DCS216 Operating Systems

Lecture 02
Computer System Organization and Architecture

Feb 28th, 2024

Instructor: Xiaoxi Zhang

Sun Yat-sen University

Administrivia

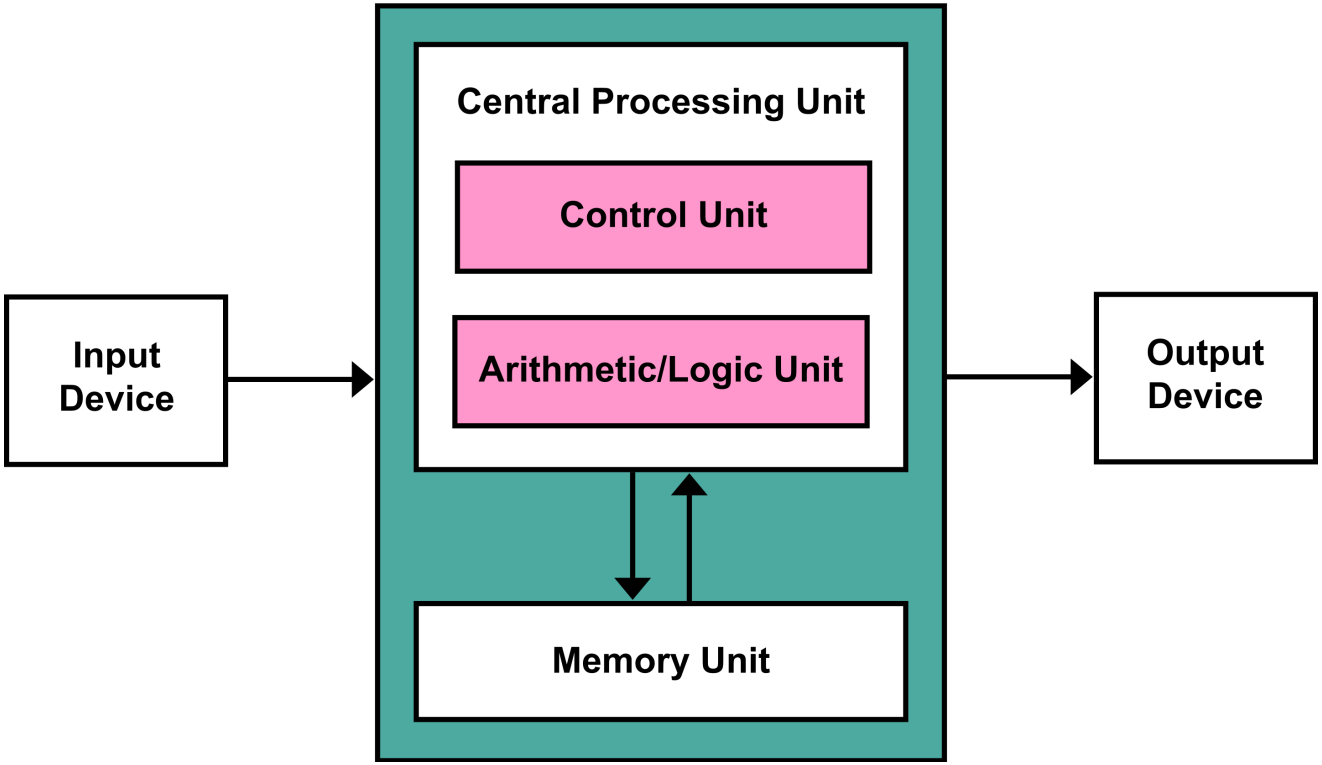
- lab01 deadline has been extended to **Wednesday Feb 28, 22:00**, please submit in time.
- Our TAs are not always available. (They have papers to write.)
- If you have a long question to ask that requires dedicated offline discussion, e.g., you have trouble in class comprehending how interrupts are implemented, you can come by TAs' office **DURING** office hours (Saturday 15:00 ~ 17:00).
- For quick questions and technical issues, you can make use of our course WeChat group. Our TAs will answer in a promptly manner when they're available, or maybe your classmates can help, too.
- Slides (PowerPoint, PDF) will be uploaded to the course homepage:
  - `https://lms.sysu.edu.cn/course/view.php?id=3336`

Computer System Organization and Architecture

- **Content**
  - Computer *Hardware* Architectures
  - Computer System Organization
    - Interrupts
    - Storage Structures
    - I/O Structures
  - Operating System Operations
    - Multiprogramming and Multitasking
    - Dual-Mode Operation
    - Timer
  - OS Resource Management
  - Computer System Architecture
    - Single-Processor Systems
    - Multiprocessor Systems
    - Clustered Systems
  - Computing Environments

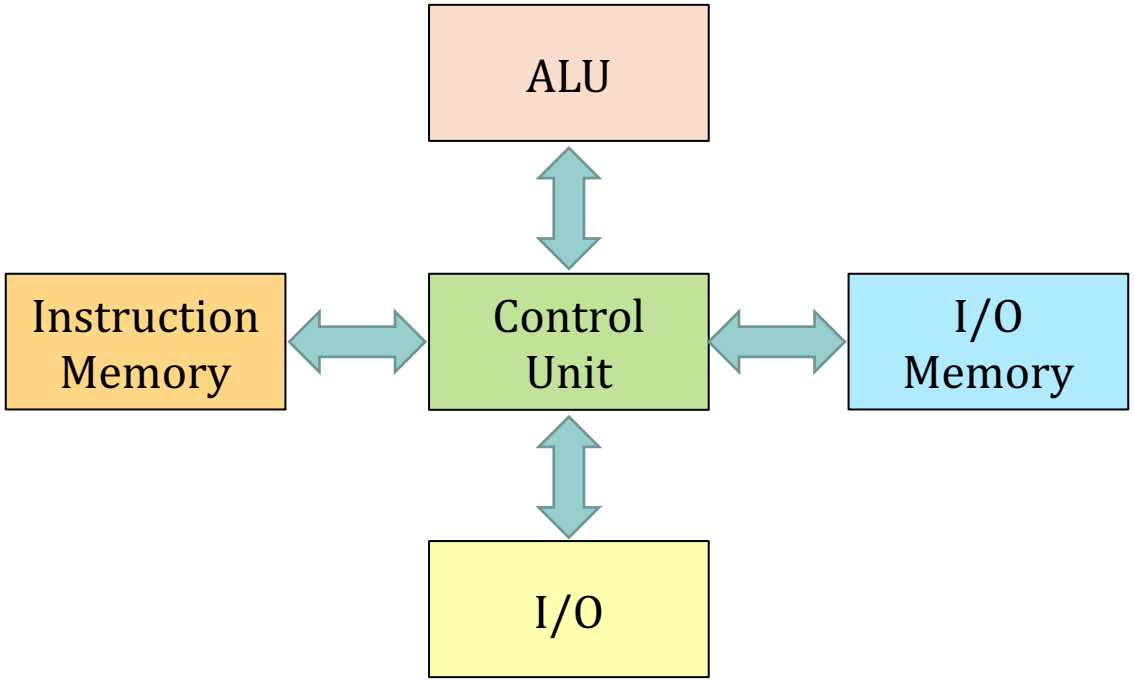
Computer Hardware Architecture

- von Neumann Architecture (冯诺依曼结构)
  - a.k.a., Princeton Architecture



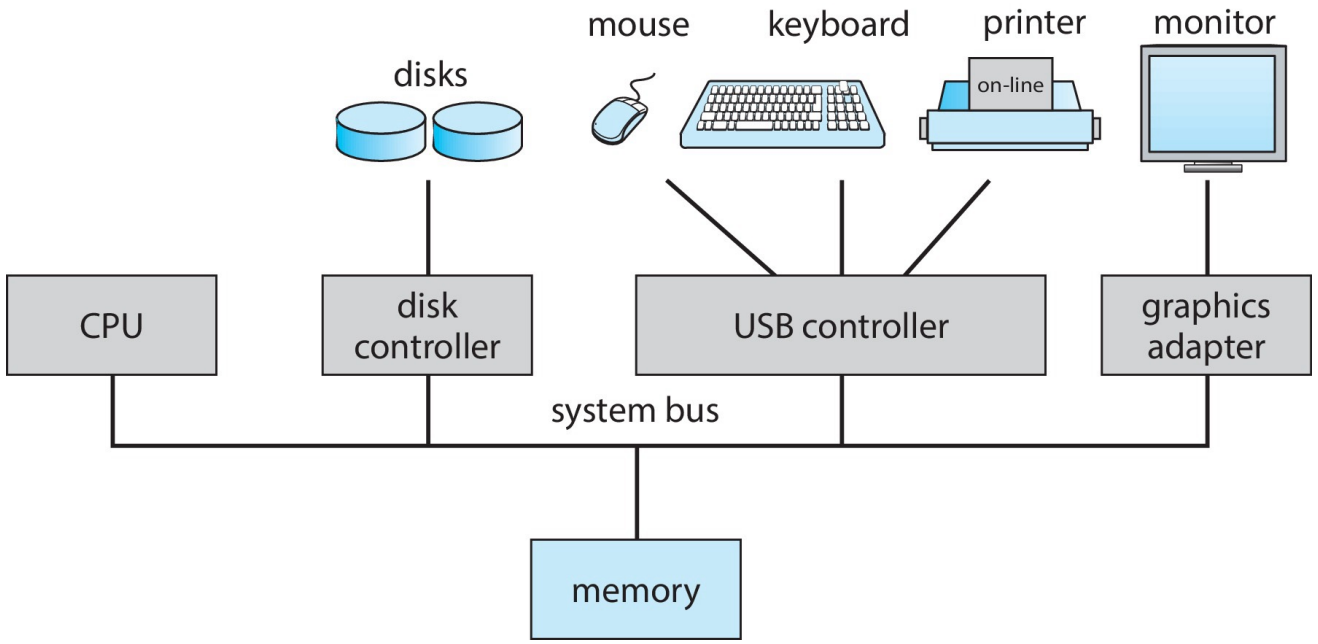
Computer Hardware Architecture

- Harvard Architecture
  - E.g., ARM9, MIC, most of DSPs, ...
    - *Note that ARM9 is no longer recommended by ARM for new IC designs, instead ARM Cortex-A, ARM Cortex-M, ARM Cortex-R cores are preferred.*
  - Modified Harvard Architecture



Computer System Organization

- A modern general-purpose computer system consists of one or more **CPUs** and a number of **device controllers** connected through a **common bus** that provides access between components and shared memory.
- Operating Systems have a **device driver** for each **device controller**.
- The CPU and the device controllers can execute **in parallel**, competing for memory cycles.

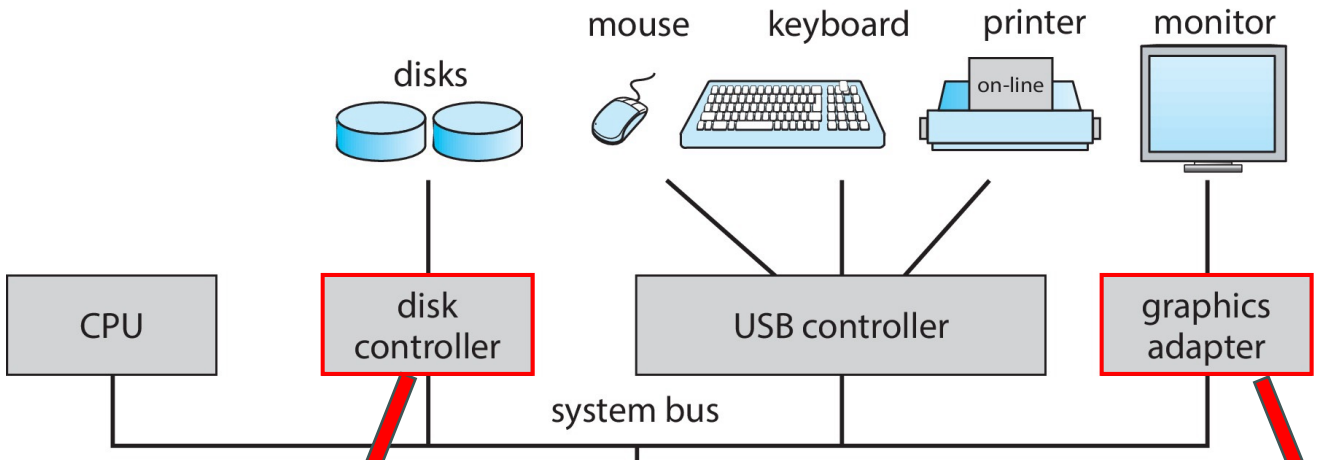


Computer System Organization


```
$ lspci -k
00:00.0 Host bridge: Intel Corporation Device 4660 (rev 02)
        DeviceName: Onboard - Other
        Subsystem: Dell Device 0aac
00:01.0 PCI bridge: Intel Corporation Device 460d (rev 02)
        Kernel driver in use: pcieport
00:02.0 VGA compatible controller: Intel Corporation Device 4680 (rev 0c)
        DeviceName: Onboard - Video
        Subsystem: Dell Device 0aac
        Kernel modules: i915
......
```

`lspci` is a utility that displays PCI devices.

`-k` shows kernel drivers handling each device.

