

# Embedded Software

---

Introduction to operating systems

# Agenda

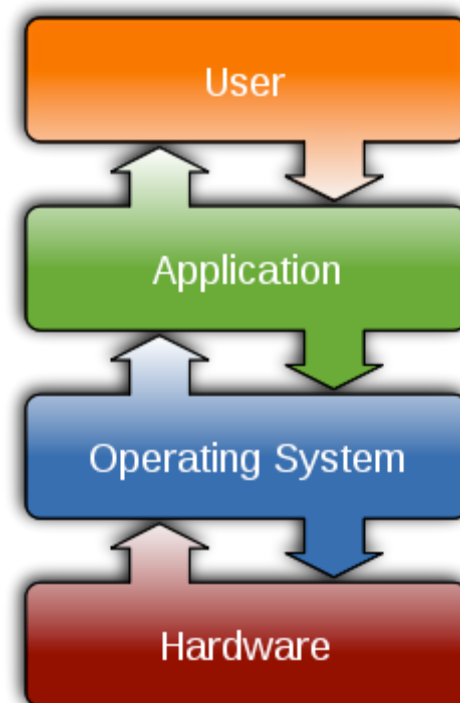
---

- What is an operating system?
- OS structure
  - ▶ Process management
  - ▶ Memory and storage mgmt.
  - ▶ I/O subsystem
- Operating systems - Real-Time OS's

# What is an operating system?

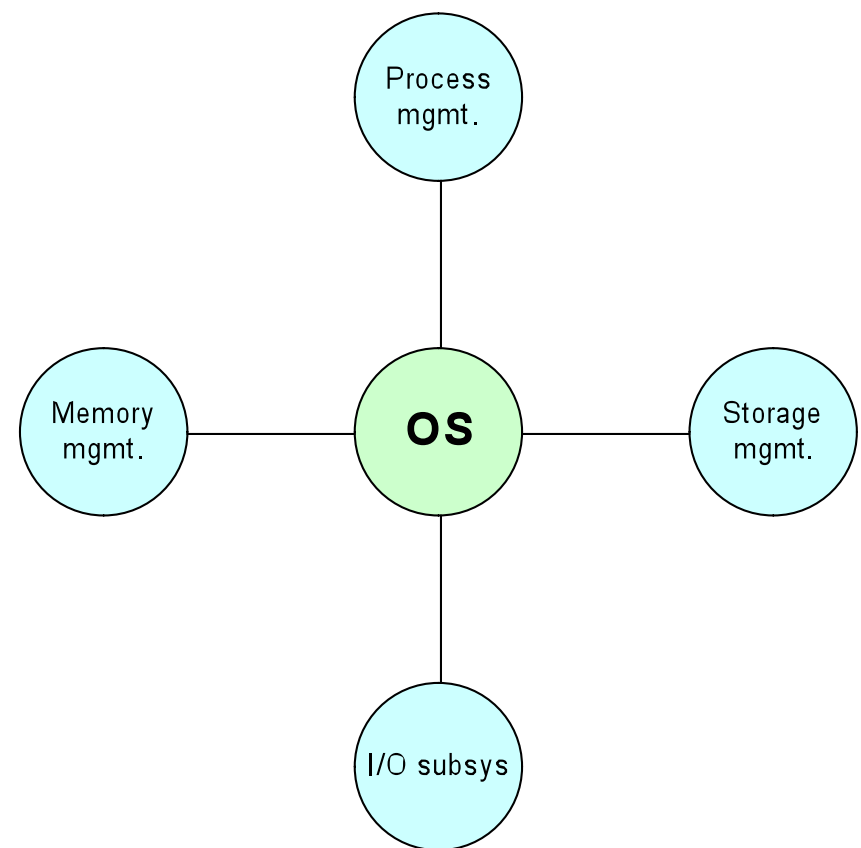
---

- What is an operating system?
  - ▶ Wikipedia: *“An operating system (OS) is software (...) that manages computer hardware resources and provides common services for efficient execution of various application software.”*



