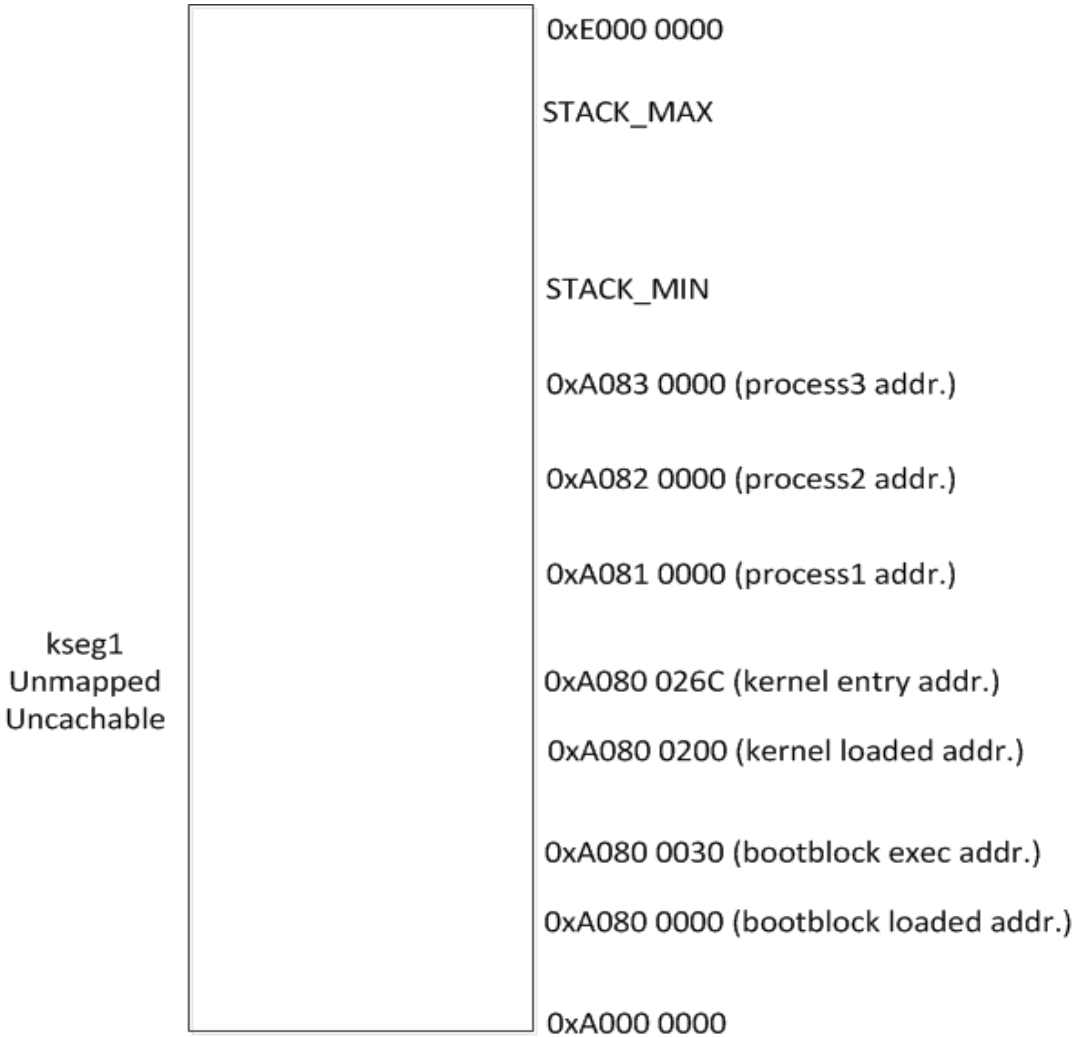
- Start a task – How to describe a task?
  - Task type
    - Process
      - Compiled separately from the kernel
      - Share the same address with the kernel
      - Only setup one stack
      - Still use faked SYSCALL to call kernel functions
    - Kernel thread
      - Share the same address with the kernel
  - Task entry point
    - Function addresses