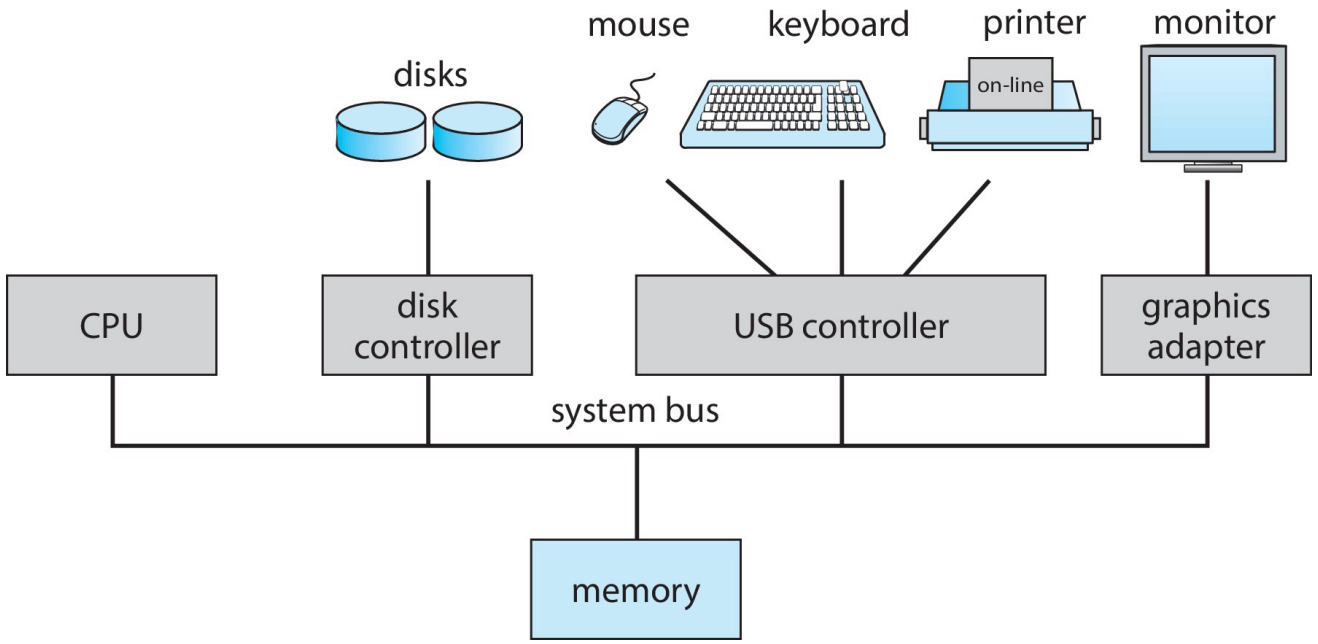



```
02:00.0 Non-Volatile memory controller: KIOXIA
Corporation Device 000d
        Subsystem: KIOXIA Corporation Device 0
        Kernel driver in use: nvme
        Kernel modules: nvme
```

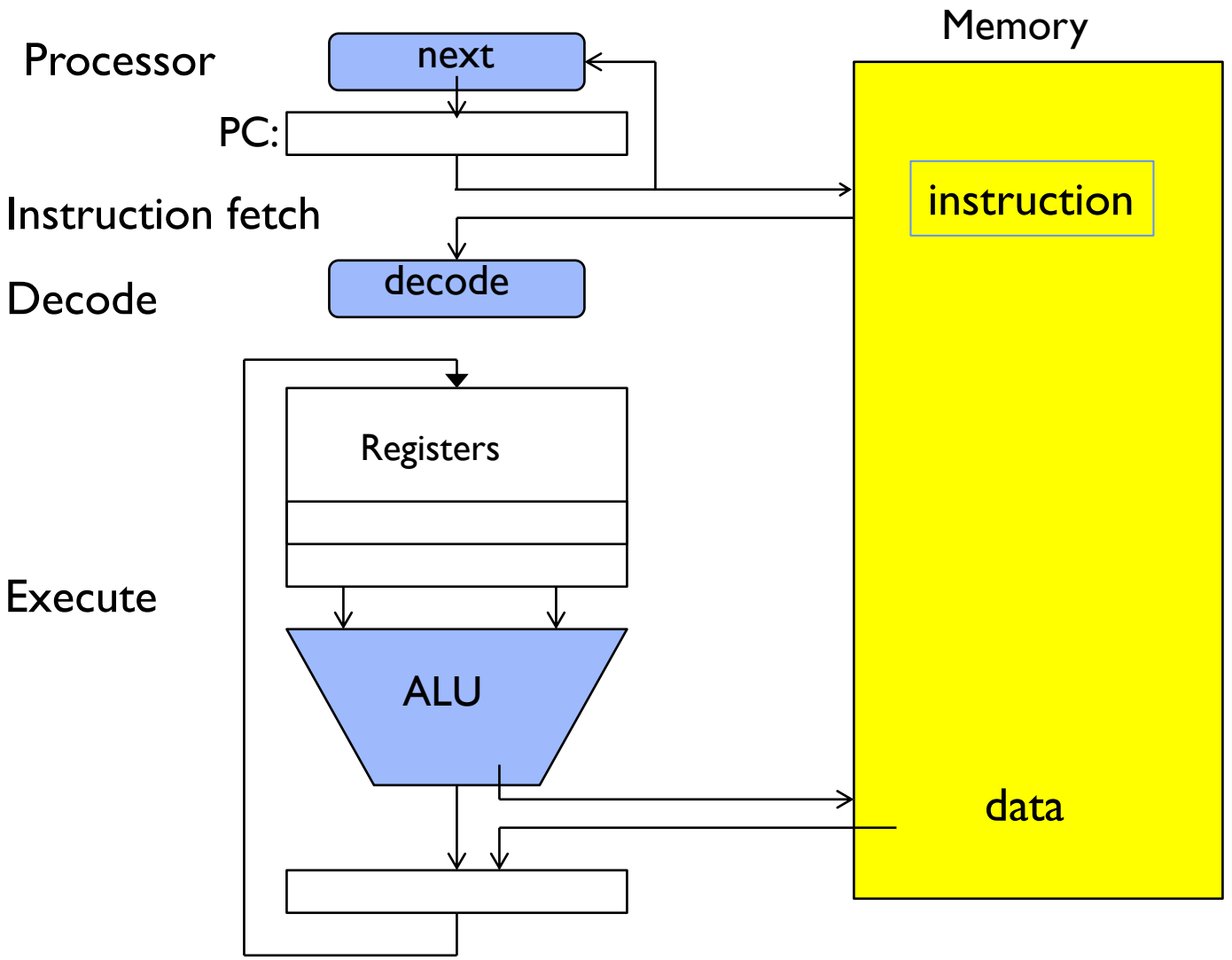
```
01:00.0 VGA compatible controller: NVIDIA
Corporation Device 2216 (rev a1)
        Subsystem: Dell Device 0890
        Kernel driver in use: nvidia
        Kernel modules: nvidiafb, nvidia
```

Computer System Organization

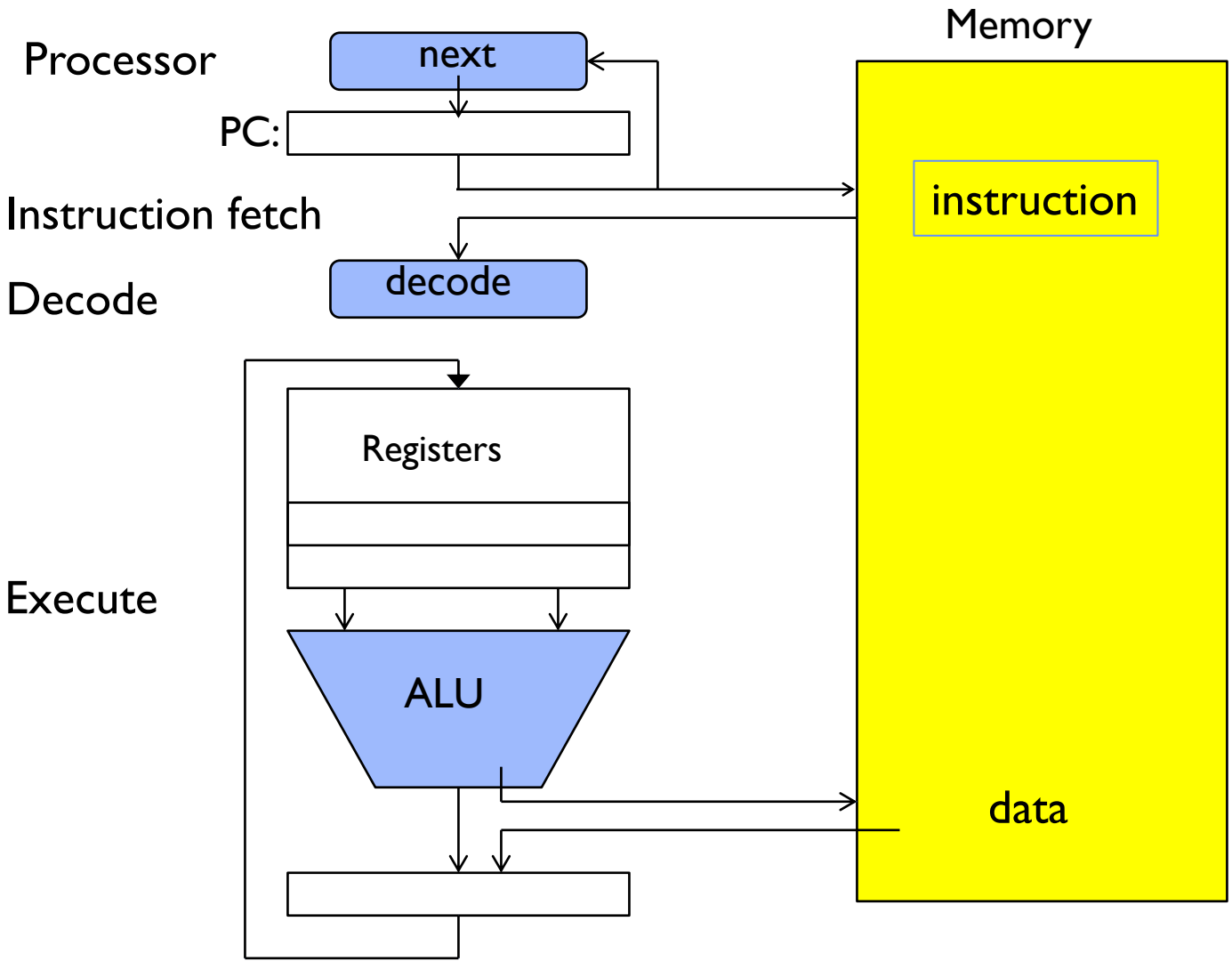
- How does such a system operate?
  - via **Interrupts**!



- How does a CPU work?

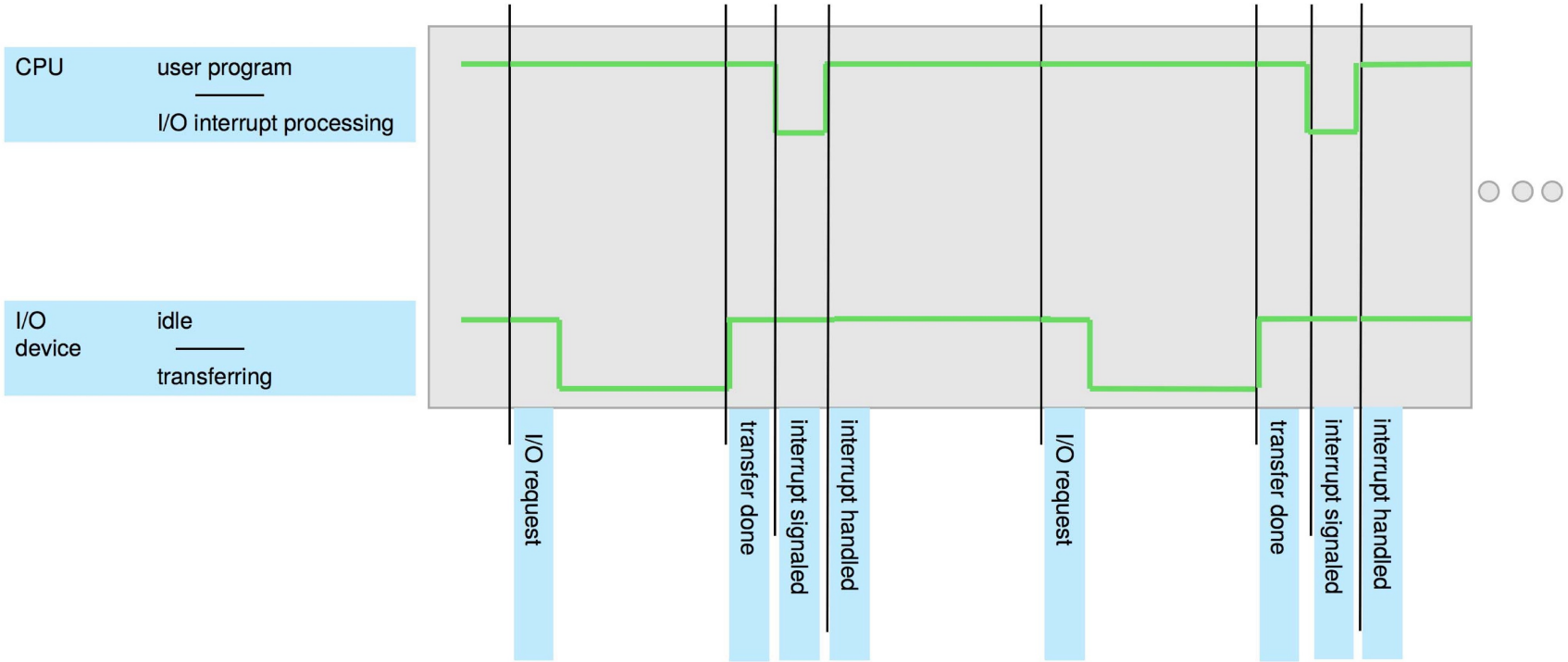


- **What will happen to a CPU without Interrupts?**
  - CPU becomes deterministic; no User-Computer interactions!



