CS330: Operating Systems

Limited direct execution

Recap: virtual view of resources

- Process
    - Each running process thinks that it owns the CPU
- Address space
    - Each process feels like it has a huge address space
- File system tree
    - The user feels like operating on the files directly
- What are the OS responsibilities in providing the above virtual notions?

Recap: virtual view of resources

- Process
    - Each running process thinks that it owns the CPU
- Address space
    - Each process feels like it has a huge address space
- File system tree
    - The user feels like operating on the files directly
- What are the OS responsibilities in providing the above virtual notions?
    - The OS performs multiplexing of physical resources efficiently
    - Maintains mapping of virtual view to physical resource

Virtualization: Efficiency/performance

- Resource virtualization should not add any overheads
- Efficient when programs use the resources directly, no OS mediation
    - Example: when a process is scheduled on CPU, it should execute without OS intervention
- What is the catch?

Virtualization: Efficiency/performance

- Resource virtualization should not add any overheads
- Efficient when programs use the resource directly, infrequent OS mediation
  - Example: when a process is scheduled on CPU, it should execute without OS intervention
- What is the catch?
  - Loss of control e.g., process running an infinite loop on a CPU
  - Isolation issues e.g., process accessing/changing OS data structures

Virtualization: Efficiency/performance

- Resource virtualization should not add any overheads
- Efficient when programs use the resource directly, infrequent OS mediation
    - Example: when a process is scheduled on CPU, it should execute without OS intervention
- What is the catch?
    - Loss of control e.g., process running an infinite loop on a CPU
    - Isolation issues e.g., process accessing/changing OS data structures

Conclusion: Some limits to direct access must be enforced.

Limited direct access

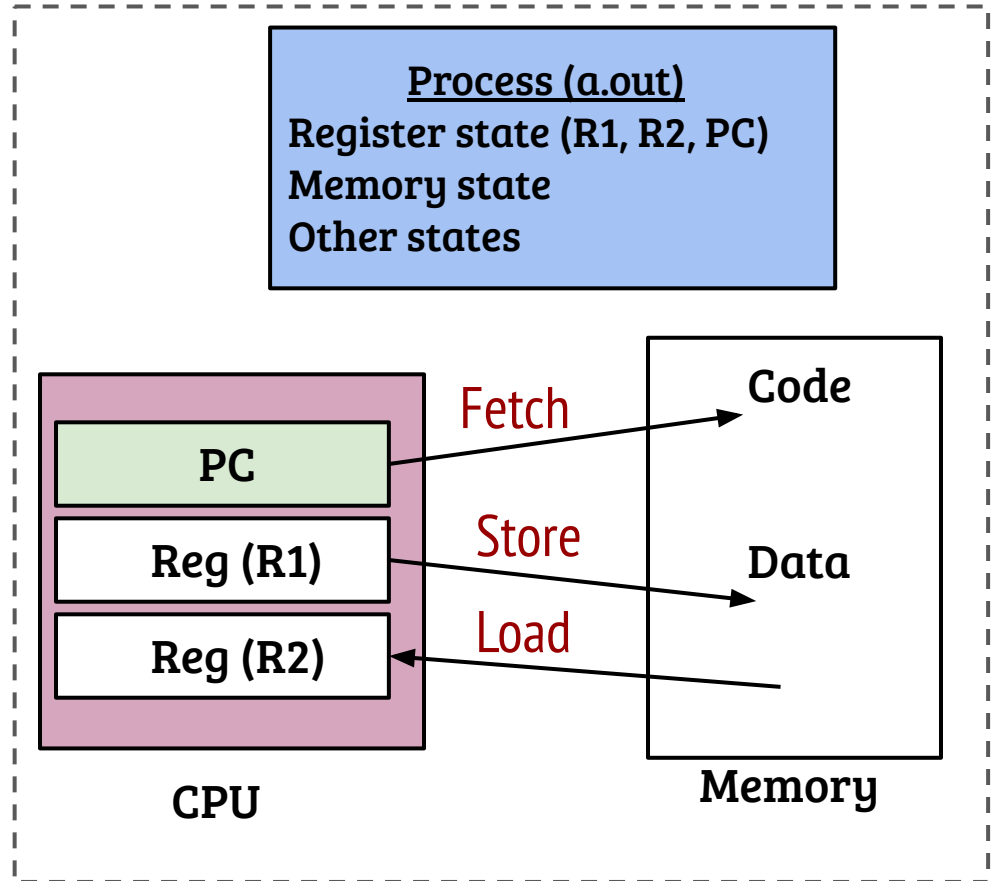- Can the OS enforce limits to an executing process by itself?