Project 2 – Non-Preemptive Kernel

- Start a task
  - How to associate each task 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 – Non-Preemptive Kernel

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



Project 2 – Non-Preemptive Kernel

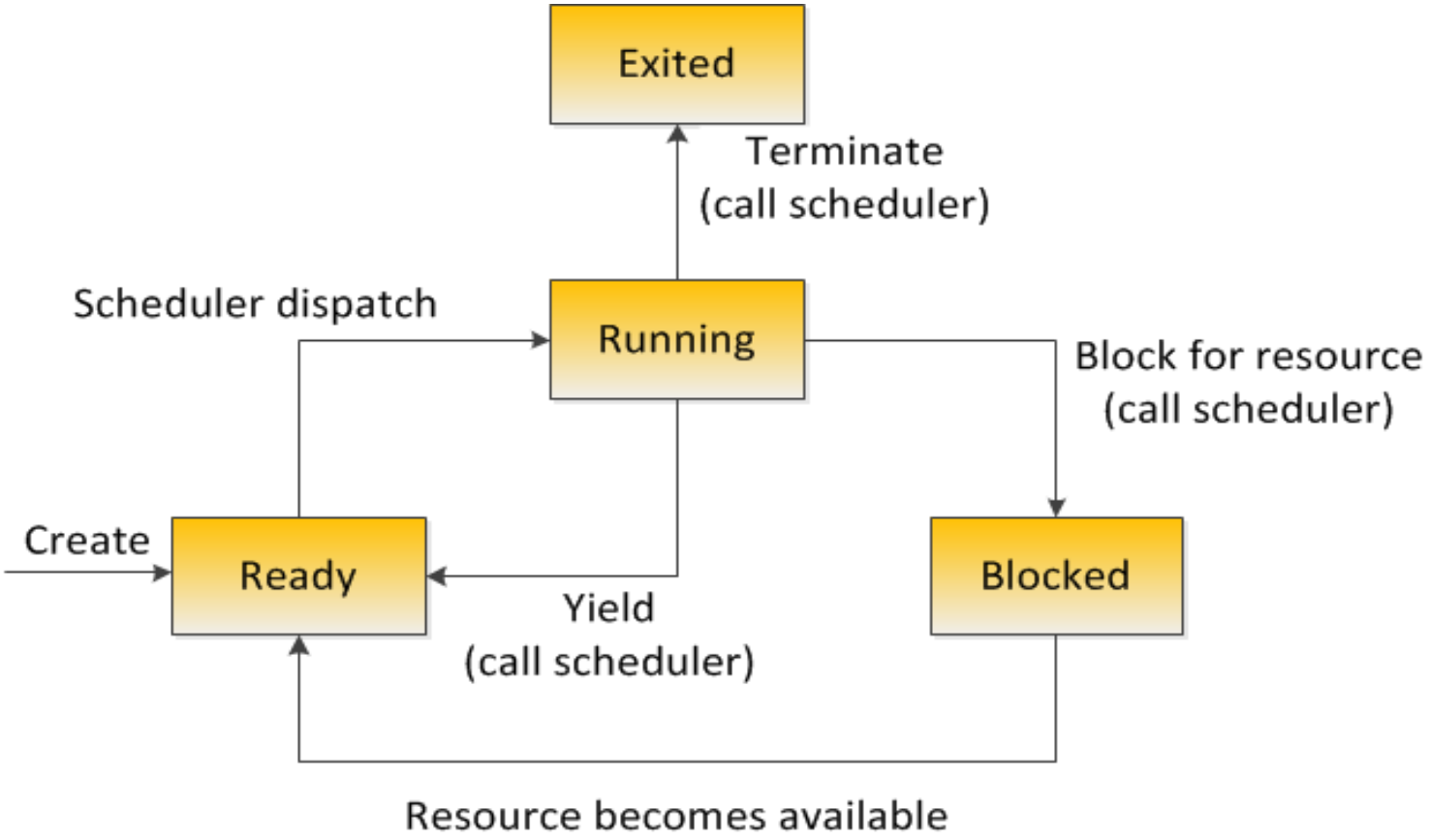
- Start a task
  - Scheduler: single task vs. multi-tasks
    - How to organize tasks being scheduled?
    - How to select the next task?
      - FIFO
      - Fairness