Interrupts

- Interrupts are a key part of how OS and hardware **interact**.
  - Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the **system bus**(系统总线).
  - The CPU stops what it is doing and immediately transfers execution to an interrupt service routine, through **interrupt vector**.
  - On completion, the CPU resumes the interrupted computation



Interrupts

- Interrupts are an important part of a computer architecture. Each computer design has its own interrupt mechanism.
- The basic interrupt mechanism works as follows. It enables the CPU to respond to an asynchronous event, as when a device controller becomes ready for service.
  - CPU senses interrupt-request line after executing every instruction
  - A controller may assert a signal on the line. CPU reads the interrupt number and jumps to the interrupt-handler routine (ISR).
  - A table of pointers to ISR addresses (this table is called the interrupt vector) is used to provide the necessary speed.
    - The table is stored in low memory
    - It is indexed by a unique interrupt number given with the interrupt request
    - Windows and UNIX dispatch interrupts in this manner

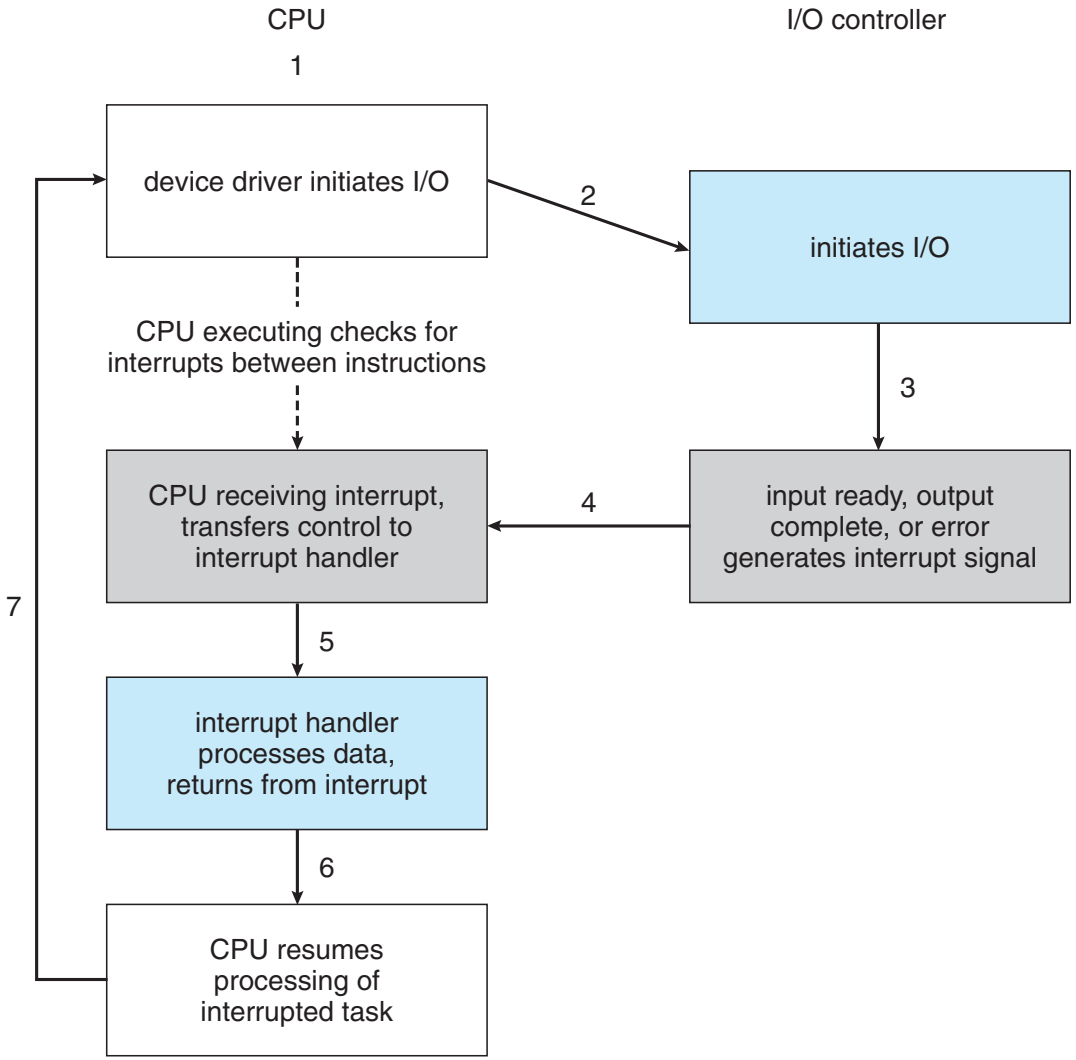
Interrupts

- Intel processor interrupt vector

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Interrupts

- Interrupt-driven I/O cycle (中断驱动的I/O周期)



Interrupts

- Interrupt Controller
  - In a modern OS, more sophisticated interrupt handling features are provided by the CPU and the interrupt-controller hardware.
    - the ability to defer interrupt handling during critical processing
    - an efficient way to dispatch to the proper interrupt handler for a device
    - multi-level interrupts, so that the OS can distinguish between high- and low-priority interrupts and can respond with the appropriate degree of urgency

- **Interrupts**
  - Maskable and Non-maskable Interrupts
    - Most CPUs have two distinct interrupt request lines
      - non-maskable interrupt line
        - reserved for events such as unrecoverable memory errors
      - maskable interrupt line
        - used by device controllers to request service
        - can be turned off by CPU before execution of critical instruction sequences that must not be interrupted

Interrupts

- Intel processor interrupt vector

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

non-maskable

maskable

Interrupts

Interrupt Types and Attributes

- An Operating System is interrupt-driven. There are three basic types:
  - **Hardware Interrupts** / External Interrupts (**硬件中断**/外部中断)
  - **Exceptions**/Traps (**异常**/陷阱)
  - **System Calls** (**系统调用**)
- Various interrupt attributes
  - Asynchronous vs Synchronous
  - External/Hardware vs Internal/Software
  - Implicit vs Explicit

	Interrupt Types		
	External/Hardware	Internal/Software	
Asynchronous	Hardware Interrupts		Implicit
Synchronous		Exceptions	
		System Calls	Explicit

Storage Structures

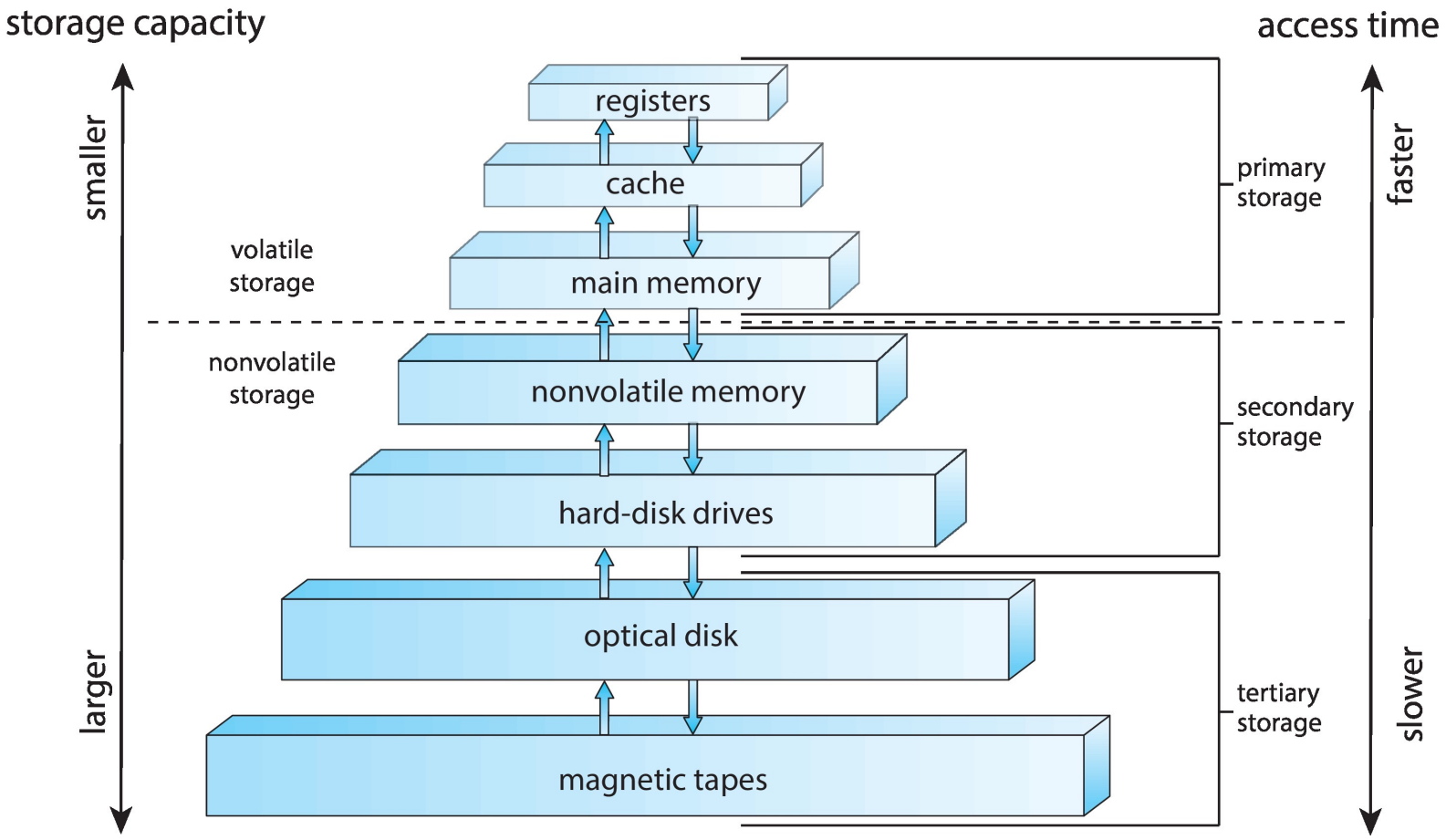
- Main Memory
  - the only large storage media that the CPU can directly access
  - usually too small to store all needed programs and data permanently
  - volatile (易失的)
    - loses its content when power is turned off or otherwise lost
- Secondary Storage
  - large capacity
  - nonvolatile memory (NVM, 非易失存储) devices
  - HDDs are the most common secondary-storage devices
    - rigid metal or glass platters covered with magnetic recording material
    - cylinders, tracks and sectors
    - The disk controller determines the logical interaction between the device and the system bus
  - SSDs or flash drives are increasingly popular

Storage Structures

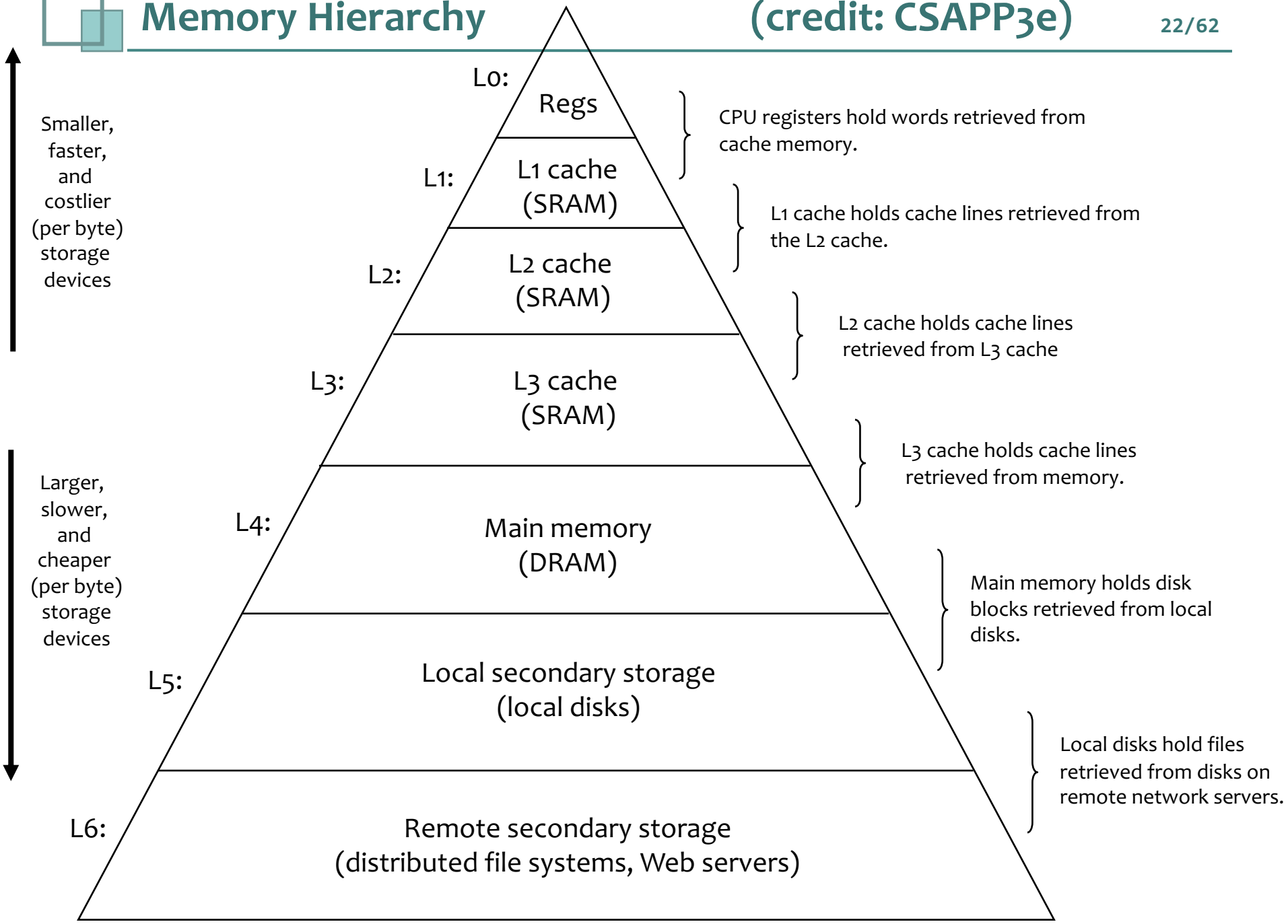
- Caching
  - Caching is an important principle performed at many levels
    - Information is copied from slower to faster storage for processing
      - hardware, operating system, apps, etc
  - Faster storage (cache) is checked first to see if it's already there
    - If yes, the CPU retrieve directly from the cache
    - Otherwise, data copied from slower storage to cache
  - Cache is usually much smaller than storage being cached
    - Cache management is an important design problem
    - Cache size and replacement policy matter
  - In most cases, cache is referred in particular to the SRAM within the CPU

Storage Structures

Storage Hierarchy



Memory Hierarchy (credit: CSAPP3e)



