Operating System Concepts

A. Silberschatz, P. B. Galvin, and G. Gagne, 10th edition

Chapter 1: Introduction

Mei-Ling Chiang, Professor

joanna@ncnu.edu.tw

Dept. of Information Management

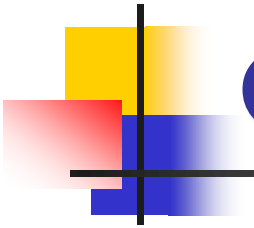
National Chi-Nan University, Taiwan, ROC

2025/09/18

Chapter 1: Introduction

Outline

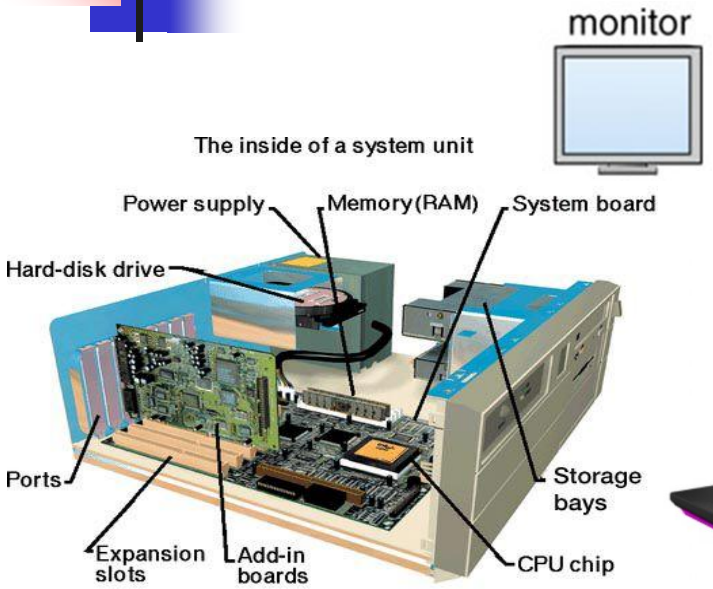
- 1.1 What Operating Systems Do
- 1.2 Computer-System Organization
- 1.3 Computer-System Architecture
- 1.4 Operating-System Operations
- 1.5 Resource Management
- 1.6 Security and Protection
- 1.7 Virtualization
- 1.8 Distributed Systems
- 1.9 Kernel Data Structures
- 1.10 Computing Environments
- 1.11 Free and Open-Source Operating Systems

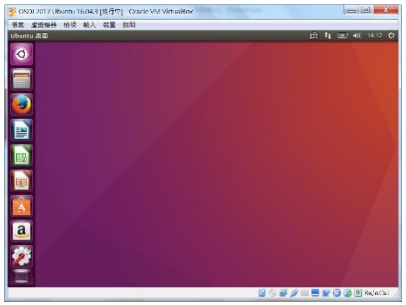


OS

- Why OS is needed?
- What is an OS?
- What does an OS do?
- When does an OS do?
- What are the types of OS?
- What are OS Services?
- How does OS work?
- What are OS components?
- What are OS structures?
- ….