Project 2 – Non-Preemptive Kernel

- Start a task
  - Scheduler: first task vs. the following ones
    - schedule_entry()
      - Locate the PCB of the first task
      - Restore PCB
    - When the scheduler is invoked?
      - Non-preemptive kernel
      - Exit vs. Yeild

Project 2 – Non-Preemptive Kernel

- Scheduler

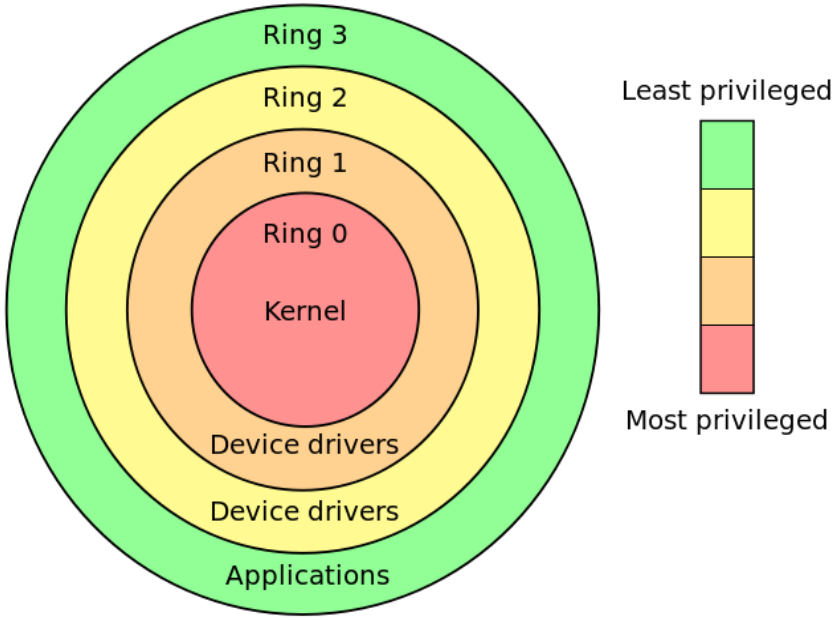


Project 2 – Non-Preemptive Kernel

- Yield
  - An action to force a processor to release control of the current running thread
  - Send the current running thread to the end of the running queue

Project 2 – Non-Preemptive Kernel

- Yield
  - How to call yield()?
  - Kernel thread vs. user process
    - Directly access the addresses of the function
    - System call
      - kernel_entry