Storage Structures

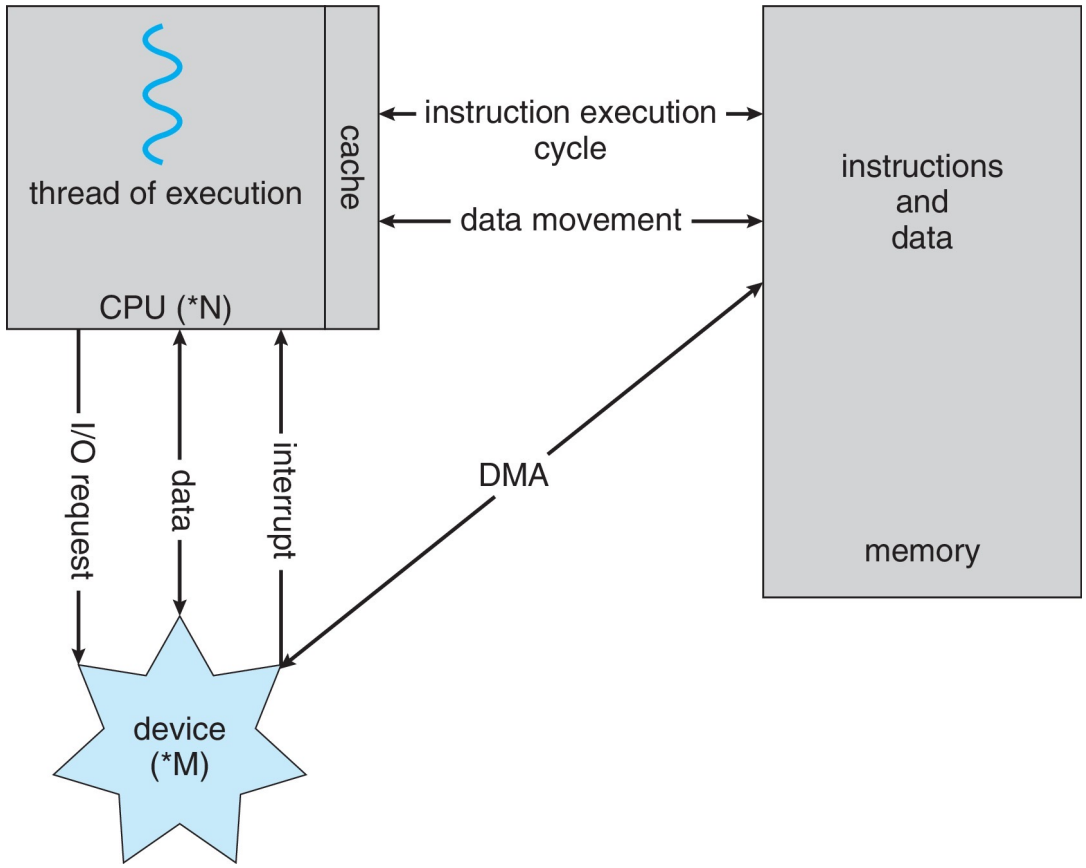
Storage Hierarchy

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

I/O Structures

- A large portion of OS code is dedicated to managing I/O
  - It is important to the reliability and performance of a system
  - It is also because of the varying nature of the devices
- Interrupt-driven I/O
  - fine for moving small amounts of data (maybe 1 or 64 bytes)
  - high overhead when used for bulk data transfer
- Direct Memory Access (DMA)
  - After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data (2^{16} bytes) directly to or from the device and main memory, with no intervention by the CPU
  - Only one interrupt is generated per block, to tell the device driver that the operation has completed, rather than one interrupt per byte generated for low-speed devices
  - While the device controller is performing these operations, the CPU is available for other tasks

How a Modern Computer Works



Operating System Operations

- **Operating System Boots**
  - Bootstrap program – simple code to:
    - initialize the system
    - load the kernel into memory
  - Kernel loads
  - Start system daemons (services provided outside of the kernel)
    - In Linux, "`systemd`" spawns many other daemons
  - Kernel interrupt driven (hardware and software)
    - Hardware interrupt by one of the devices
    - Software interrupt (exception or trap):
      - Software error (e.g., division by zero)
      - Request for operating system service – system call
      - Other process problems include infinite loop, processes modifying each other or the operating system

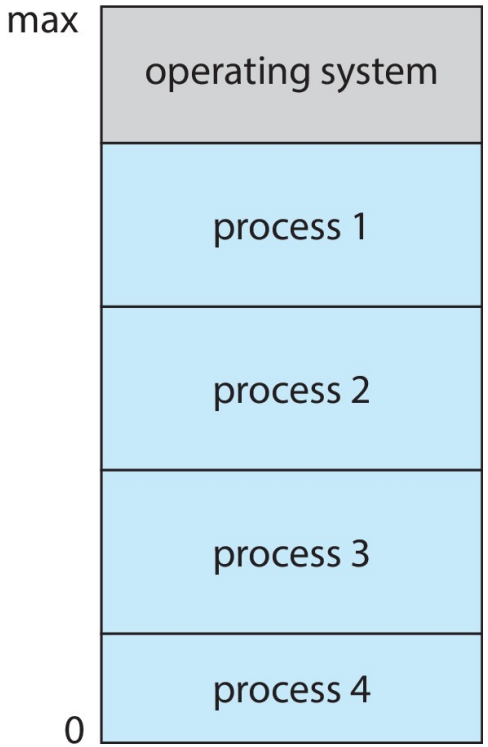
Operating System Operations

- **Operating System Operations**
  - **Multiprogramming** and **Multi-tasking**
  - **Dual-Mode** Operation
  - **Timer**

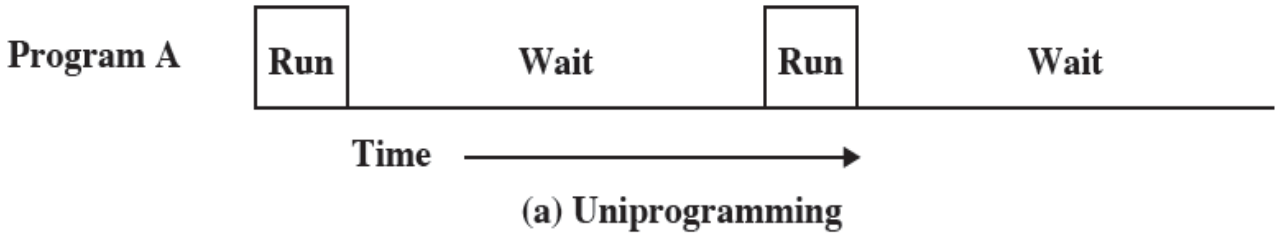
Operating System Operations

- **Multiprogramming (Batch System)**
  - Single user cannot always keep CPU and I/O devices busy
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job is selected and run via job scheduling
  - When job has to wait (e.g., for I/O), OS switches to another job
  - Batch multiprogramming (批处理多道程序设计) does not support interaction with users.

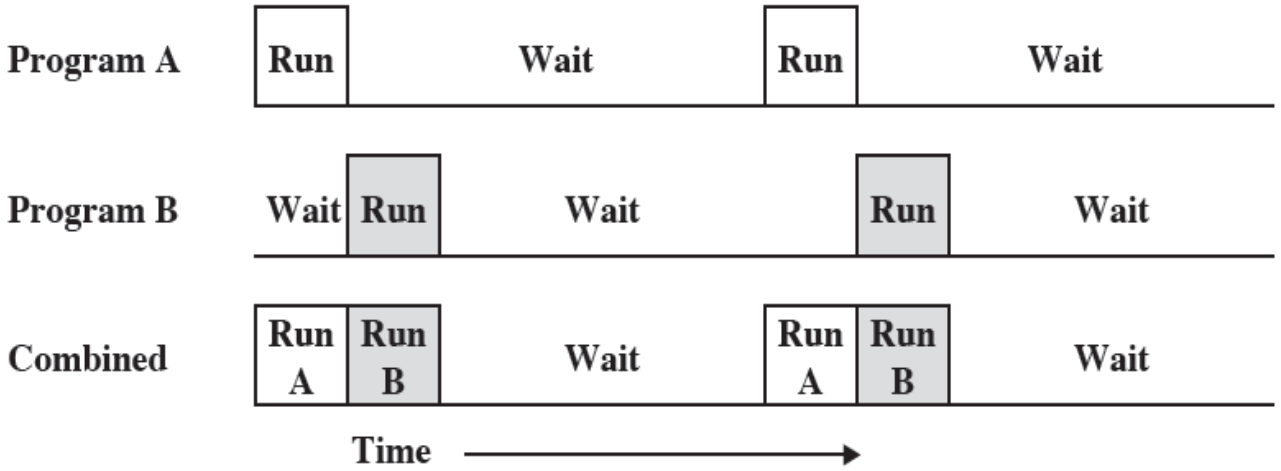
- **Multiprogramming (Batch System)**
  - Memory Layout for Multiprogrammed Systems



- **Multiprogramming (Batch System)**
  - CPU efficiency of Batch Multiprogramming

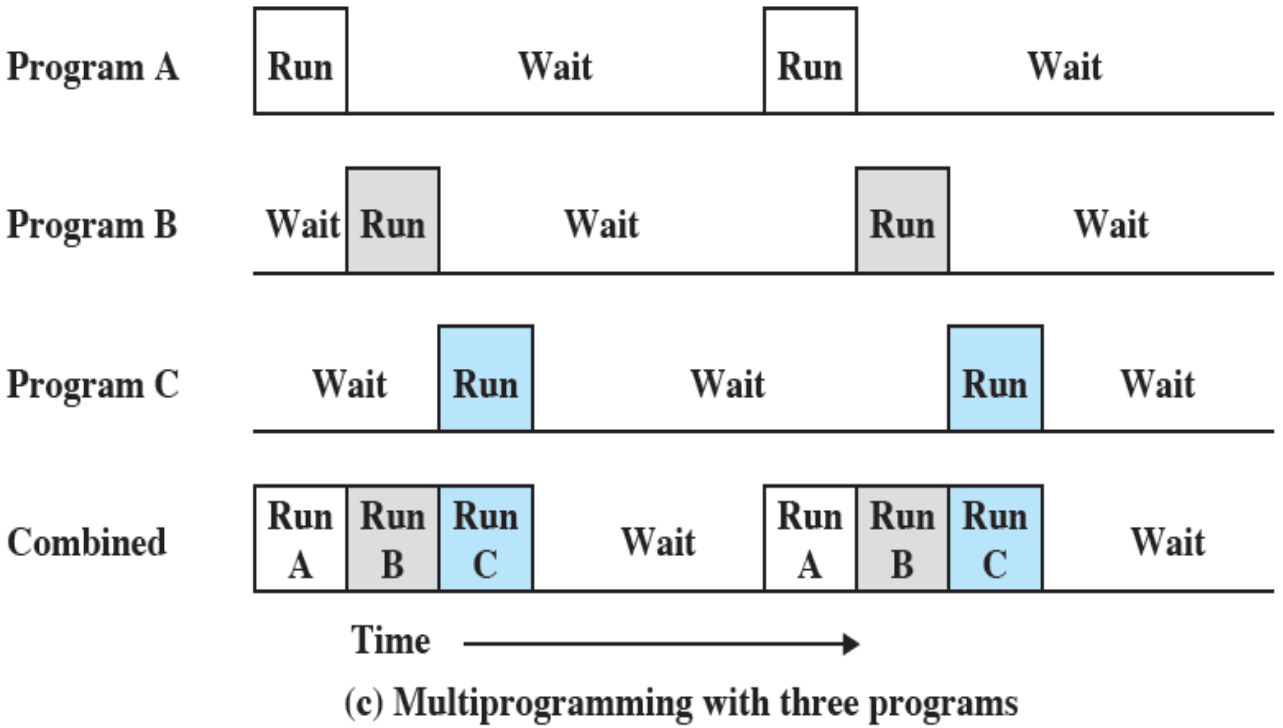


(a) Uniprogramming



(b) Multiprogramming with two programs

- **Multiprogramming (Batch System)**
  - CPU efficiency of Batch Multiprogramming



(c) Multiprogramming with three programs

- **Multiprogramming (Batch System)**
  - Requirements for Multiprogramming
    - Hardware support
      - I/O interrupts and DMA controllers
        - in order to execute instructions while I/O device is busy
      - Timer interrupts for CPU to gain control
      - Memory management
        - several ready-to-run jobs must be kept in memory
      - Memory protection (code and data)
    - Software support from the OS
      - Scheduling, i.e., which program should run next
      - Manage resource contention

Operating System Operations

Multitasking (Time-Sharing)

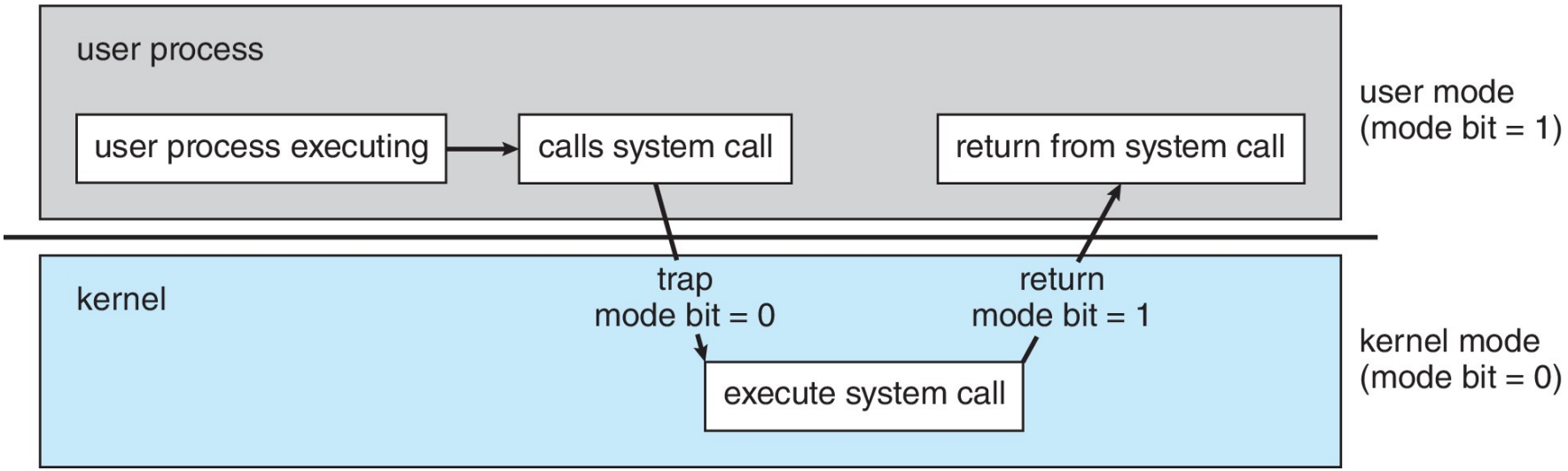
- Time-Sharing extends Batch Multiprogramming to handle multiple interactive jobs.
  - Interactive Multiprogramming (交互式多道程序设计)
  - Multiple users simultaneously access the system through terminal
  - Processor's time is shared among multiple users
- The CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
  - Response time should be < 1 seconds
  - Each user has at least one program executing in memory → process
  - If several jobs ready to run at the same time → CPU Scheduling
  - If processes don't fit in memory, swapping moves them in and out to run
  - Virtual Memory allows execution of processes not completely in memory

- **Multitasking (Time-Sharing)**
  - Why would Time-Sharing work?
    - Humans are slow; CPUs are fast
      - A typical user needs 2 seconds of processing time per minute
      - Then many users should be able to share the same system without noticeable delay in the computer reaction time.
      - The user should get a good response time.
    - It fits the economical basis of mainframe computer installation.