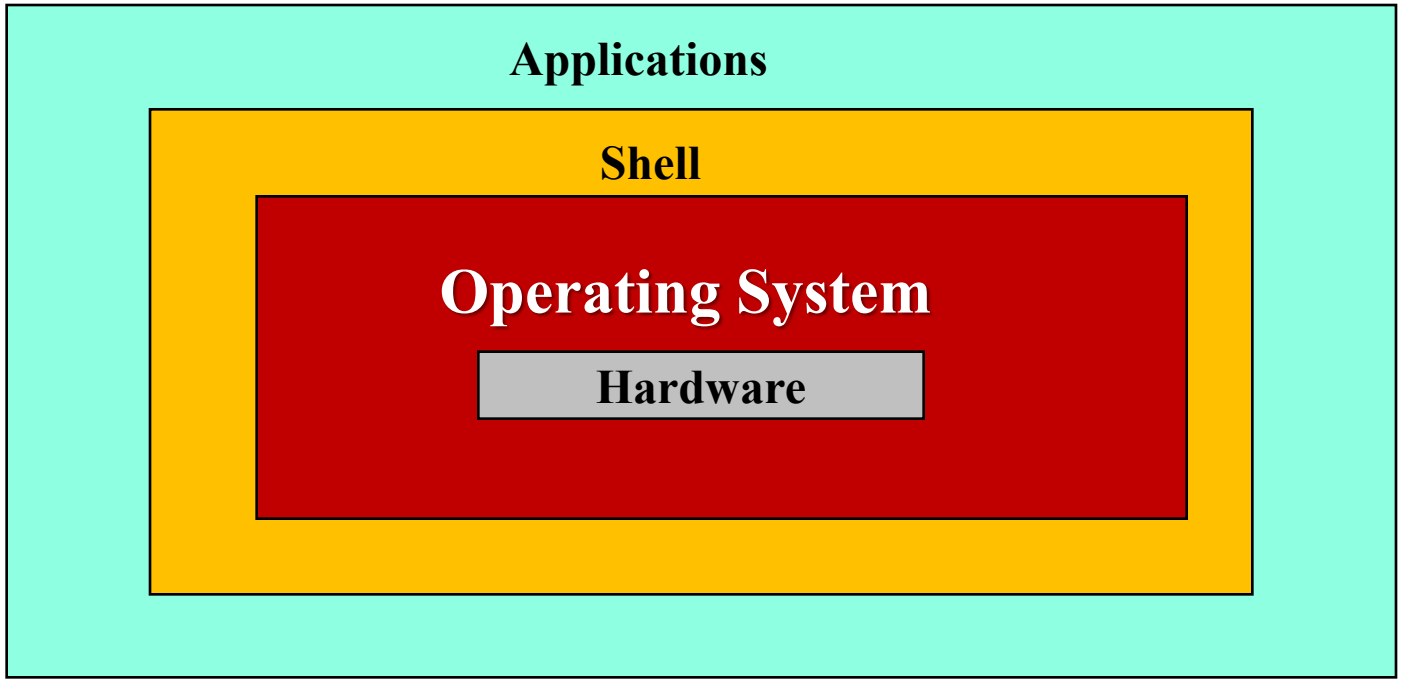
Why OS is needed?