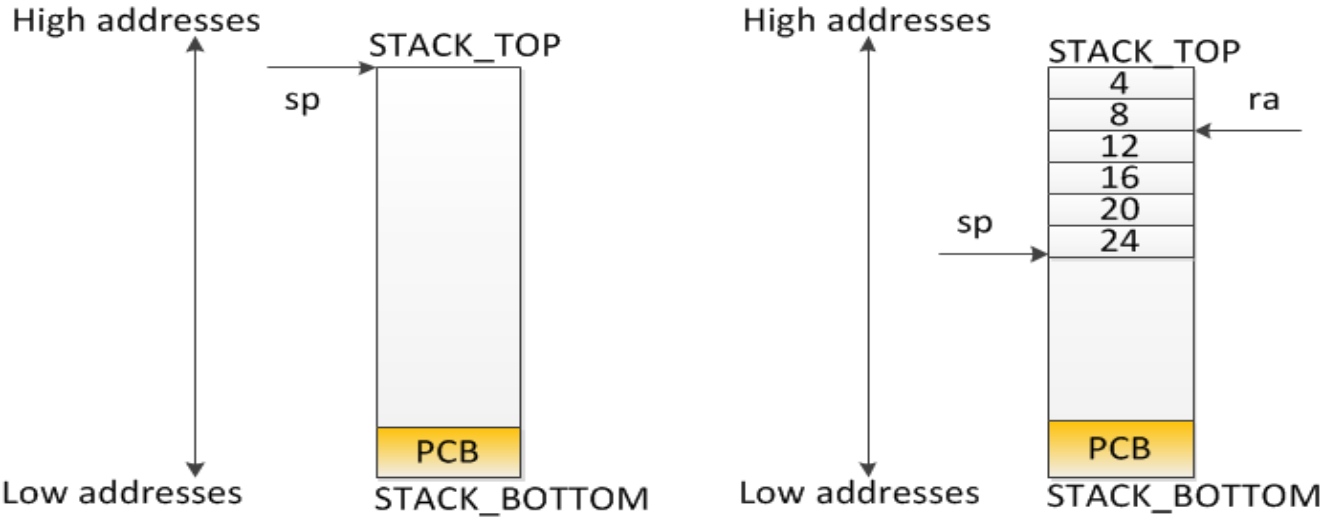
Project 2 – Non-Preemptive Kernel

- Context switch
  - Save PCB
    - What kind of things to save
    - Registers → Memory
  - Restore PCB
    - Memory → Registers
    - schedule_entry

Project 2 – Non-Preemptive Kernel

- Tips on context switch

```
398 a08007e4 <do_yield>:
399 a08007e4:   27bdffe8    addiu   sp,sp,-24
400 a08007e8:   afbf0010    sw  ra,16(sp)
401 a08007ec:   0c20015a    jal a0800568 <save_pcb>
402 a08007f0:   00000000    nop
```