- **Dual-Mode Operation**
  - allows OS to protect itself and other system components
    - User Mode and Kernel Mode
  - Mode bit provided by the hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - When a user is running: mode bit is "user"
    - When kernel code is executing: mode bit is "kernel"
  - How do we guarantee that user does not explicitly set the mode bit to "kernel"?
    - System call changes mode to kernel, return from call resets it to user
  - Some instructions designated as privileged, only executable in kernel mode

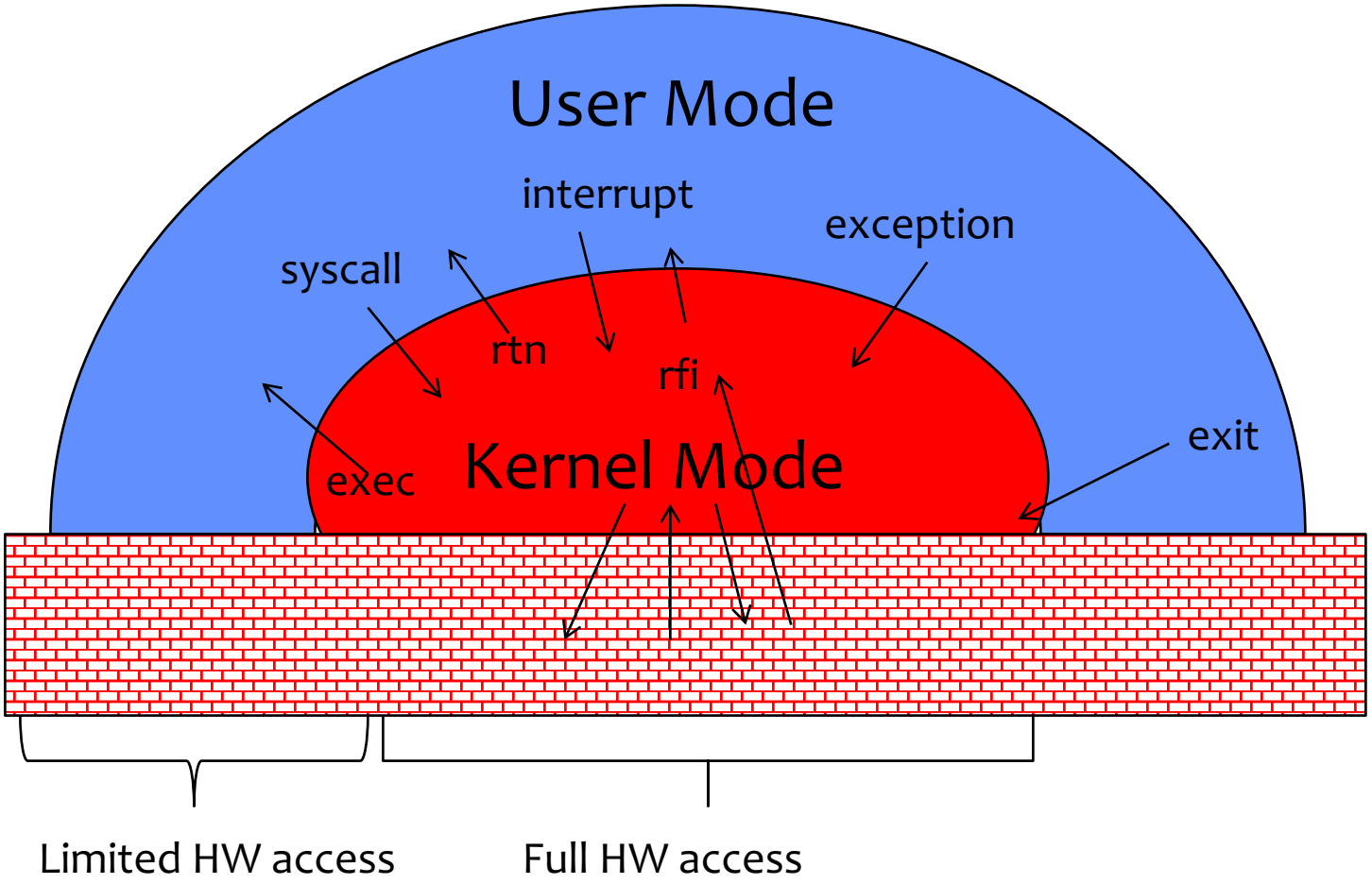
Dual-Mode Operation

- Transition from User to Kernel Mode



Dual-Mode Operation

- Transition from **User** to **Kernel** Mode, or from **Kernel** to **User** Mode



Dual-Mode Operation

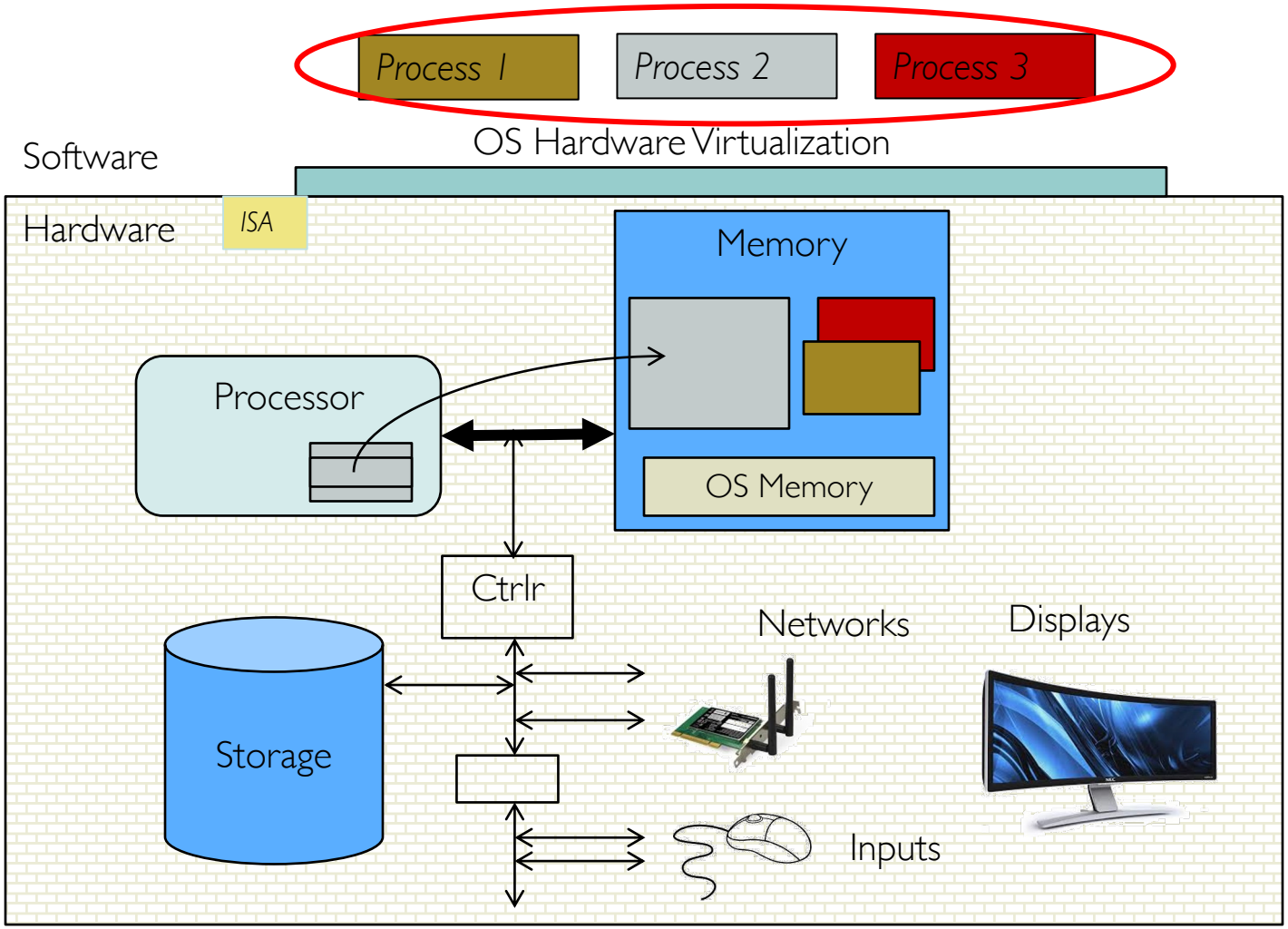
- 3 Types of Mode Transfer
  - **Syscalls** (System Calls，系统调用)
    - Process requests a system service, e.g., exec(), or exit()
    - Like a function call, but "outside" the process
    - Does not have the address of the system function to call
    - Save the syscall id and args in registers, and execute `syscall`
  - **Interrupts** (Hardware Interrupts，硬件中断)
    - External asynchronous event triggers context switch
    - e.g., Timer, I/O devices
    - Independent of user process
  - **Exceptions** (also called traps，异常、陷阱)
    - Internal synchronous event in process triggers context switch
    - e.g., Protection violation (segmentation fault), Division by zero, ...
- All 3 types are **Unprogrammed Control Transfer**
  - Outside the programmer's control
  - implemented via **Interrupt Vector (中断向量表)**.

- **Timer**
  - Timer to prevent infinite loop (or process hogging resources)
    - Timer is set to interrupt the computer after some time period
    - Keep a counter that is decremented by the physical clock
    - Operating system set the counter (privileged instruction)
    - When counter reaches zero, generate an interrupt
    - Set up before scheduling process to regain control or terminate program that exceeds allotted time

- **Resource Management**
  - Process Management
  - Memory Management
  - File-System Management
  - Mass-Storage Management
  - Cache Management
  - I/O Management

- **Resource Management**
  - Process Management

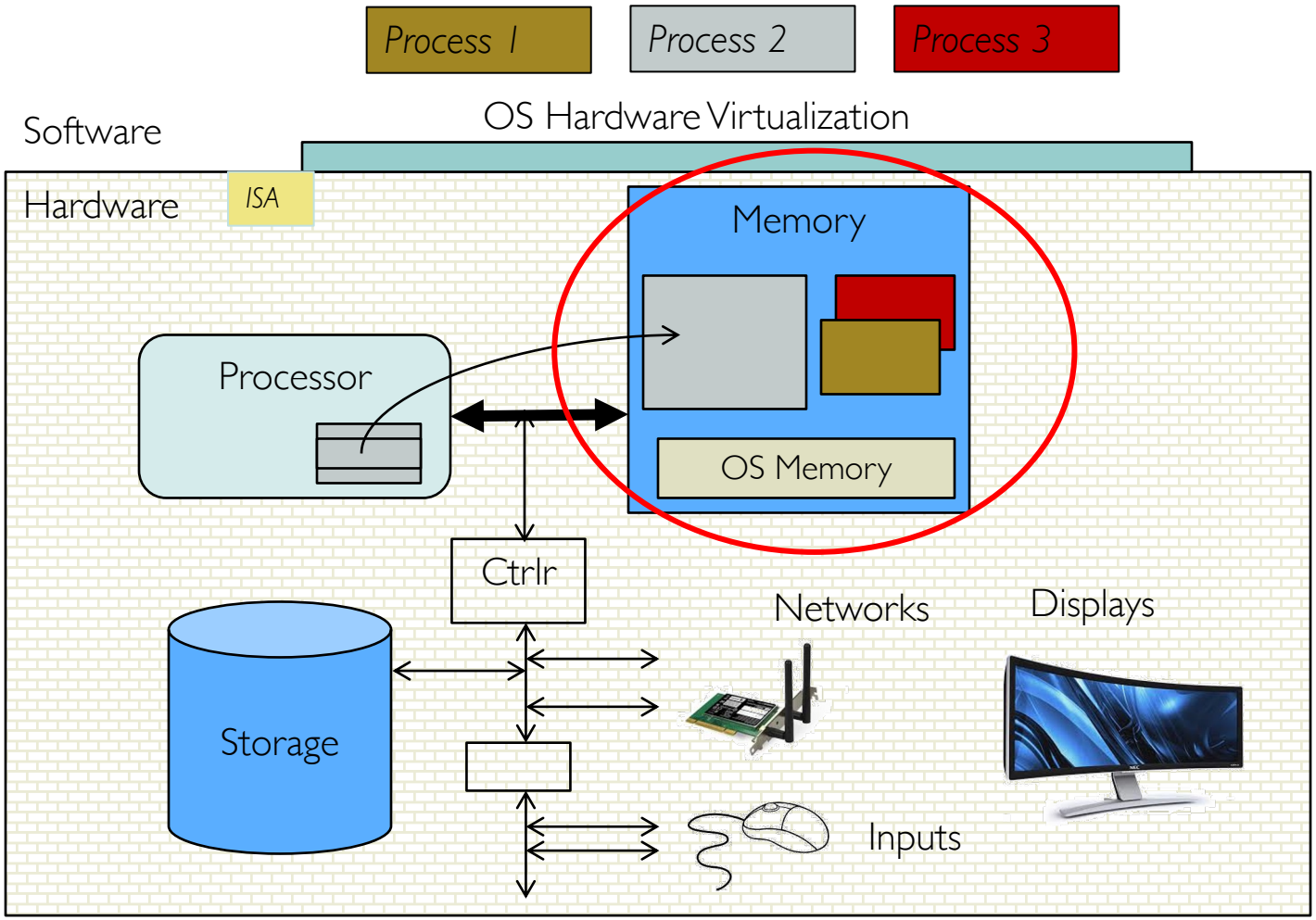
- Creating and deleting processes
- Scheduling processes
- Suspending and resuming processes
- Process communication and synchronization