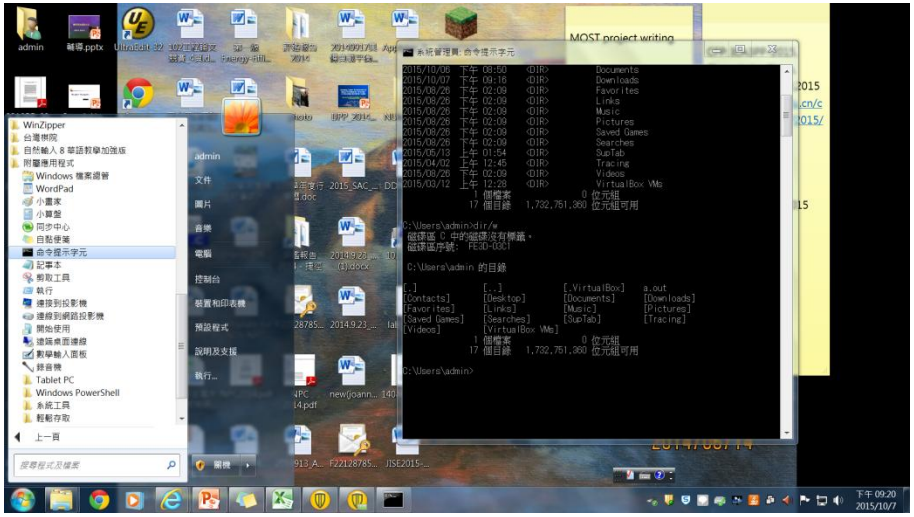
Shell: Command Interpreter

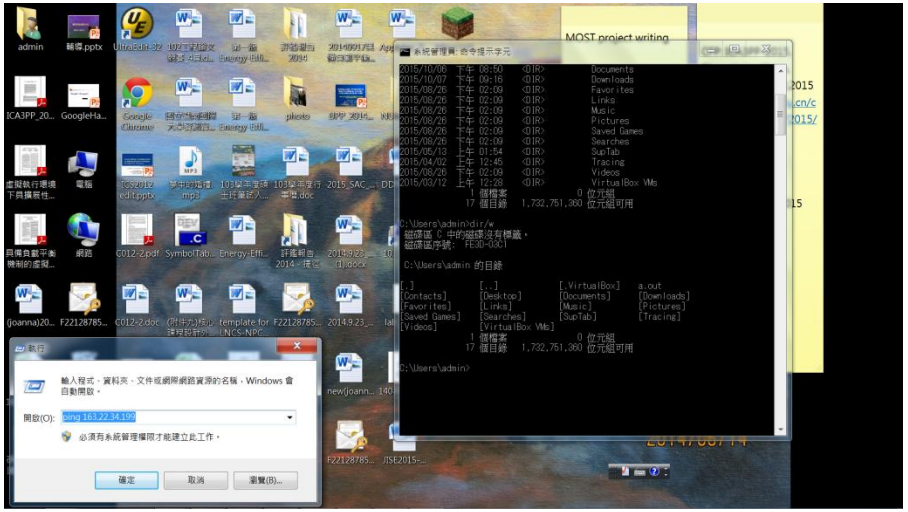
- Get and execute the next user-specified command
- GUI, CLI (Command Line Interface)




Grab File Edit Capture Window Help
Apple Computer Inc.
main 1-800-MY-APPLE
other 800-275-2273
home page http://www.apple.com
work 1 Infinite Loop
C.part no DA 9041
United States
operating system
op·er·at·ing sys·tem
the software that supports a computer's basic functions, such as scheduling tasks, executing applications, and controlling peripherals.