# OS structure

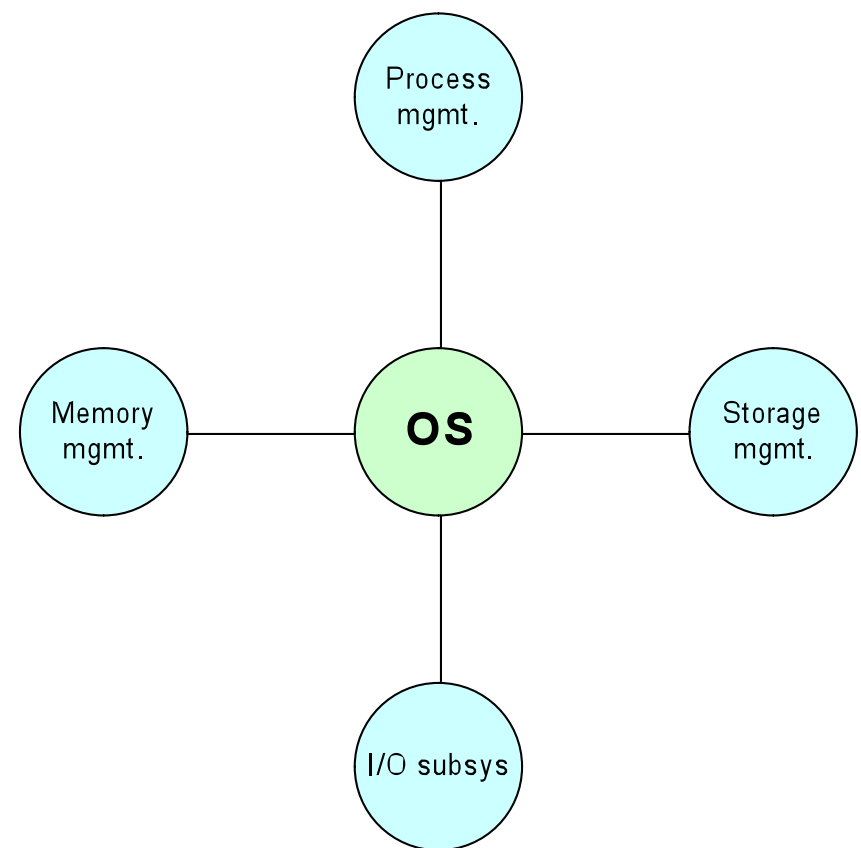
---



# OS structure

---

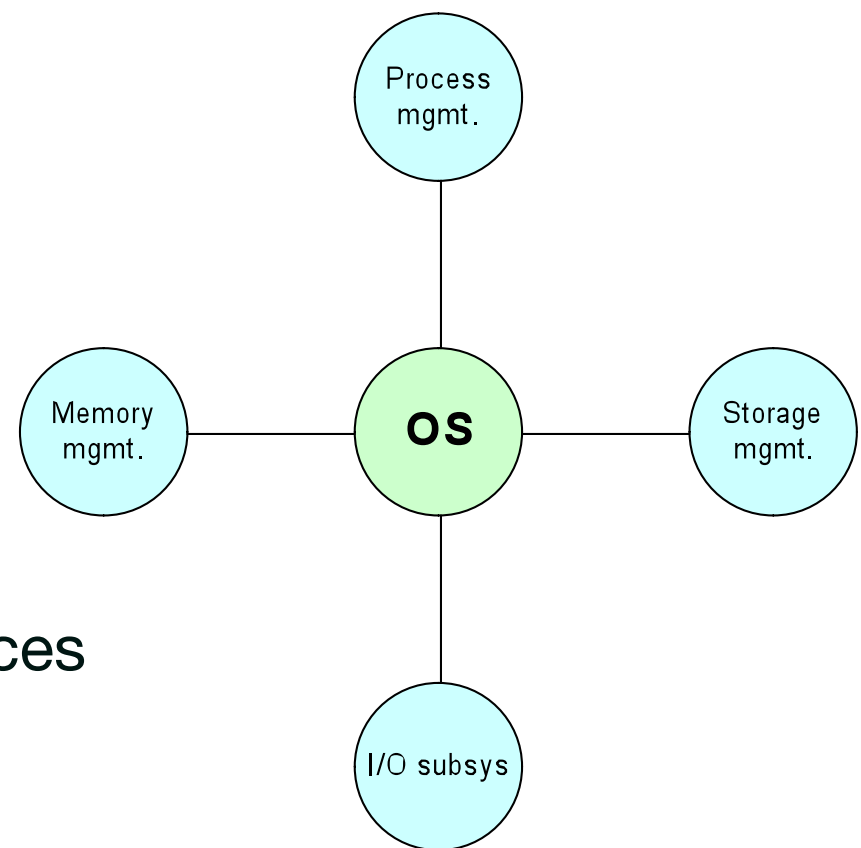
- Many computer types – many OS designs
  - ▶ Mainframe OSs are optimized for HW utilization
  - ▶ Desktop OSs are optimized for generality
  - ▶ Embedded OSs are optimized for efficiency, size, safety, speed, low power
  - ▶ ...