A process in execution

I want to take control of the CPU from this process which is executing an infinite loop, but how?

OS

I want to restrict this process accessing memory of other processes, but how? Monitoring each memory access is not efficient!

Process (a.out)
Register state (R1, R2, PC)
Memory state
Other states

CPU

PC

Reg (R1)

Reg (R2)

Fetch

Store

Load

Code

Data

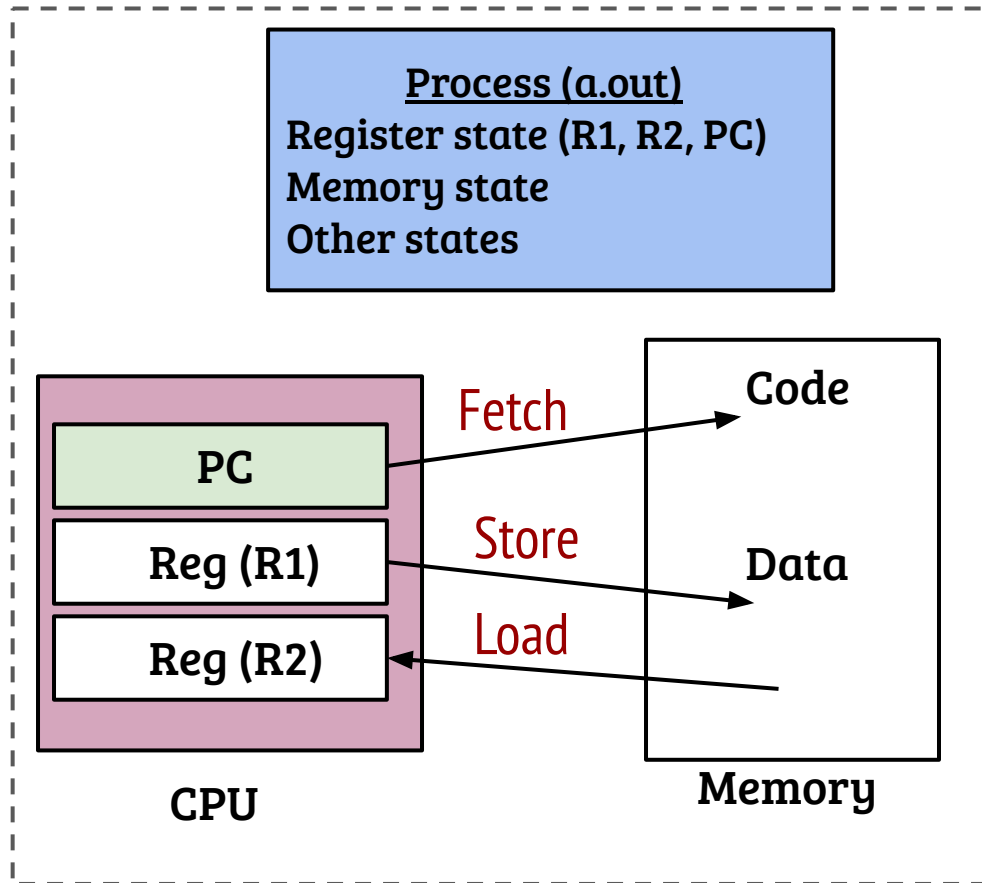**Memory**

A process in execution

I want to take control of the CPU from this process which is executing an infinite loop, but how?

Help me!

OS

I want to restrict this process accessing memory of other processes, but how? Monitoring each memory access is not efficient!

Process (a.out)
Register state (R1, R2, PC)
Memory state
Other states

PC

Reg (R1)

Reg (R2)

CPU

Fetch

Store

Load

Code

Data

Memory

Limited direct access

- Can the OS enforce limits to an executing process by itself?
- No, the OS can not enforce limits by itself and still achieve efficiency
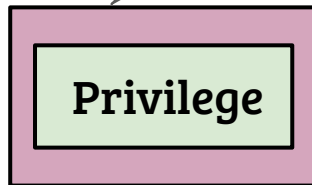- OS requires support from hardware!