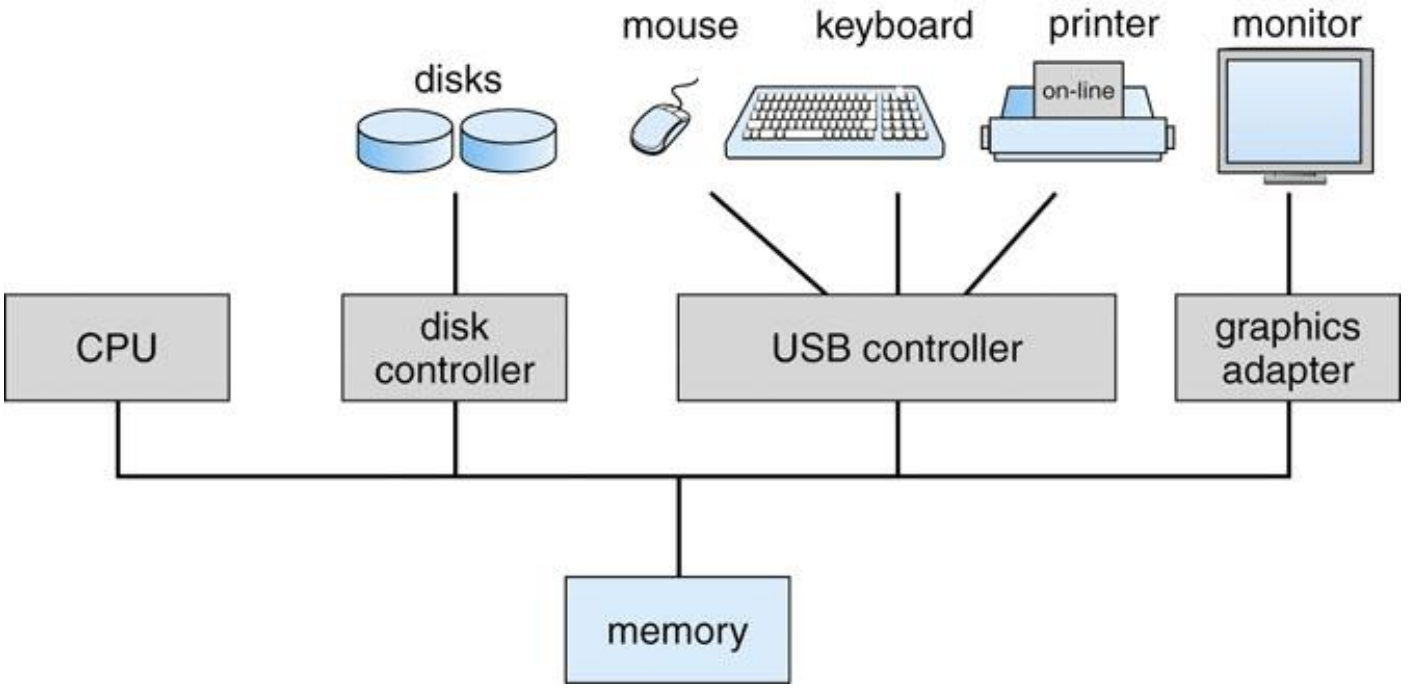

Verizon
10:31
42%
App Store
Settings
Clock
Contacts
Fahrenheit
Maps
Reminders
Notes
Calculator Pro
Google
Pocket
Kindle
Wunderlist
Evernote
Travel
Reference
Apple2
Entertainment
Music
Social
Utilities
Apple
News W S
Safari
Calendar
Mail