# OS structure

---

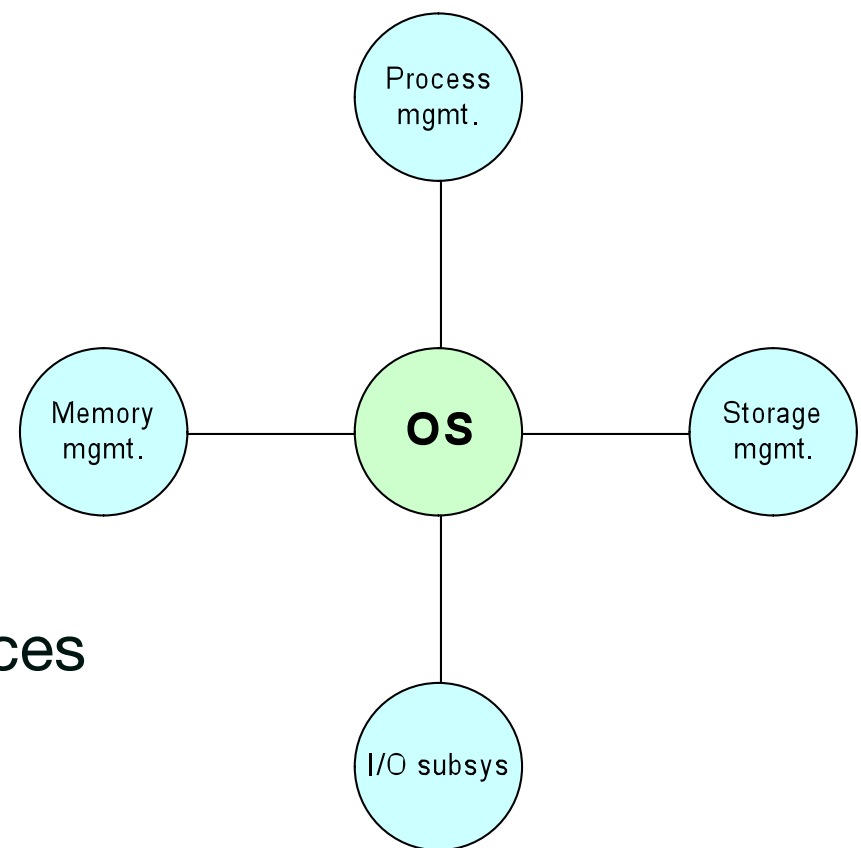
- Many computer types – many OS designs
  - ▶ Mainframe OSs are optimized for HW utilization
  - ▶ Desktop OSs are optimized for generality
  - ▶ Embedded OSs are optimized for efficiency, size, safety, speed, low power
  - ▶ ...
- Some commonalities, though:
  - ▶ Process management – handles multiprogramming and keep the CPU busy
  - ▶ Memory management – (de)allocation and process swapping
  - ▶ Storage management – persistent storage and cache
  - ▶ I/O subsystem management – manage I/O devices



# OS structure

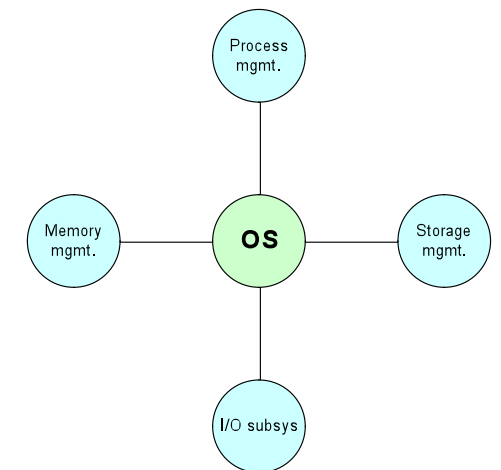
---

- Many computer types – many OS designs
  - ▶ Mainframe OSs are optimized for HW utilization
  - ▶ Desktop OSs are optimized for generality
  - ▶ Embedded OSs are optimized for efficiency, size, safety, speed, low power
  - ▶ ...
- Some commonalities, though:
  - ▶ Process management – handles multiprogramming and keep the CPU busy
  - ▶ Memory management – (de)allocation and process swapping
  - ▶ Storage management – persistent storage and cache
  - ▶ I/O subsystem management – manage I/O devices
- Let's take a look at process management