Limited direct access

- Can the OS enforce limits to an executing process?
- No, the OS can not enforce limits by itself and still achieve efficiency
- OS requires support from hardware!
- What kind of support is needed from the hardware?
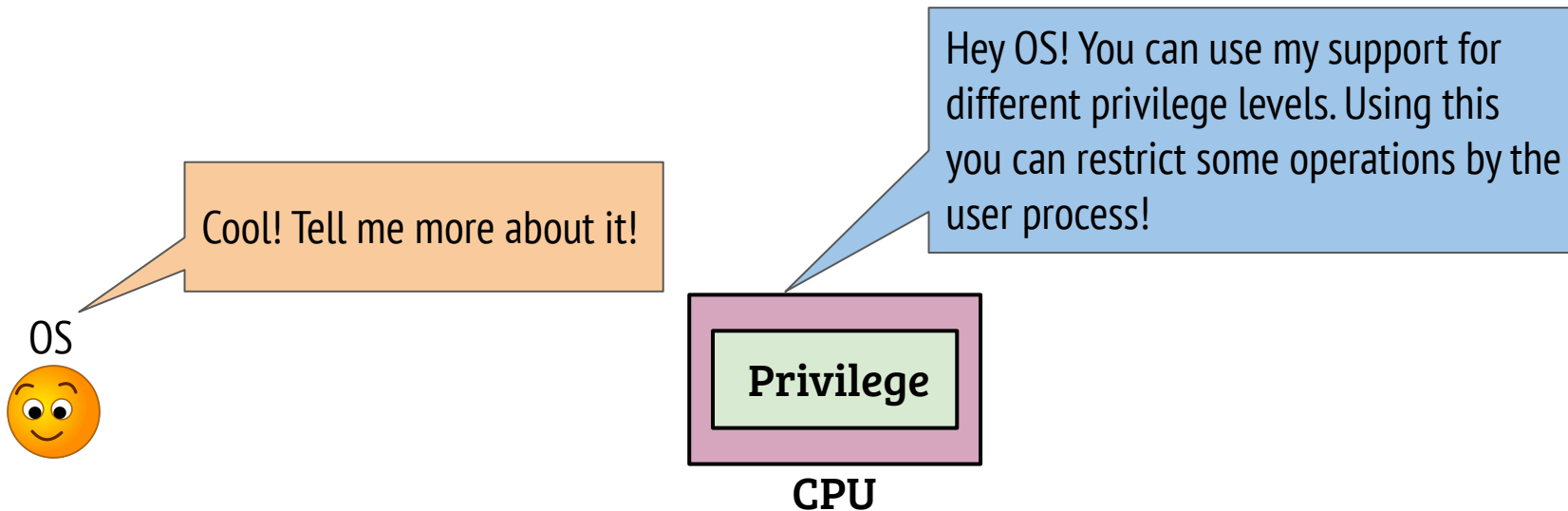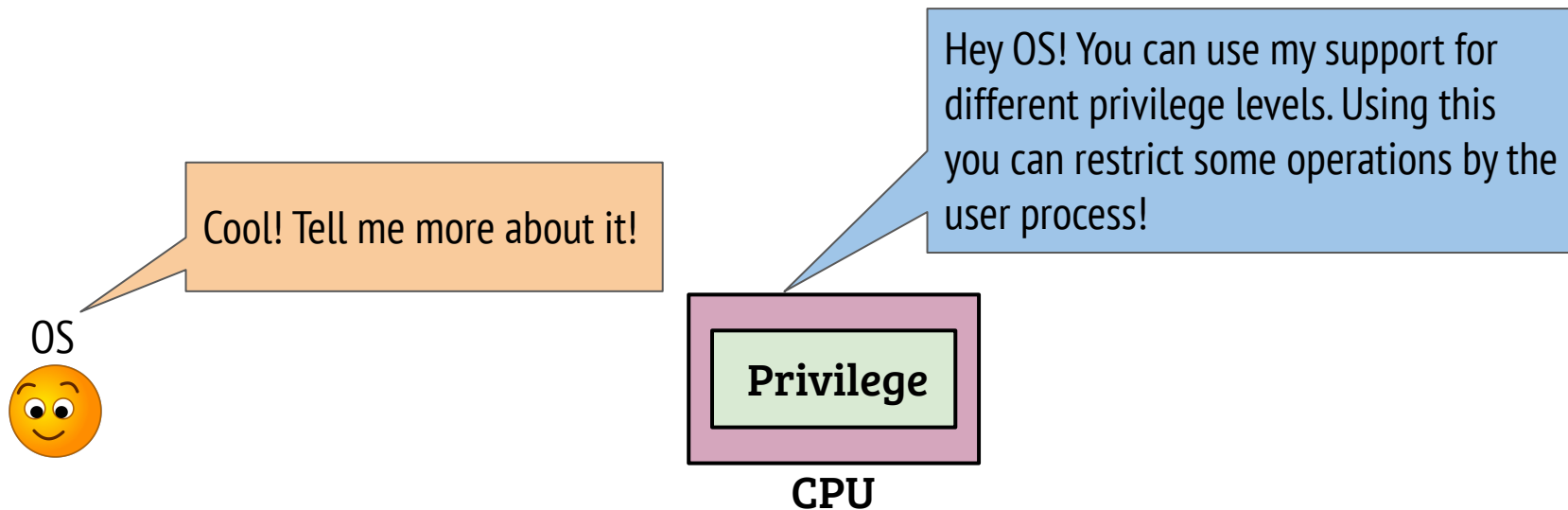
Hardware support: Privilege levels