MOST project writing


MOST project writing

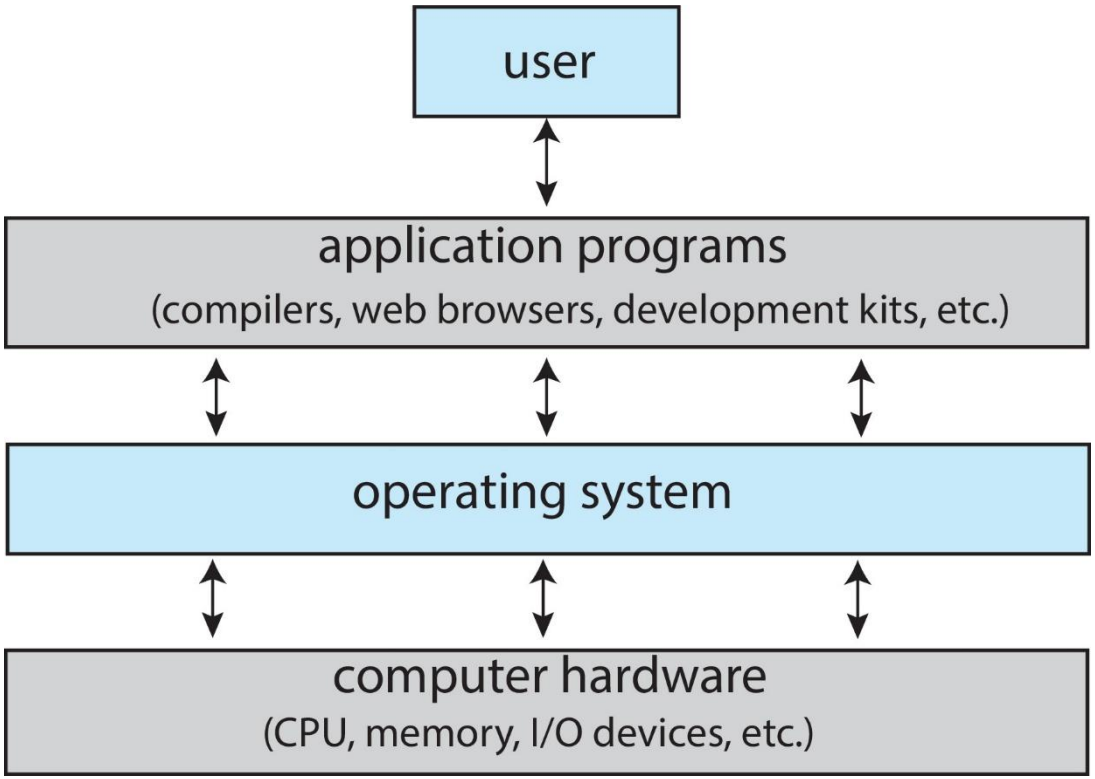
Computer System Organization

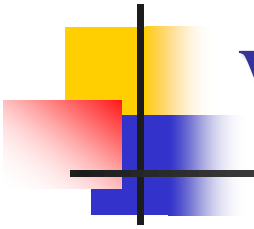
- **A modern computer system**
  - consists of one or more CPUs and devices controllers connected through a common bus that provides access to shared memory



Abstract View of Components of a Computer System

- **A computer system can be divided into 4 components:**





What is an OS?

- ***Operating System***
  - software that manages the computer hardware
  - a program that acts as an intermediary between the computer user and the computer hardware
- **Purpose**
  - provide an *environment* in which a user can execute programs in a *convenient* and *efficient* manner

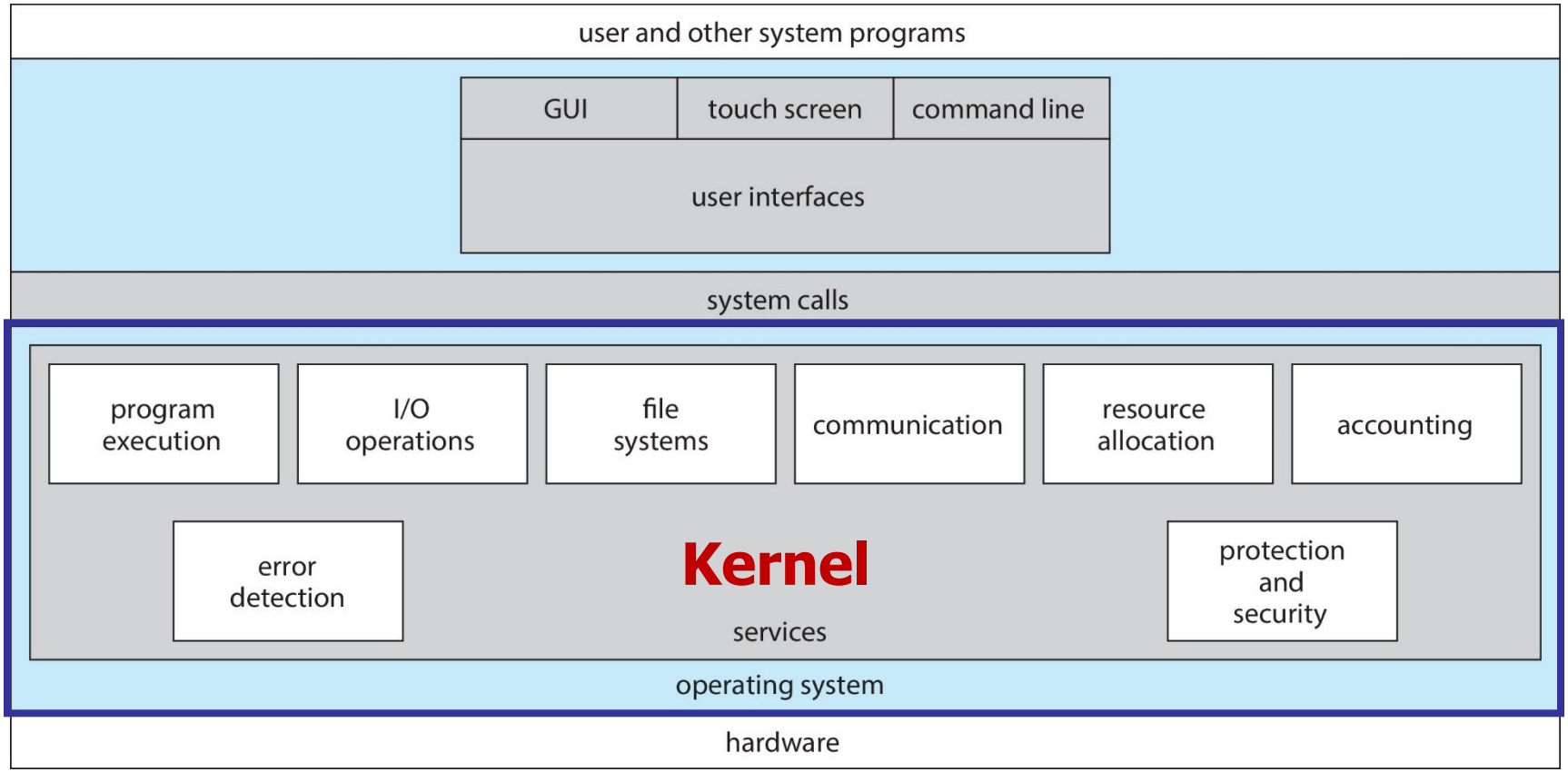
What is an OS? (Cont.)