# OS structure - Process management

---

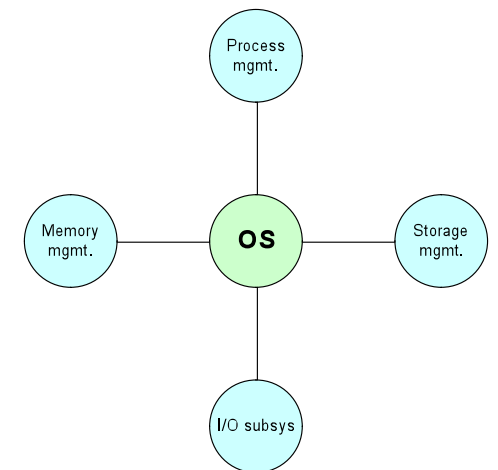




# OS structure - Process management

---

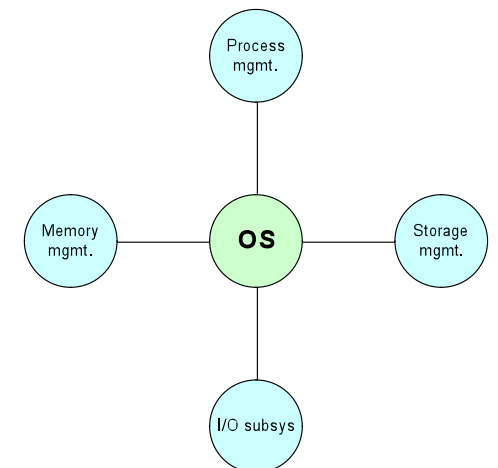
- ***What is a process? Is it a program?***
  - ▶ No - a process is a program in execution



# OS structure - Process management

---

- ***What is a process? Is it a program?***
  - ▶ No - a process is a program in execution



# OS structure - Process management

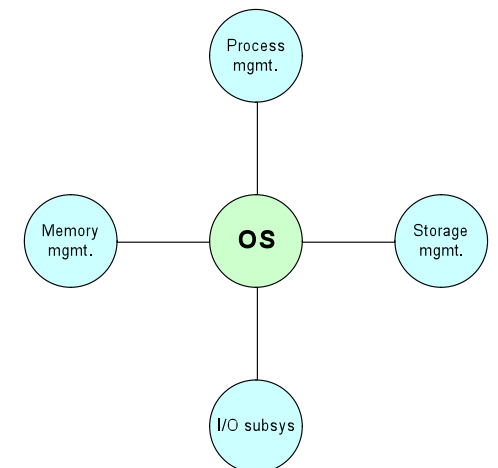
---

- ***What is a process? Is it a program?***

- ▶ No - a process is a program in execution

- ***How many processes can run at a time?***

- ▶ There can be many processes that want to run, but only one per CPU that actually runs



# OS structure - Process management

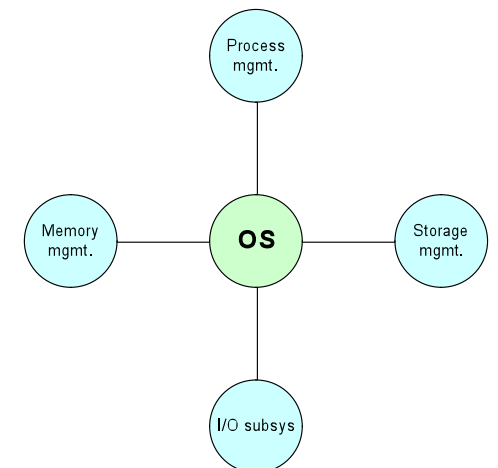
---

- ***What is a process? Is it a program?***

- ▶ No - a process is a program in execution

- ***How many processes can run at a time?***

- ▶ There can be many processes that want to run, but only one per CPU that actually runs



# OS structure - Process management

---

- ***What is a process? Is it a program?***

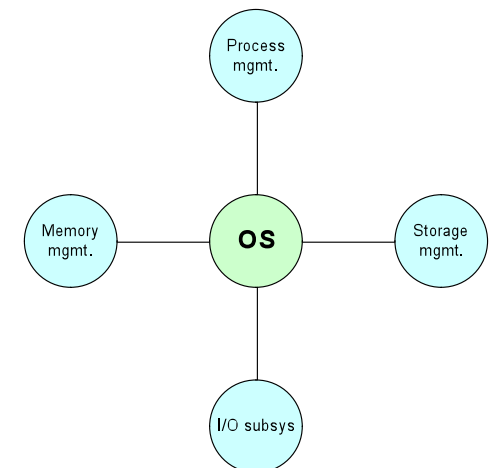
- ▶ No - a process is a program in execution

- ***How many processes can run at a time?***

- ▶ There can be many processes that want to run, but only one per CPU that actually runs