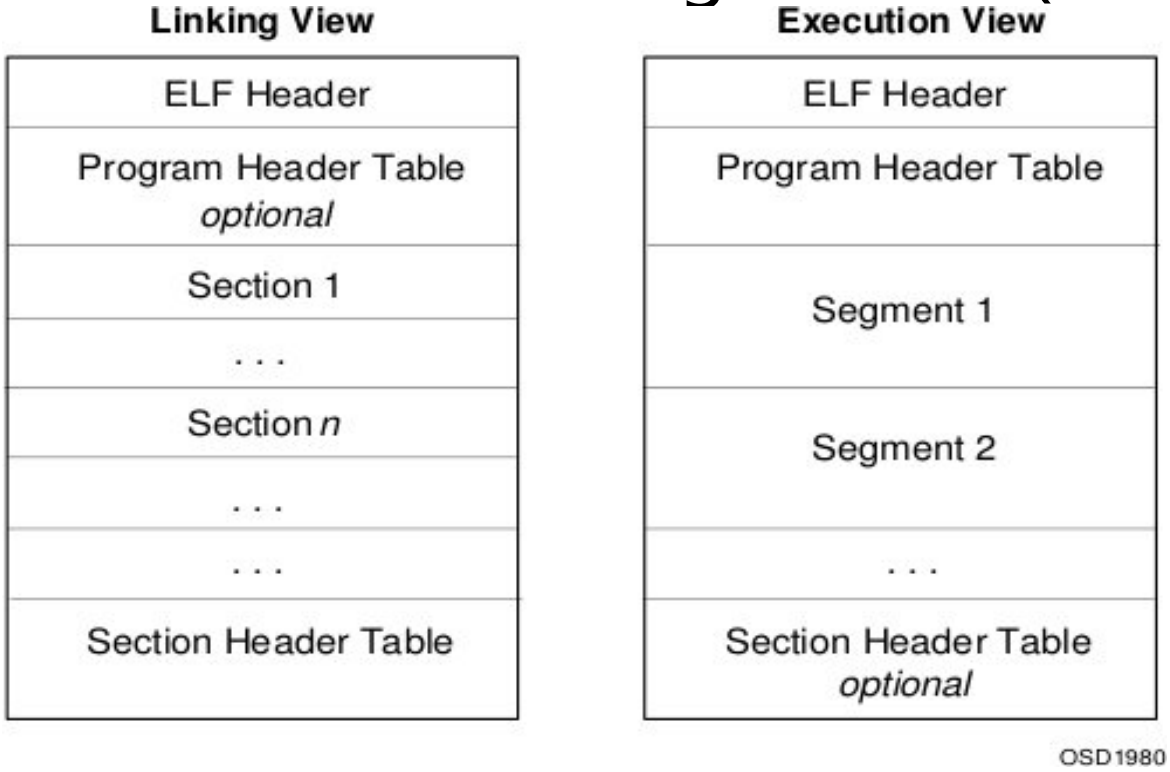


Project 2 – Non-Preemptive Kernel

- Create machine image
  - Combine binary of kernel, processX together into a machine image
  - Pay attention to the disk layout of the image between the kernel and the process segments

Project 2 – Non-Preemptive Kernel

- Recall: ELF object file format
  - Executable and Linking Format (ELF)



Project 2 – Non-Preemptive Kernel

- Recall: ELF object file format
  - Program header

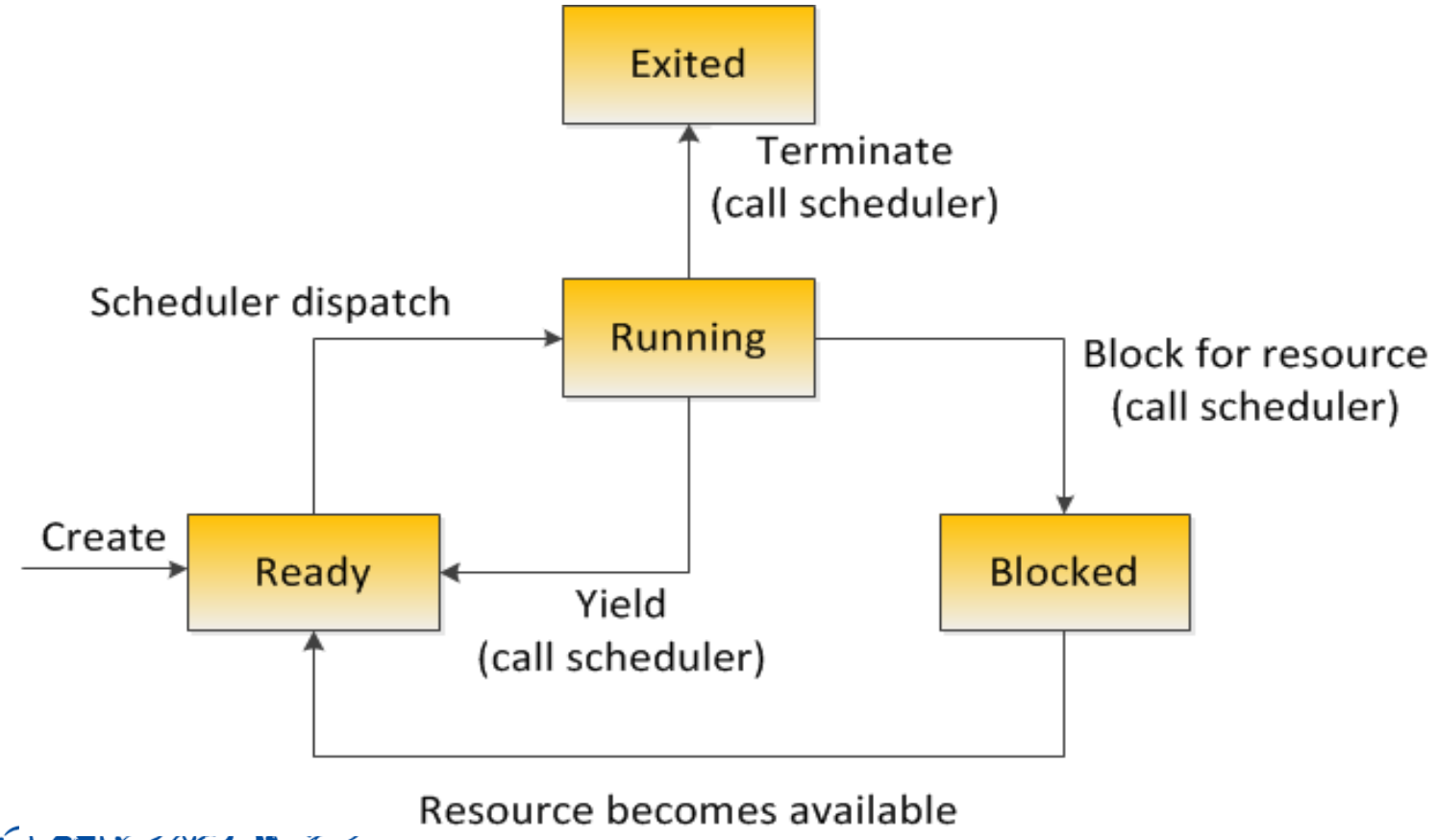
```
typedef struct
{
  Elf32_Word    p_type;       /* Segment type */
  Elf32_Off     p_offset;     /* Segment file offset */
  Elf32_Addr    p_vaddr;      /* Segment virtual address */
  Elf32_Addr    p_paddr;      /* Segment physical address */
  Elf32_Word    p_filesz;     /* Segment size in file */
  Elf32_Word    p_memsz;      /* Segment size in memory */
  Elf32_Word    p_flags;      /* Segment flags */
  Elf32_Word    p_align;      /* Segment alignment */
} Elf32_Phdr;
```