- ***Resource allocator*** **– manages all resources**
  - Decides between conflicting requests for efficient and fair resource use
- ***Control program*** **– manages the execution of user programs to prevent errors and improper use of the computer**

Operating System Definition

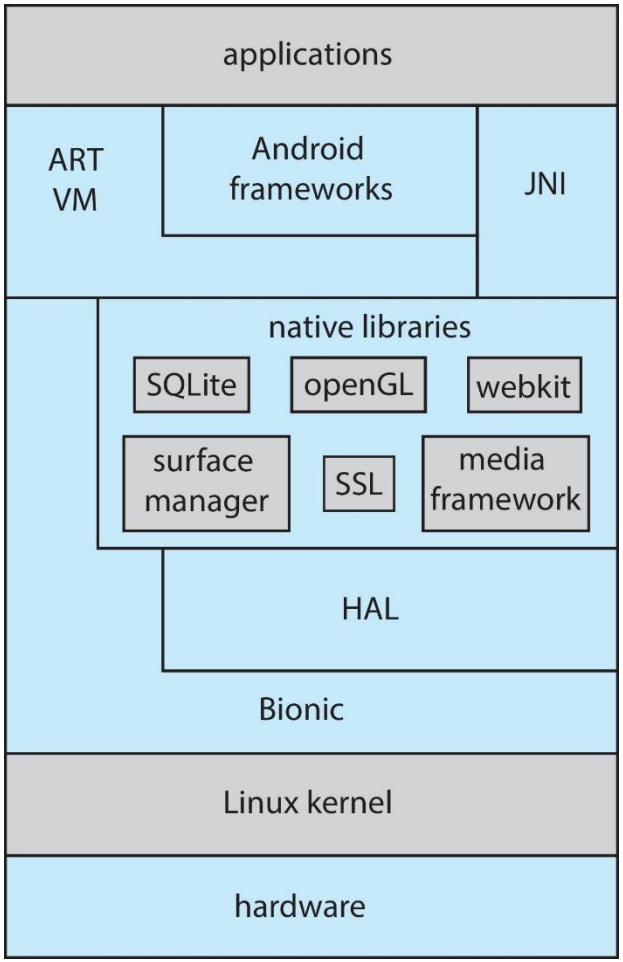
- **No universally accepted definition**
- **A simple viewpoint is that it includes everything a vendor ships when you order an OS.**
  - But varies greatly across systems
- **A more common definition:**
  - OS is the one program running at all times on the computer -- usually called the **kernel**.
  - Along with the kernel, there are two other types of programs:
    - system programs, which are associated with the OS but are not necessarily part of the kernel
    - application programs, which include all programs not associated with the operation of the system

A View of OS Services



OS Definition (Cont.)