- The OS manages processes

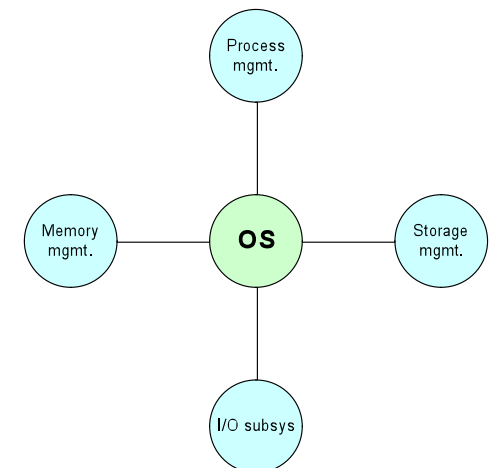
- ▶ Creates, deletes, and allocates resources for them
- ▶ Swaps them in and out of memory
- ▶ Suspends and resumes them
- ▶ Provides mechanisms for synchronization and communication between processes



# Process management – why?

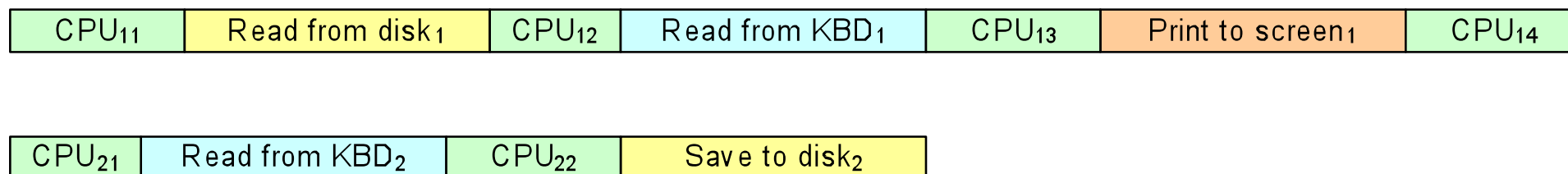
---

- Processes either compute or perform device I/O
- ***What does a process do while it performs I/O?***
  - ▶ It must wait for I/O to complete before it can resume
- ***What should the system do meanwhile?***
  - ▶ Without process management: CPU idles
  - ▶ With process management: Switch to another ready process

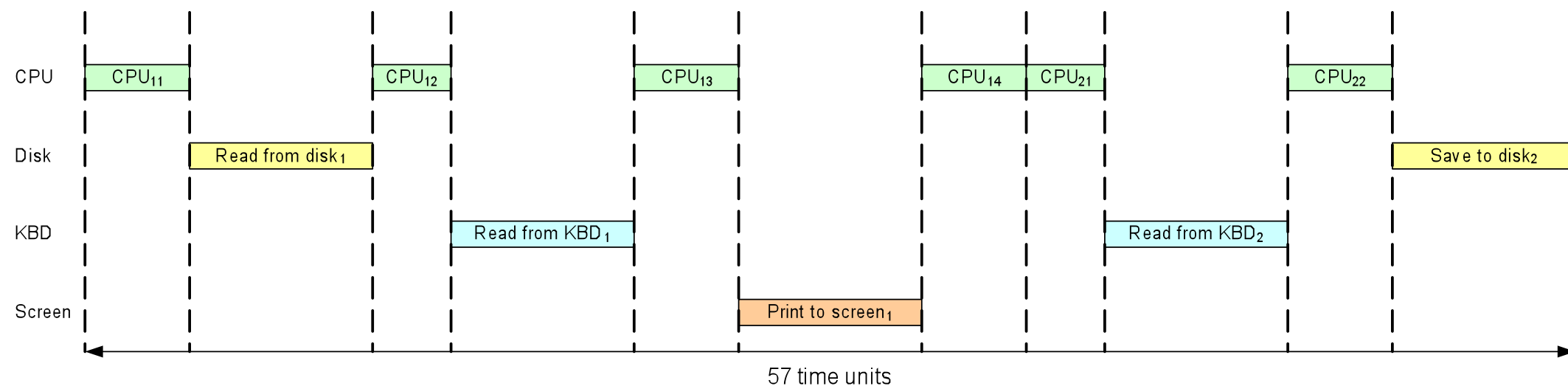


# Process management - example

- Consider two tasks:

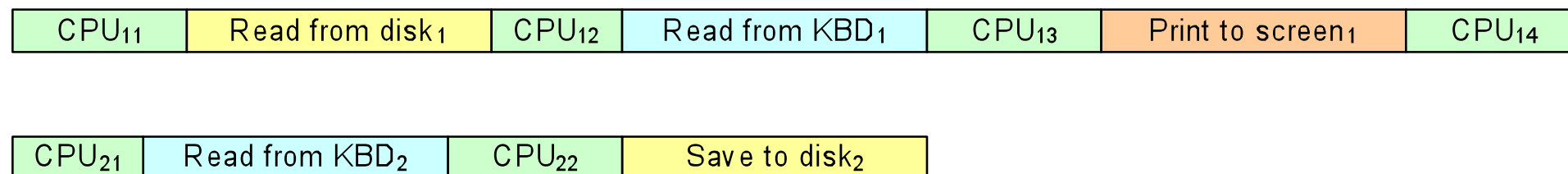


- Scheduling without resource management (batch processing)?***

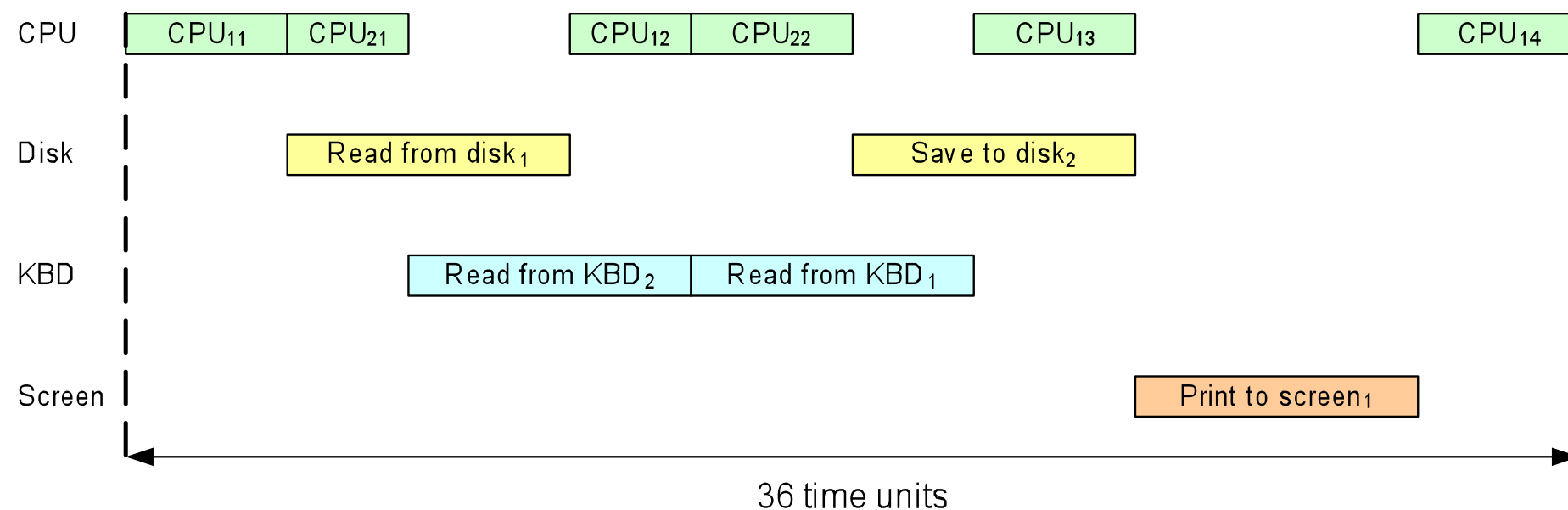


# Process management - example

- Consider two tasks:



- Scheduling with resource management?***





# Process management - protection

---

# Process management - protection

---

- Consider an “evil” process – what damage could it do?
  - ▶ Destroy, eavesdrop on, change other processes
  - ▶ Destroy OS
  - ▶ Destroy files and HW

# Process management - protection

---

- Consider an “evil” process – what damage could it do?
  - ▶ Destroy, eavesdrop on, change other processes
  - ▶ Destroy OS
  - ▶ Destroy files and HW

# Process management - protection

---

- Consider an “evil” process – what damage could it do?
  - ▶ Destroy, eavesdrop on, change other processes
  - ▶ Destroy OS
  - ▶ Destroy files and HW
- The OS guards against this using dual-mode operation (MODE bit in CPU)
  - ▶ Applications run in user mode (AKA restricted mode)
  - ▶ The OS kernel runs in kernel mode (AKA protected, privileged, supervisor mode)

# Process management - protection

---

- Consider an “evil” process – what damage could it do?
  - ▶ Destroy, eavesdrop on, change other processes
  - ▶ Destroy OS
  - ▶ Destroy files and HW
- The OS guards against this using dual-mode operation (MODE bit in CPU)
  - ▶ Applications run in user mode (AKA restricted mode)
  - ▶ The OS kernel runs in kernel mode (AKA protected, privileged, supervisor mode)