Resource Management

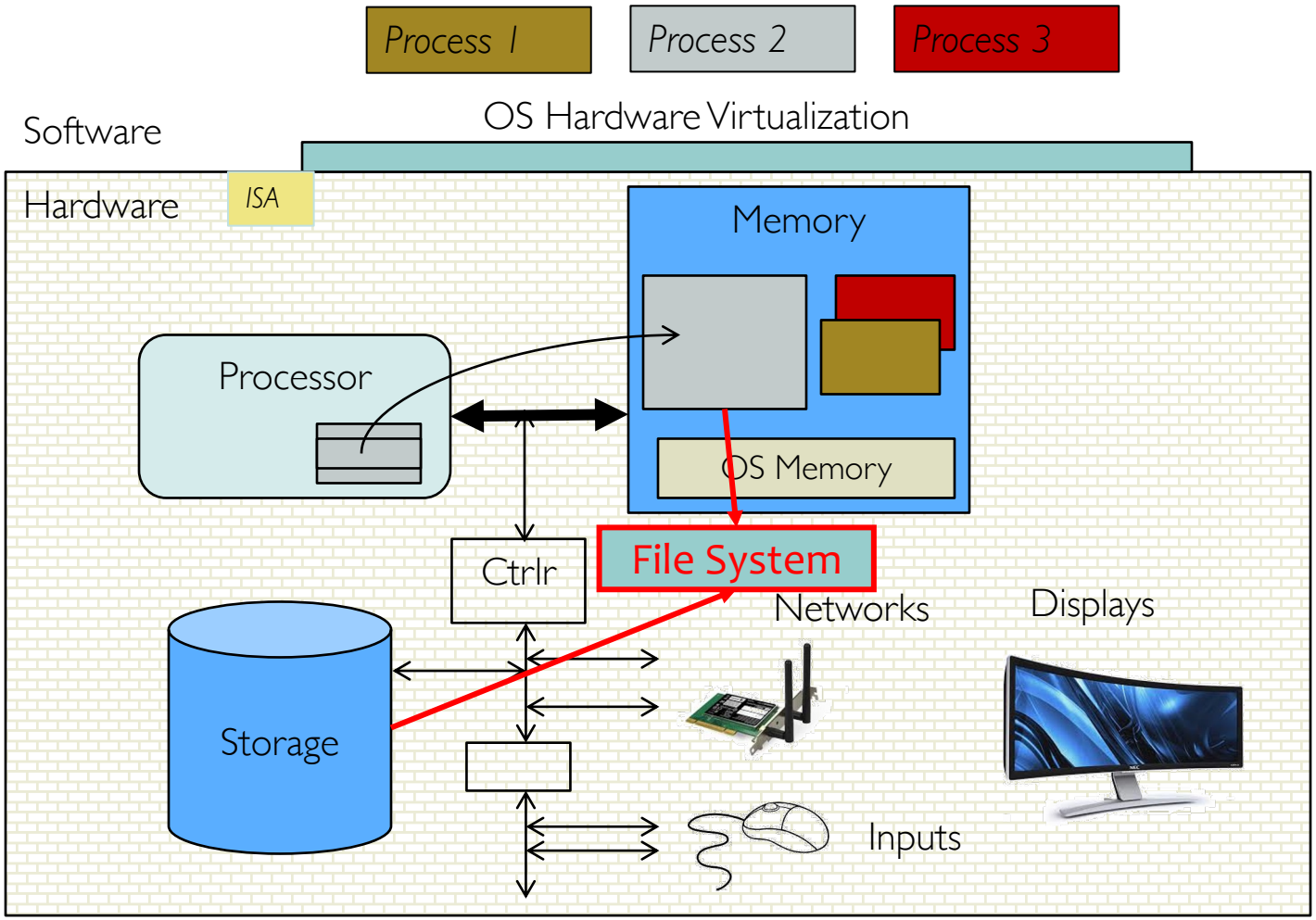
Memory Management

- Keep track of which parts of memory are used
- malloc() and free()
- Decide which processes to move in/out of memory

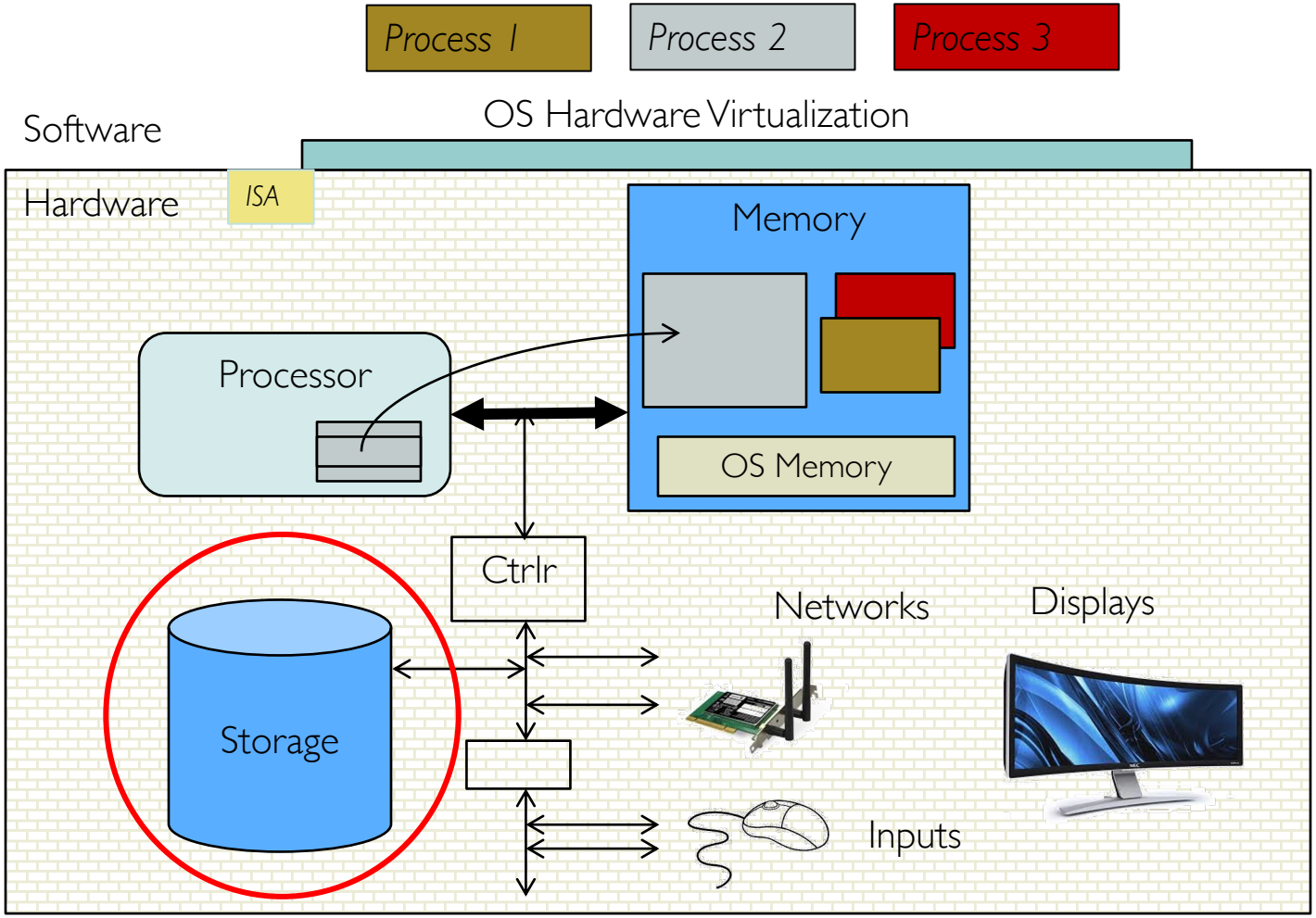


- **Resource Management**
  - File-System Management

"Everything is a FILE."
-- UNIX Philosophy

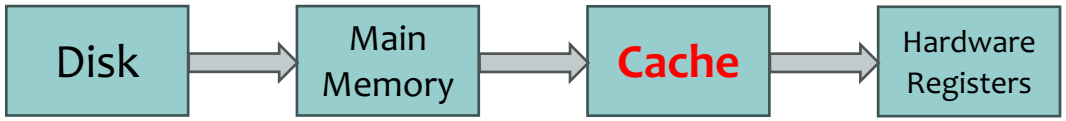


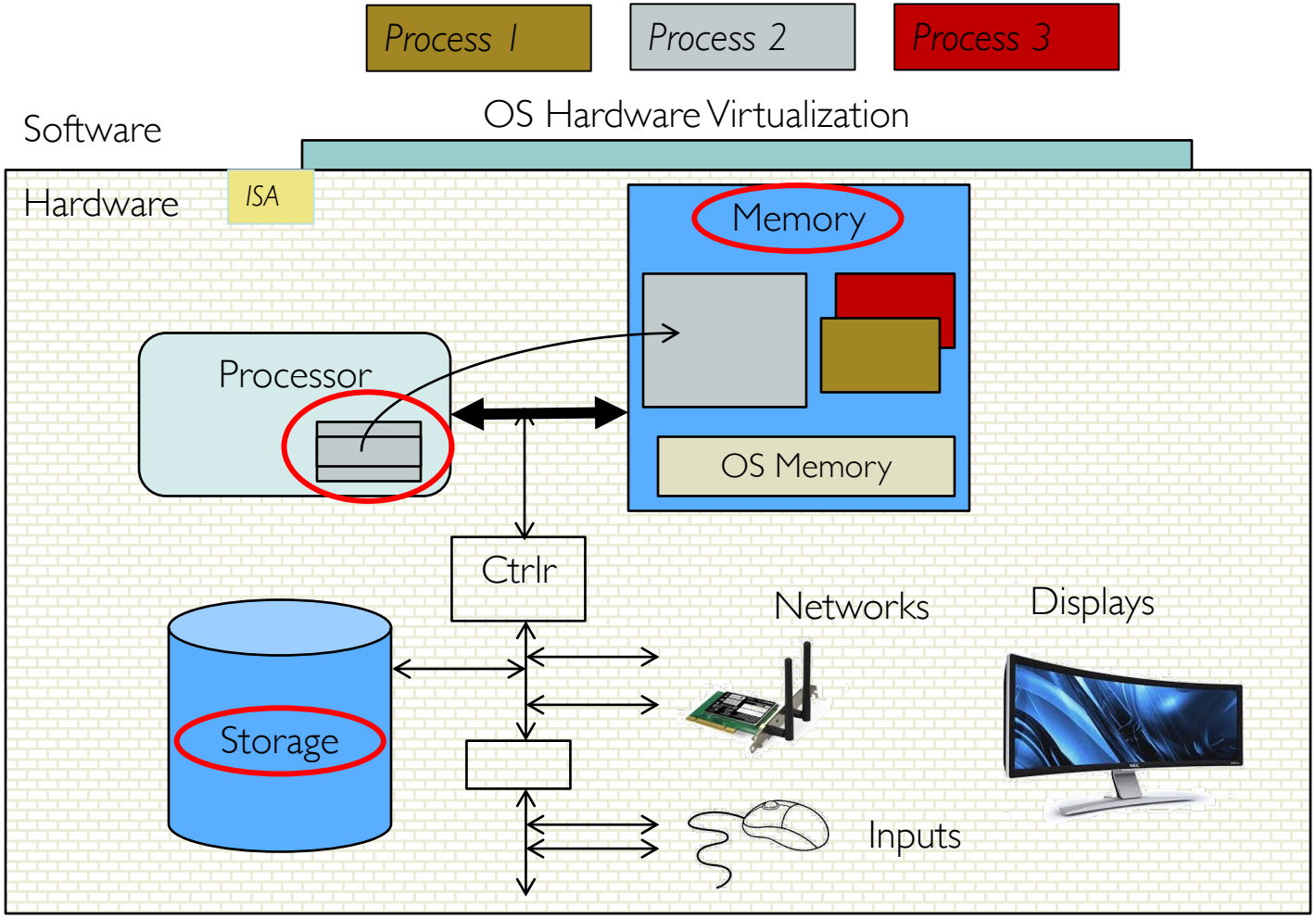
- **Resource Management**
  - Mass-Storage Management
    - Mounting/unmounting storage
    - Storage allocation
    - Disk scheduling
    - Disk partitioning



Resource Management

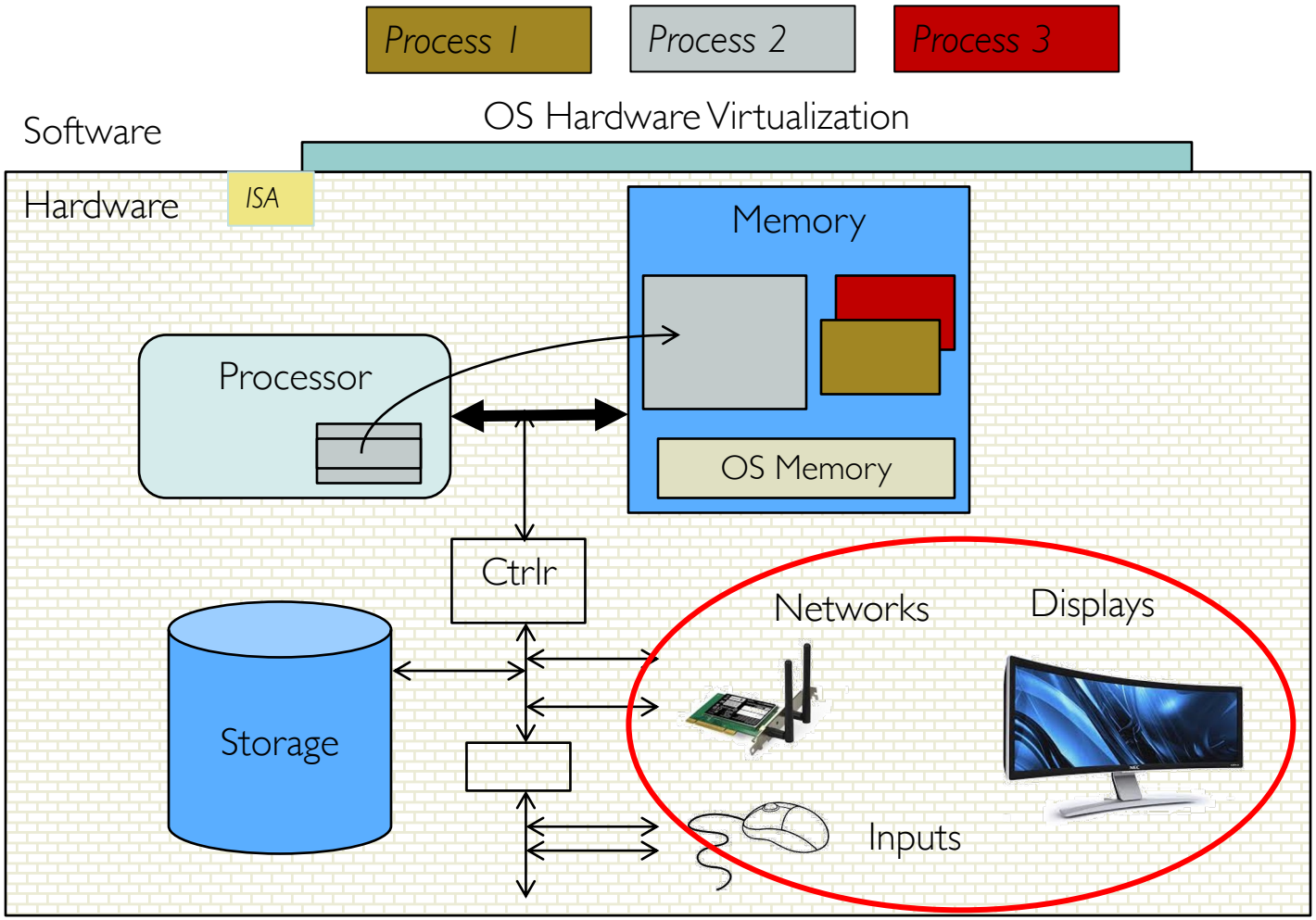
- Cache Management





- **Resource Management**
  - I/O Management

- A memory-management component that includes buffering, caching, and spooling
- A general device-driver interface
- Drivers for specific hardware devices