Architecture of Android



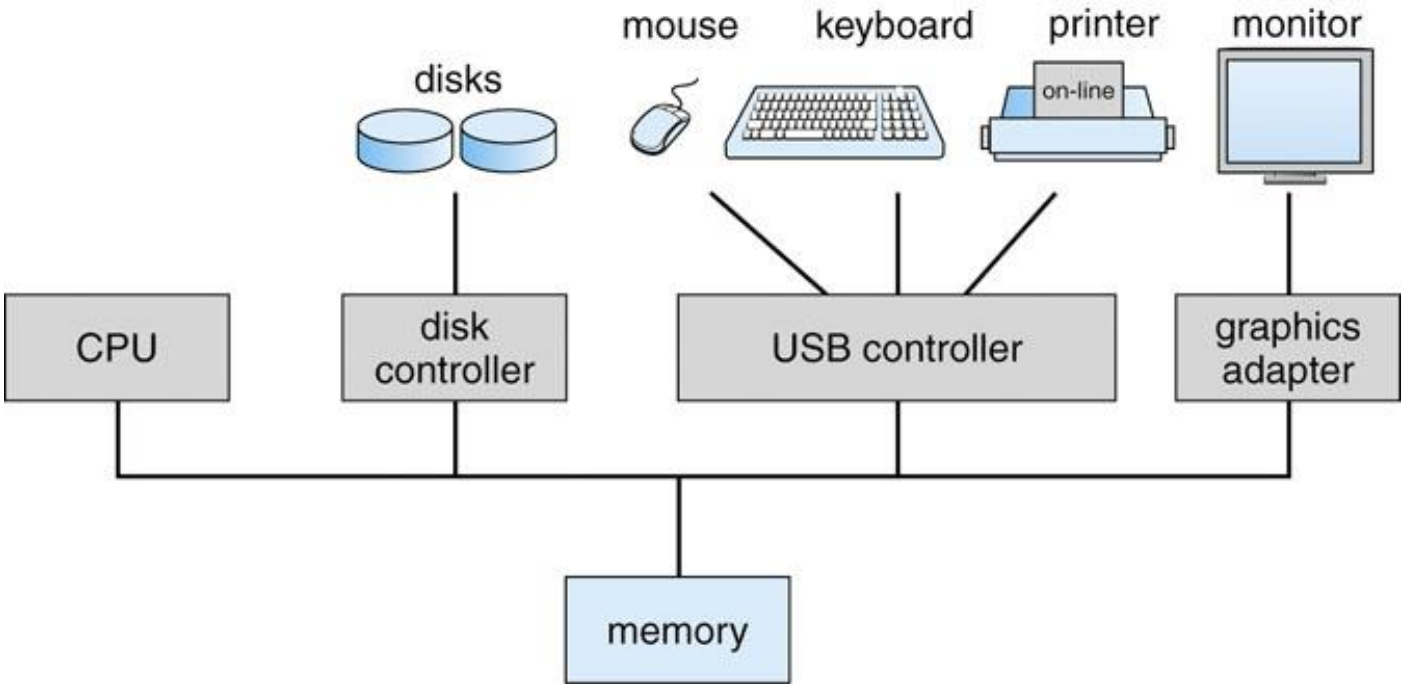
- **Today's OSes for general purpose and mobile computing**
  - includes the always running kernel, middleware frameworks that ease application development and provide features (such as databases, multimedia, graphics), and system programs that aid in managing the system while it is running.

What dos an OS do?