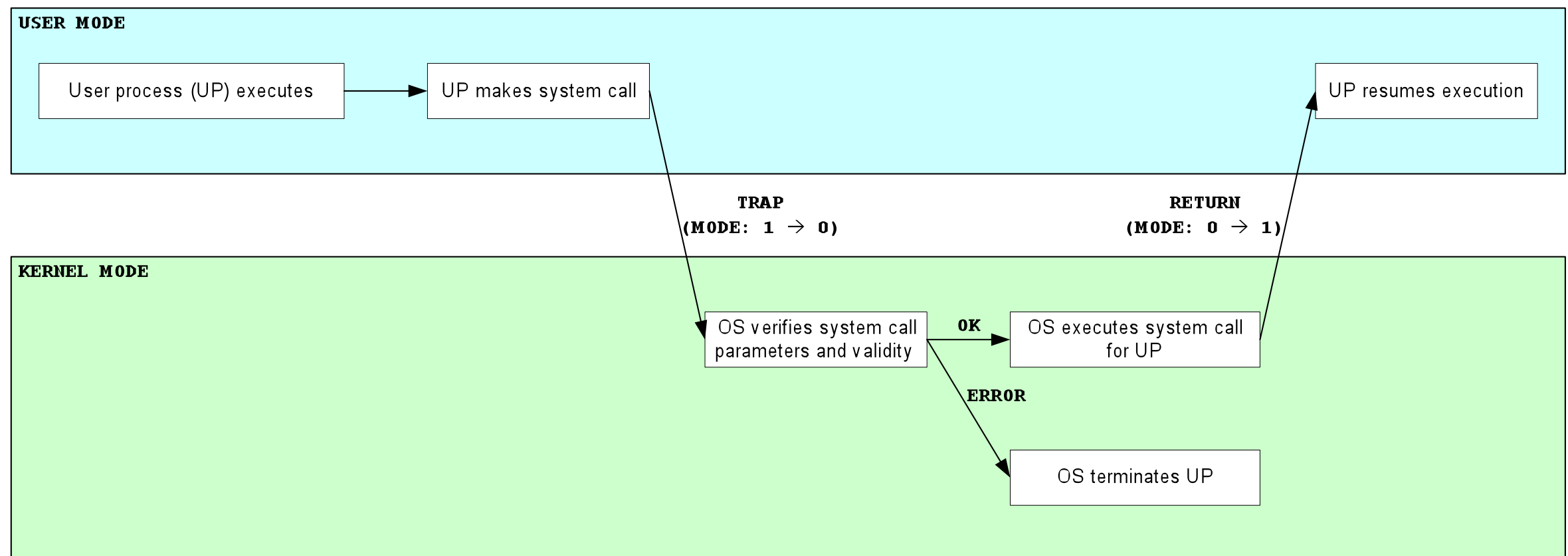
# Process management - protection

---

- Consider an “evil” process – what damage could it do?
  - ▶ Destroy, eavesdrop on, change other processes
  - ▶ Destroy OS
  - ▶ Destroy files and HW
- The OS guards against this using dual-mode operation (MODE bit in CPU)
  - ▶ Applications run in user mode (AKA restricted mode)
  - ▶ The OS kernel runs in kernel mode (AKA protected, privileged, supervisor mode)
- Potentially dangerous operations (I/O, IPC, ...) can only be done via privileged instructions
  - ▶ Restricted instructions – user and kernel mode
  - ▶ Privileged instructions – kernel mode only

# Dual-mode operation – system calls

- When processes need to perform I/O, it does so via the OS via well-defined **system calls** (version 2.6.35: 337 different syscalls)
- The OS (which is in kernel mode) verifies the system call and its parameters



# Dual-mode operation – system calls

---

- How often are system calls made?



# Dual-mode operation – system calls

---

- How often are system calls made?

```
$ ./hello  
Hello World!
```

# Dual-mode operation – system calls

---

- How often are system calls made?

```
$ ./hello  
Hello World!  
$ strace ./hello
```

# Dual-mode operation – system calls

- How often are system calls made?

[illegible]

# Dual-mode operation – system calls

- How often are system calls made?