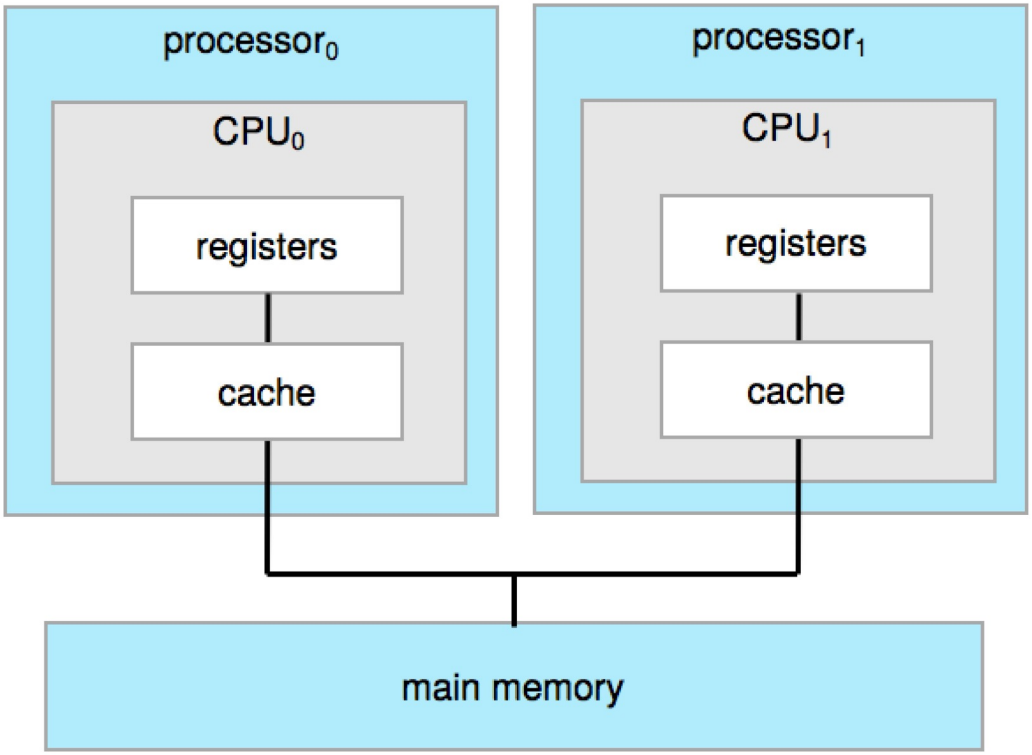


Computer System Architecture

- **Single-Processor Systems**
  - Most computer systems used a single processor containing only one CPU with a single processing core (Many years ago, not any more)
- **Multiprocessor Systems**
  - Now dominates the landscape of computing
  - Also known as parallel systems, tightly-coupled systems
  - Advantages:
    - Increased throughput
    - Economy of scale
    - Increased reliability
  - Two types:
    - **Symmetric Multiprocessing** (**SMP**) – each processor performs all tasks
    - **Asymmetric Multiprocessing** – each processor is assigned a specific task
- **Clustered Systems**
  - Like multiprocessor systems, but multiple systems working together via **networking**

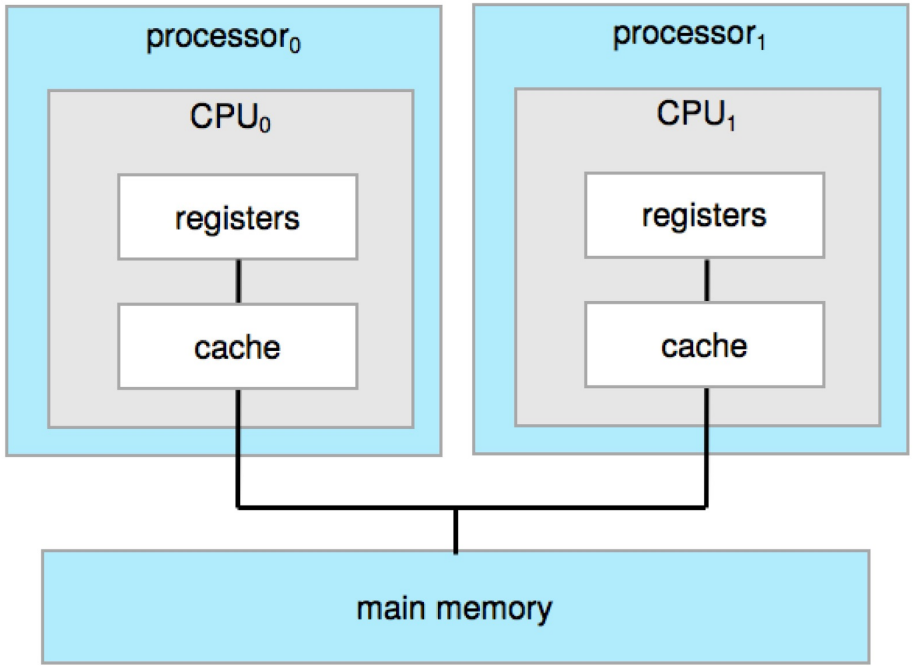
Multiprocessor Systems

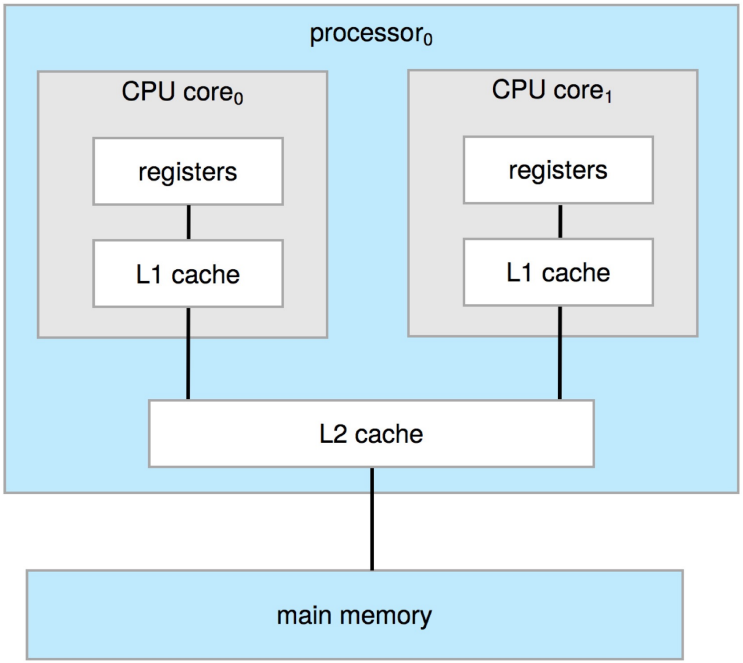
- Symmetric Multiprocessing (SMP, 对称多处理器)
  - SMP is the most common architecture
  - Each peer CPU processor has its own set of registers
  - All processors share physical memory over the system bus
  - Existence of multiple processors is transparent to the user



Multiprocessor Systems

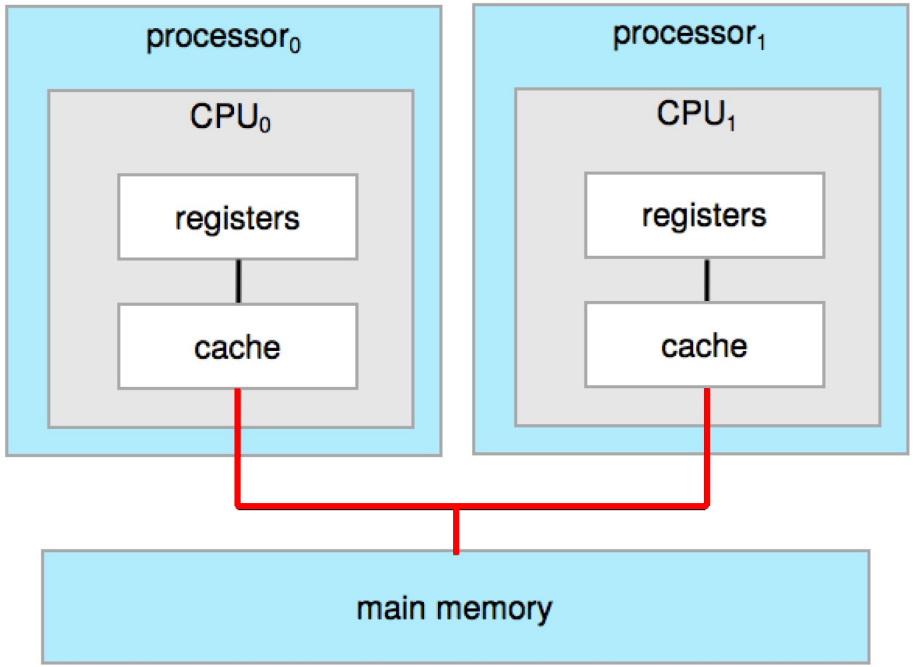
- The definition of multiprocessor now includes **Multicore Systems**, in which multiple computing cores reside on a single chip.
- Multicore on a single chip can be more efficient than traditional multiprocessor (multiple chips each with a single core).
  - On-chip communication is faster
  - Less power is required. This is important for mobile devices.

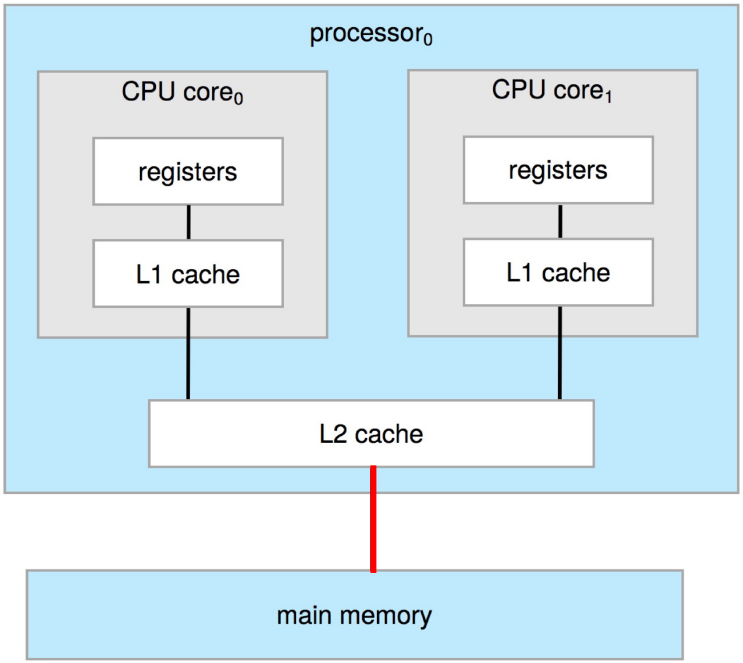




Multicore Systems

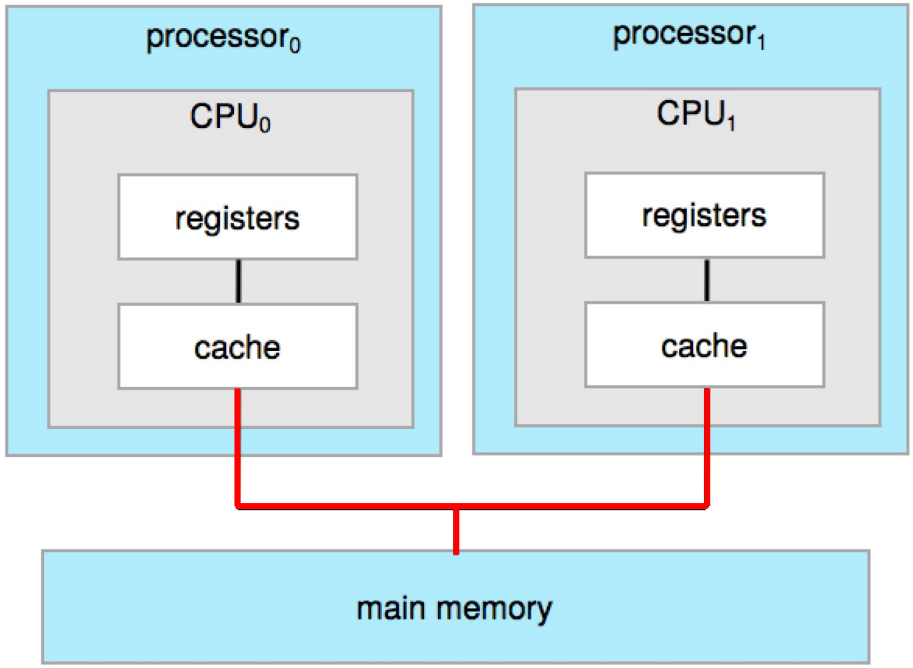
- The definition of multiprocessor now includes **Multicore Systems**, in which multiple computing cores reside on a single chip.
- Multicore on a single chip can be more efficient than traditional multiprocessor (multiple chips each with a single core).
  - On-chip communication is faster
  - Less power is required. This is important for mobile devices.