Hey OS! You can use my support for different privilege levels. Using this you can restrict some operations by the user process!

OS

Help me!

Privilege

CPU

Hardware support: Privilege levels

Hardware support: Privilege levels

Cool! Tell me more about it!

OS

Hey OS! You can use my support for different privilege levels. Using this you can restrict some operations by the user process!

Privilege

CPU

- CPU can execute in two modes: *user-mode* and *kernel-mode*
- Some operations are allowed only from kernel-mode (privileged OPs)
    - If executed from user mode, hardware will notify the OS by raising a fault/trap

Hardware support: Privilege levels

Good. I will run the user process in user-mode. Hang on! The user process requires to invoke my services. How can the user change the mode?

OS

Hey OS! You can use my support for different privilege levels. Using this you can restrict some operations by the user process!

Privilege

CPU

Hardware support: Privilege levels

Good. I will run the user process in user-mode. Hang on! The user process requires to invoke my services. How can the user change the mode?

OS

Hey OS! You can use my support for different privilege levels. Using this you can restrict some operations by the user process!
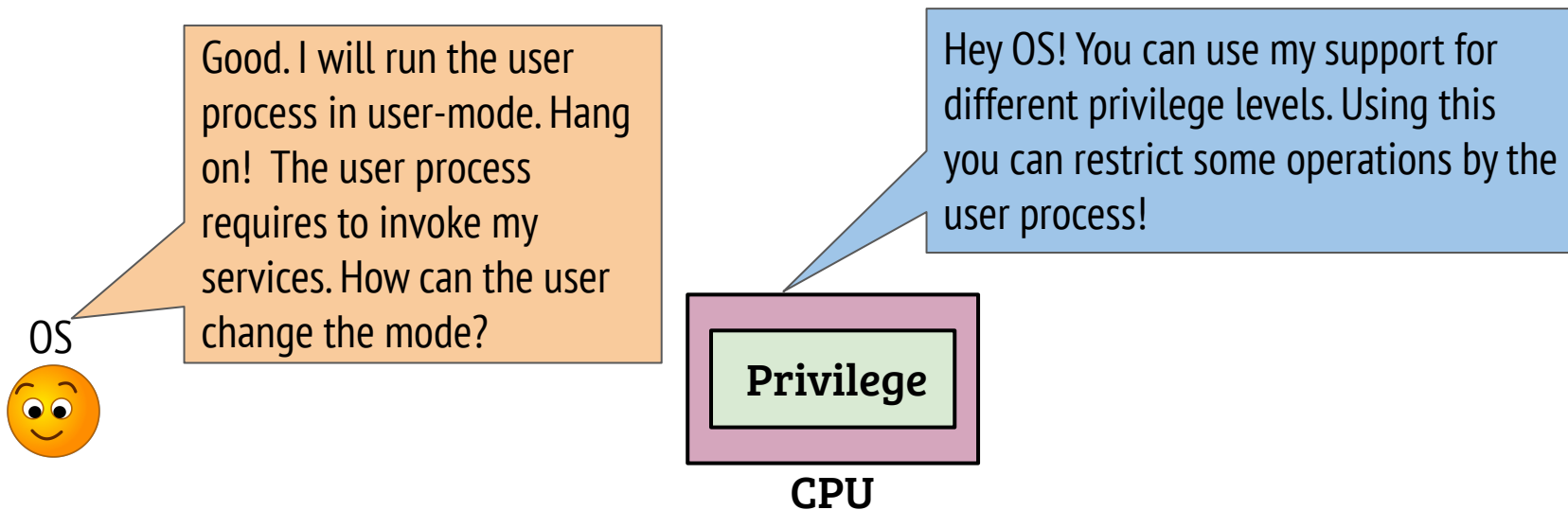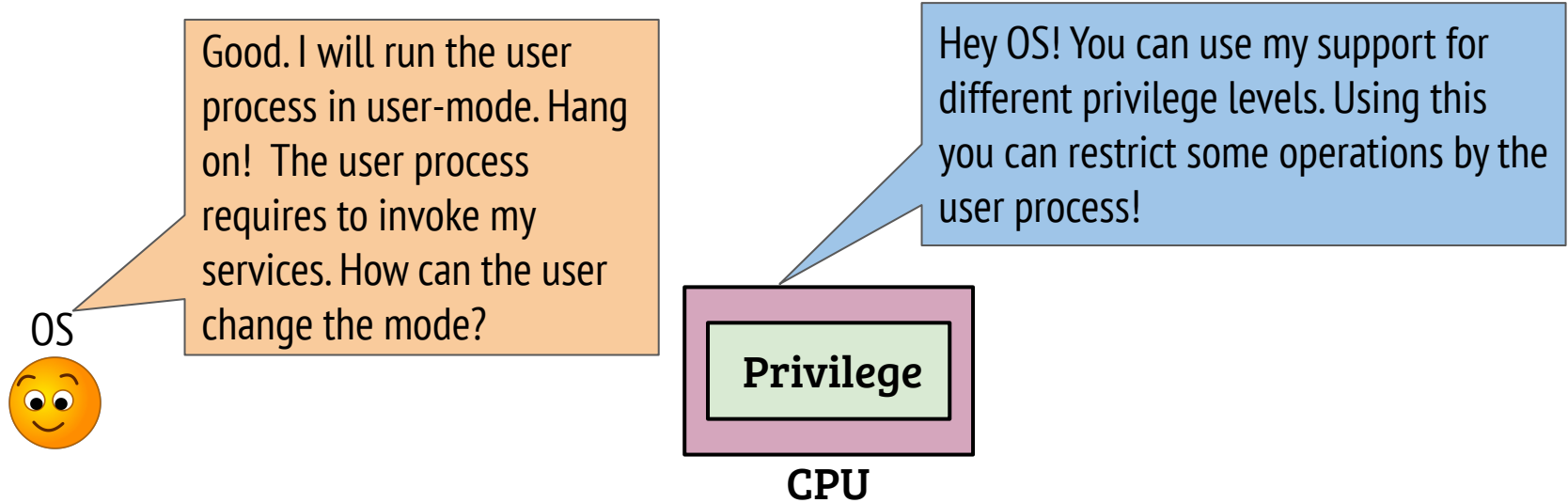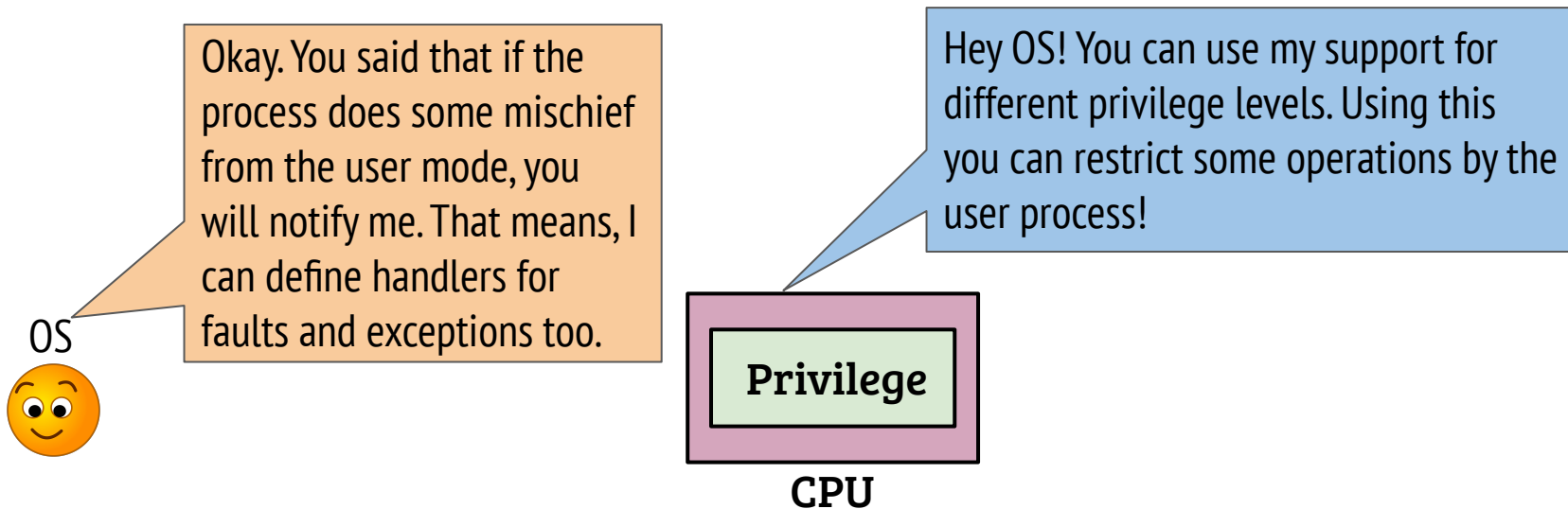
Privilege