## 56 system calls

[illegible]

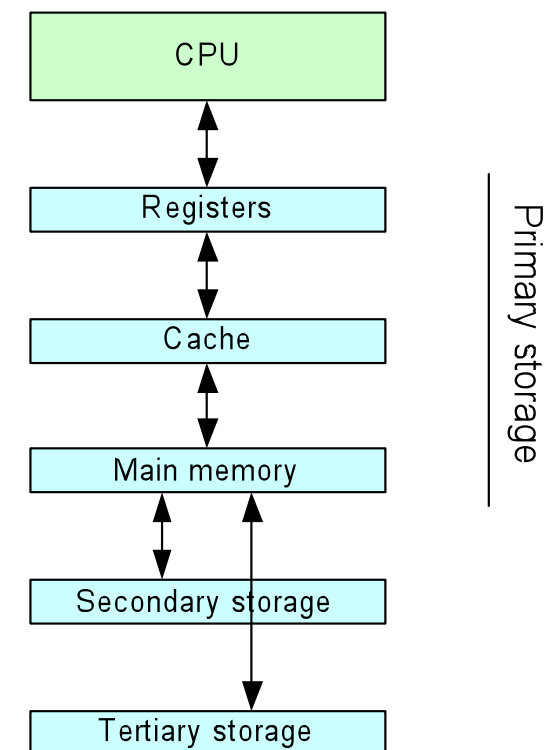
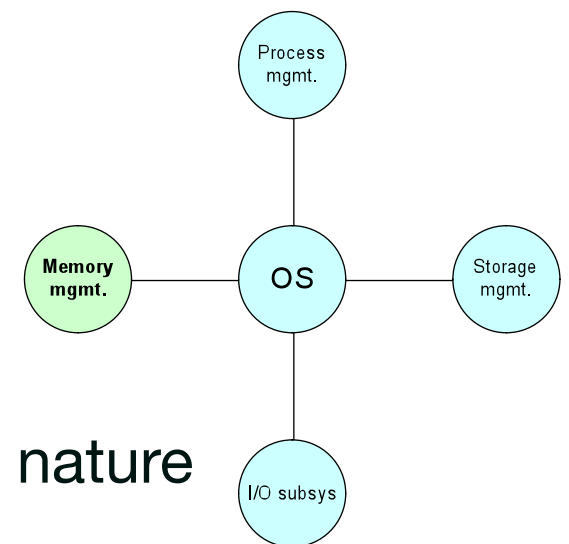
# OS Structure - Memory and storage mgmt.

- Memory management

- ▶ Keep track of several processes in memory at a time
- ▶ Decide which (parts of) processes to move in and out of memory
  - ▶ Many different algorithms depending on hardware and the OS' nature
- ▶ Allocate and deallocate memory as necessary

- Storage management

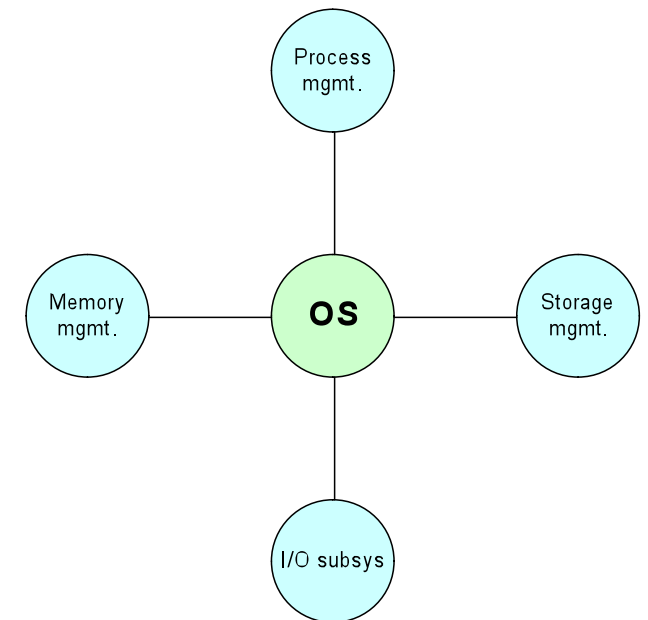
- ▶ The primary storage is never big enough to accommodate all needs
- ▶ A hierarchy of memory:
  - ▶ Size? Price (per MB)? Capacity? Bandwidth?
- ▶ Move data in/out of hierarchy



# OS Structure - I/O subsystem

---

- The I/O subsystem hides the oddities of individual I/O devices
- Instead, it provides a uniform interface (in Linux: a file)
  - ▶ The file is I/F to a device driver
  - ▶ The device driver knows how to operate the device



# Operating systems - Real-Time OS's

---

- Real-Time Operating Systems (RTOSs) are OSs intended for RT systems (!)
- ***Some key properties?***
  - ▶ Minimum interrupt latency
  - ▶ Minimum task switching latency
    - ▶ Includes known worst case latency (must be small)
  - ▶ Static task priorities
- The programmer (you!) is responsible for correct priority assignment