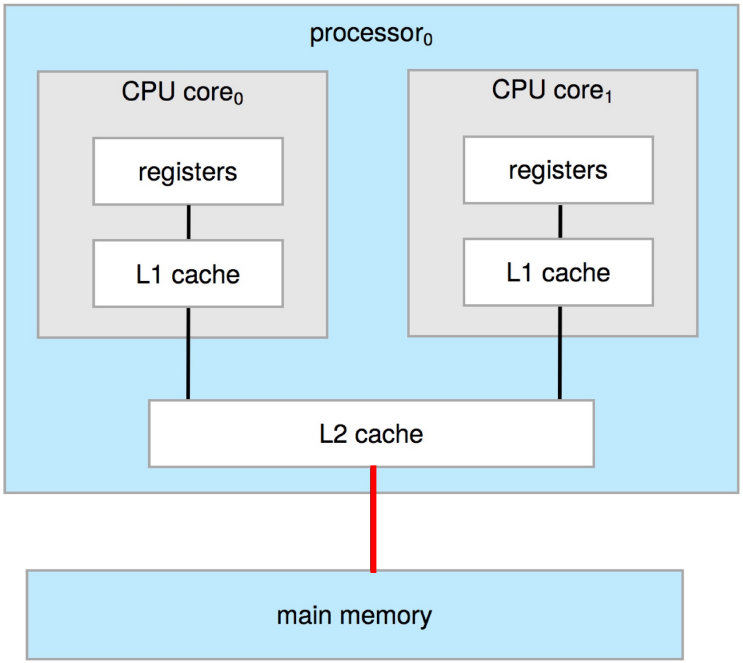




Multicore Systems

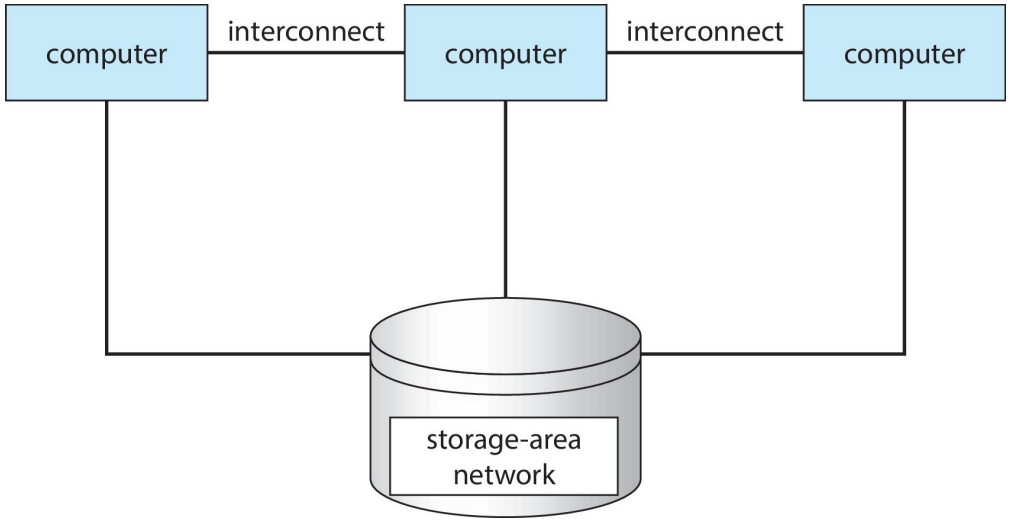
- A dual-core design with two cores on the same chip.
- Each core has its own register set and its own local cache (L1).
- L2 cache is local to the chip but is shared by the two cores
  - Low-level local caches are generally smaller and faster
  - A multicore processor with N cores appears to the OS as N normal CPUs.





Clustered Systems

- Like multiprocessor systems, but multiple systems working together via networking
  - Usually sharing storage via a Storage-Area-Network (**SAN**)
  - Some clusters are for high-performance computing (**HPC**, e.g., TH2)
    - Applications must be written to use parallelization
  - Some have distributed lock manager (**DLM**) to avoid conflicting operations



Computing Environments

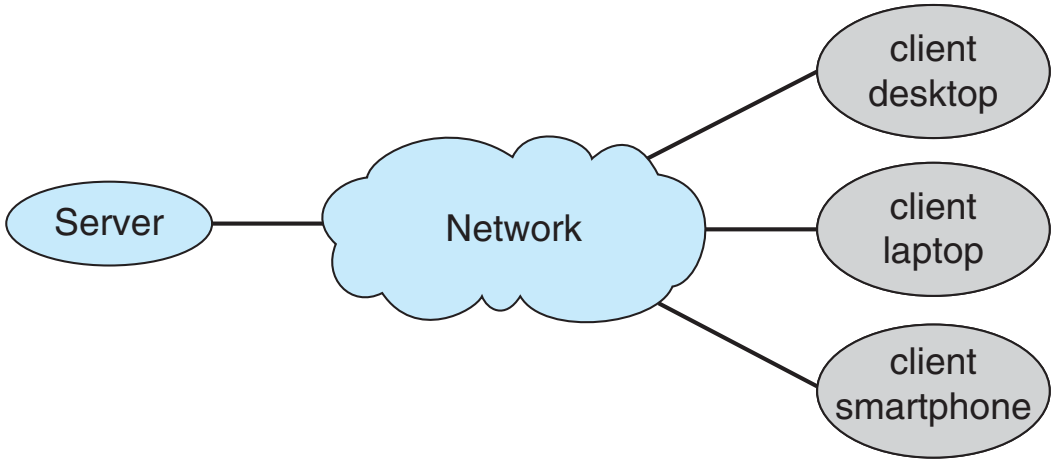
- **How OSes are used in different computing environments?**
  - Traditional Computing
  - Mobile Computing
  - Client-Server Computing
  - Peer-to-Peer Computing
  - Cloud Computing
  - Real-Time Embedded Systems

- **Traditional Computing**
  - Stand-alone general-purpose machines, e.g., desktops, laptops, netbooks
  - As computing has matured, the line separating many of the traditional computing environments have blurred.
  - Portals provide web access to internal systems
  - Network computers (thin clients) are like Web Terminals
  - Mobile computers (laptops) interconnected via wireless networks
  - Networking becoming ubiquitous – even home systems use firewalls to protect home computers from Internet attacks

- **Mobile Computing**
  - Handheld smartphones, tablets, etc.
  - Functional difference between them and a "traditional" laptop?
  - Extra feature – more OS features (GPS, sensors)
  - Allow new types of apps like *augmented reality*
  - Use IEEE 802.11 wireless, or cellular data networks for connectivity
  - Leaders are Apple iOS and Google Android.
  - Question: Is **Apple Vision Pro** considered mobile computing device?

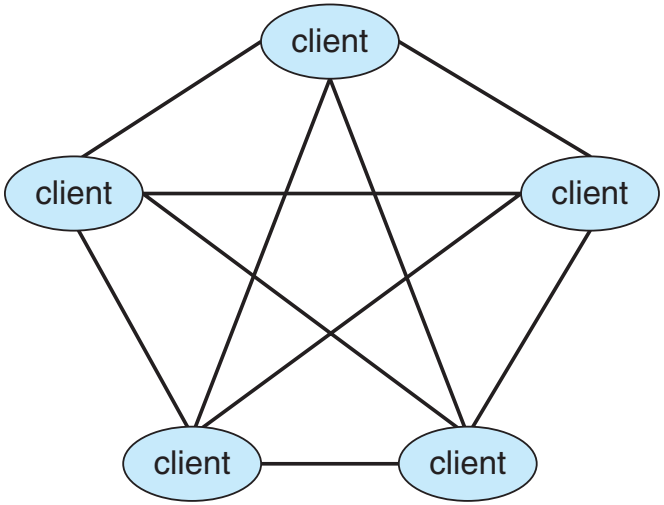
Client Server Computing

- servers responding to requests generated by clients
  - Compute-server system provides an interface to client to request services (i.e., database)
    - code-server, adopted by our lab sessions, is one kind of compute-server system
  - File-server provides interface for clients to store and retrieve files
    - Dropbox, OneDrive, static web servers, etc.



Peer-to-Peer Computing

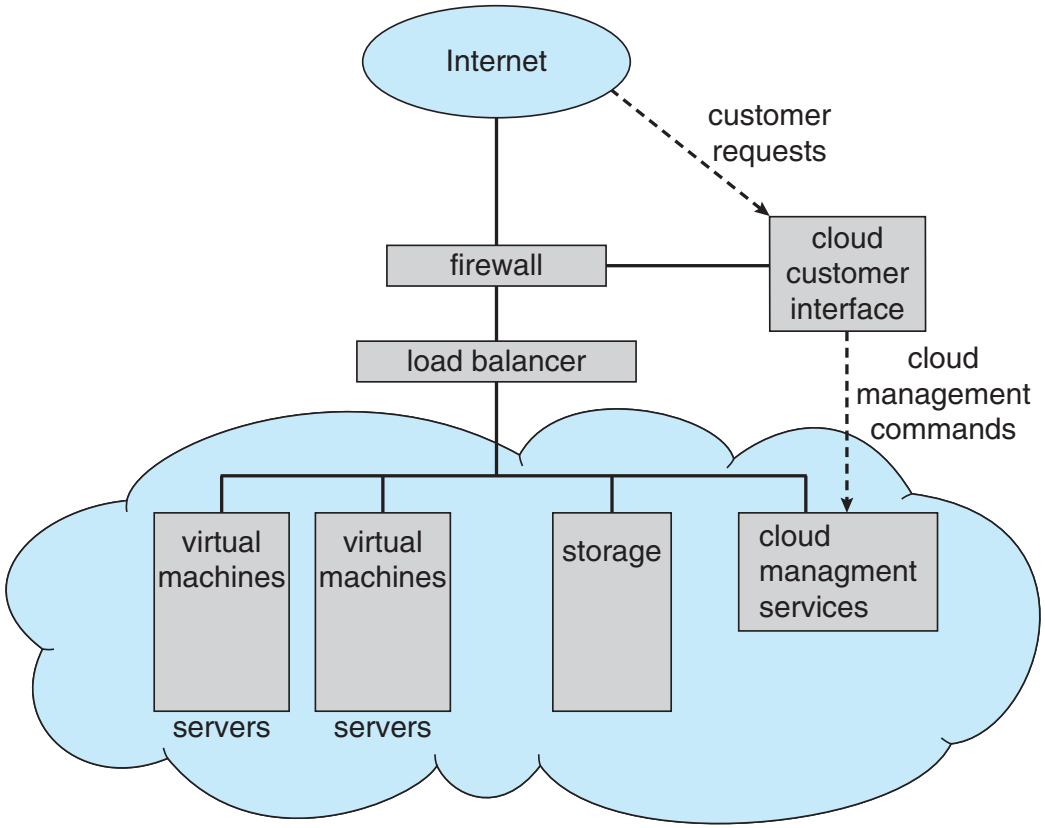
- Another model of distributed systems
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server, or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via discovery protocol
  - Example: Bit-torrent



- **Cloud Computing**
  - Delivers computing, storage, and even applications as a service across a network
  - Types:
    - Infrastructure as a service (**IaaS**)
      - servers or storage available over the Internet
      - Examples: AWS EC2, Linode, DigitalOcean Droplets
    - Platform as a service (**PaaS**)
      - a software stack ready for application use over the Internet
      - Examples: AWS Elastic Beanstalk, Google App Engine, Heroku
    - Software as a service (**SaaS**)
      - one or more applications available via the Internet
      - Examples: Slack, Docusign, and most other software companies

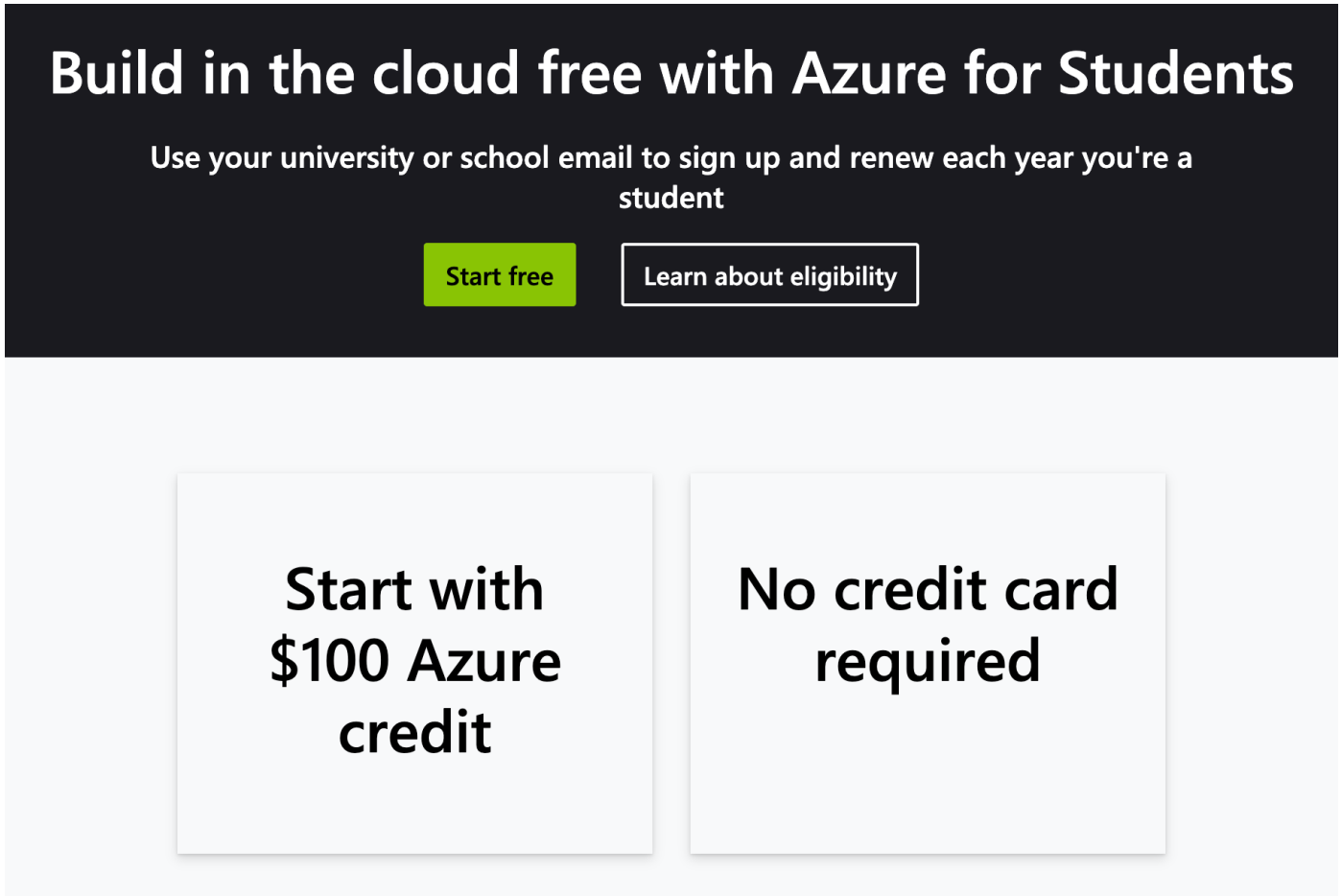
Cloud Computing

- Cloud computing environments composed of traditional OSes, plus VMMs, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



Cloud Computing

- Most Cloud providers offer FREE accounts for students. USE IT!
- Apply with your @mail2.sysu.edu.cn address.
  - https://azure.microsoft.com/en-us/free/students



Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
  - E.g., car engines, manufacturing robots, microwave ovens
  - Vary considerable, special purpose, limited purpose OS, real-time OS
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS, e.g., ASICs
- Real-time OS has well-defined fixed time constraints
  - Processing **must** be done within constraint
  - Correct operation only if constraints met
  - The entire system might breakdown if the system does not complete a task in a certain time frame. For example, switches and routers mostly run on RTOS to manage packet delivery with minimal latency, if the system fails to process one packet in time, other packets will suffer significant latency. As a result, most routers choose to **DROP** the bad packets if they take too long to process.

Thank you!