Project 2 – Non-Preemptive Kernel

- Measure the cost of context switch
  - get_timer()
    - Return the cycles that have been executed since boot
  - Measure the number of cycles required by do_yield and yield
  - Note that
    - the 32bit register used to provide cycle record will overflow sooner in P2
    - Try to understand how to measure the cost instead of focusing on the final numeric result

Project 2 – Non-Preemptive Kernel

- Mutual lock



Project 2 – Non-Preemptive Kernel

- Mutual lock
  - What if no thread currently holds the lock?
  - What if the lock is currently held?
  - Implement three functions
    - lock_init(l)
    - lock_acquire(l)
    - lock_release(l)
  - How to manage tasks that do not acquire the lock?

Project 2 – Non-Preemptive Kernel

- Step by step
  - Task 0: PLEASE read the start code carefully
  - Task 1: start a set of processes/threads and do context switch, package all binaries into bootable machine image
  - Task 2: measure the cost of context switch
  - Task 3: implement mutual lock

Project 2 – Non-Preemptive Kernel

- Requirement for design review (40 points)
  - What is PCB? What are included in PCB?
  - What need to be done for initializing tasks?
  - When is context switching in this project? What need to be done for context switching?
  - How do you measure the context switch?
  - How do mutual locks work?
  - How do you manage blocked tasks?
  - How do you create machine image?

Project 2 – Non-Preemptive Kernel

- Requirement for developing (60 points)
  - Start tasks and set PCBs: 10
  - Execute context switch without errors: 30
  - Measure the time for do_yield and yield between threads and proecesses: 5
  - Implement the mutual lock between kernel threads: 10
  - Package all binaries into a bootable machine image 5

Project 2 – Non-Preemptive Kernel

- P2 schedule
  - P2 design review: 11th Oct.
  - P2 due: 18th Oct.
- Final reminder
  - 重要的事情说三遍
  - Start early
  - Start early
  - Start early