CPU

- From user-mode, privilege level of CPU can not be changed directly
- The hardware provides entry instructions from the user-mode which causes a mode switch
- The OS can define the handler for different entry gates

Hardware support: Privilege levels

Okay. You said that if the process does some mischief from the user mode, you will notify me. That means, I can define handlers for faults and exceptions too.

OS

Hey OS! You can use my support for different privilege levels. Using this you can restrict some operations by the user process!

Privilege

CPU

Hardware support: Privilege levels

Okay. You said that if the process does some mischief from the user mode, you will notify me. That means, I can define handlers for faults and exceptions too.

OS

Hey OS! You can use my support for different privilege levels. Using this you can restrict some operations by the user process!
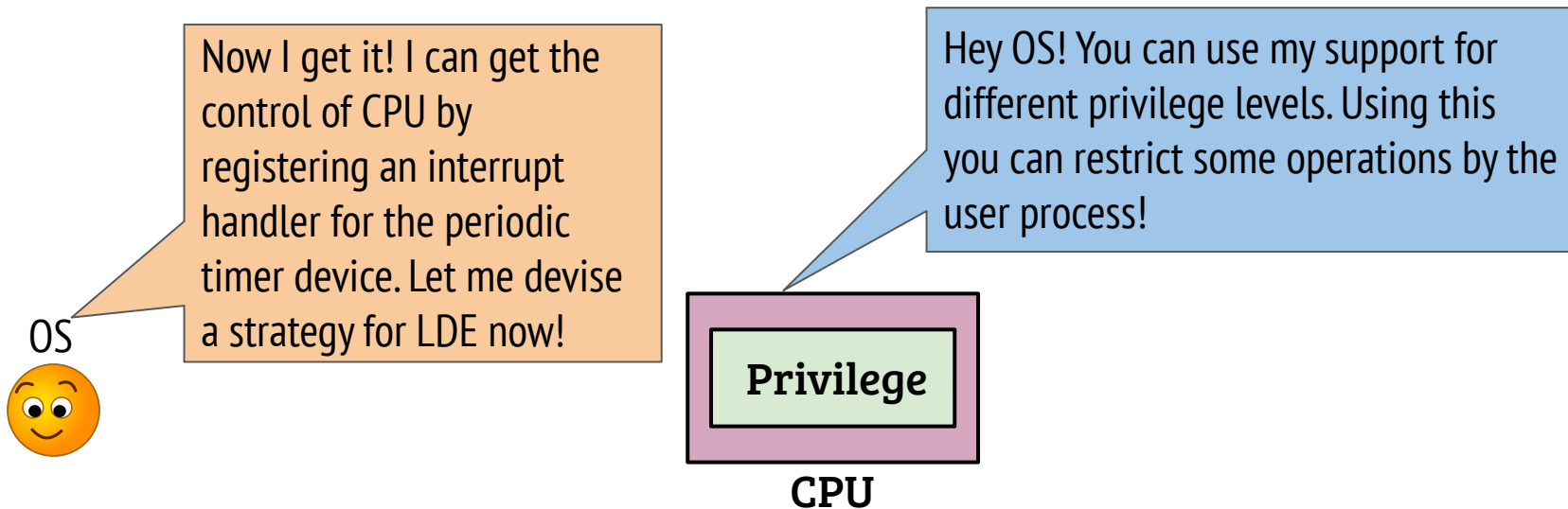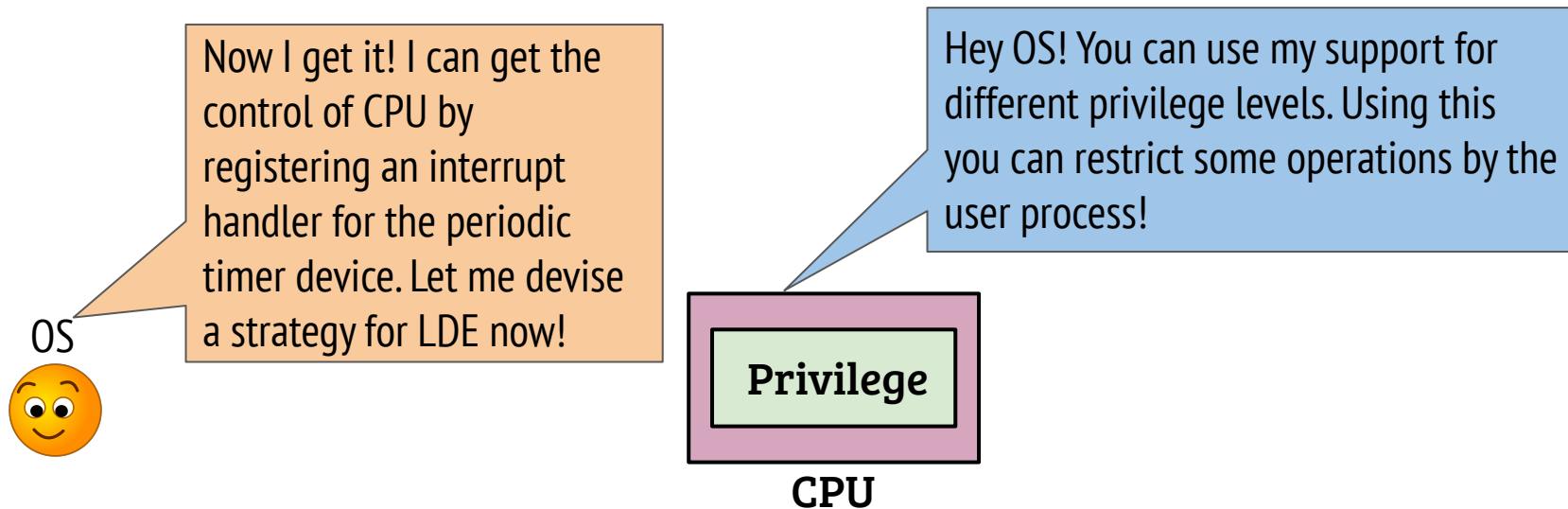
Privilege

CPU

- The OS can register the handlers for faults and exceptions
- The OS can also register handlers for device interrupts
- *Registration of handlers is privileged!*

Hardware support: Privilege levels

Now I get it! I can get the control of CPU by registering an interrupt handler for the periodic timer device. Let me devise a strategy for LDE now!

OS

Hey OS! You can use my support for different privilege levels. Using this you can restrict some operations by the user process!

Privilege

CPU

Hardware support: Privilege levels

OS: Now I get it! I can get the control of CPU by registering an interrupt handler for the periodic timer device. Let me devise a strategy for LDE now!

CPU: Hey OS! You can use my support for different privilege levels. Using this you can restrict some operations by the user process!

Privilege

CPU

- After the boot, the OS needs to configure the handlers for system calls, exceptions/faults and interrupts
- The handler code is invoked by the hardware when user invokes a system call or an exception or an external interrupt

Limited direct access

- Can the OS enforce limits to an executing process?
- No, the OS can not enforce limits by itself and still achieve efficiency
- OS requires support from hardware!
- What kind of support is needed from the hardware?
- CPU privilege levels: user-mode vs. kernel-mode
- Switching between modes, entry points and handlers