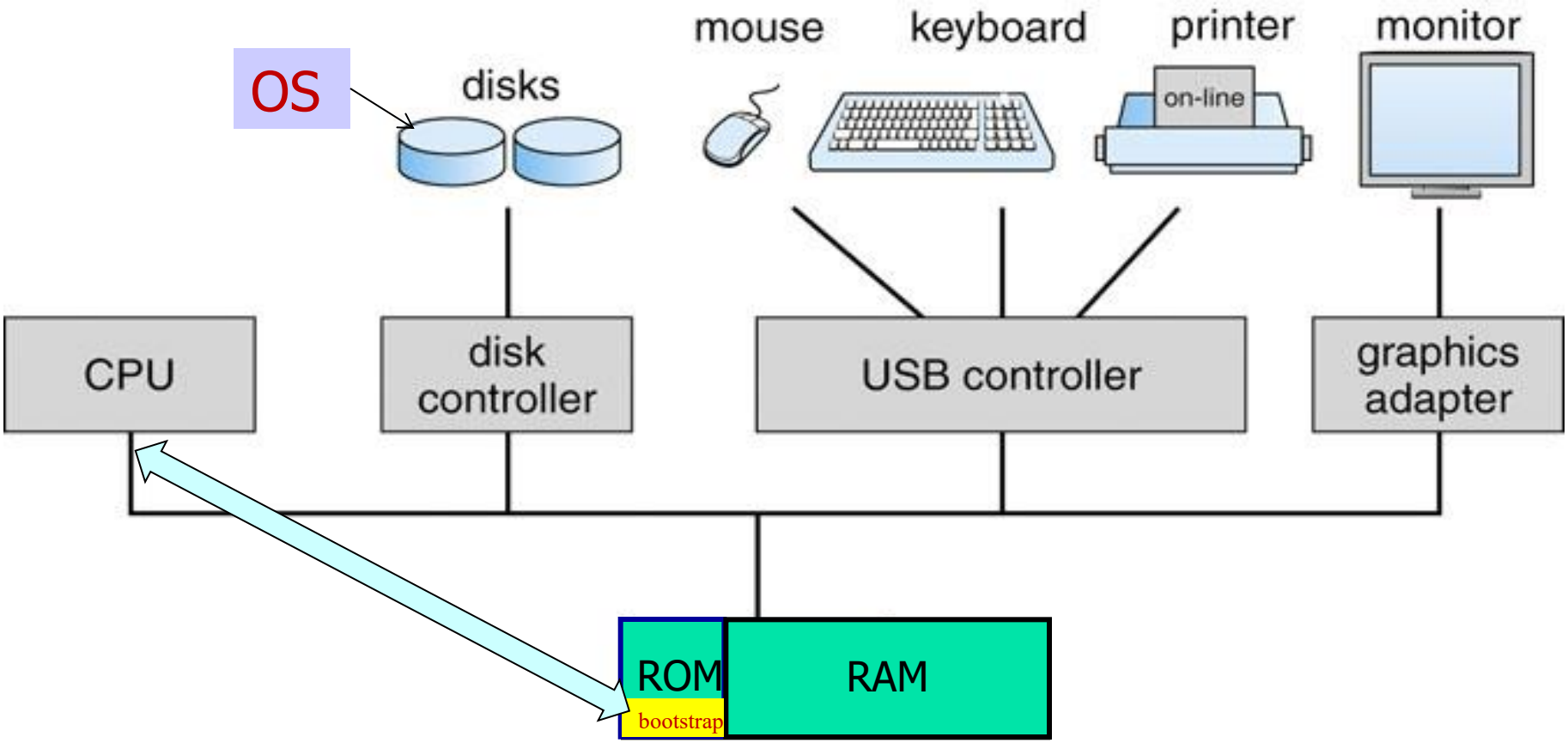
- Management
- Allocation
- Control
- Service

1.2 Computer System Organization

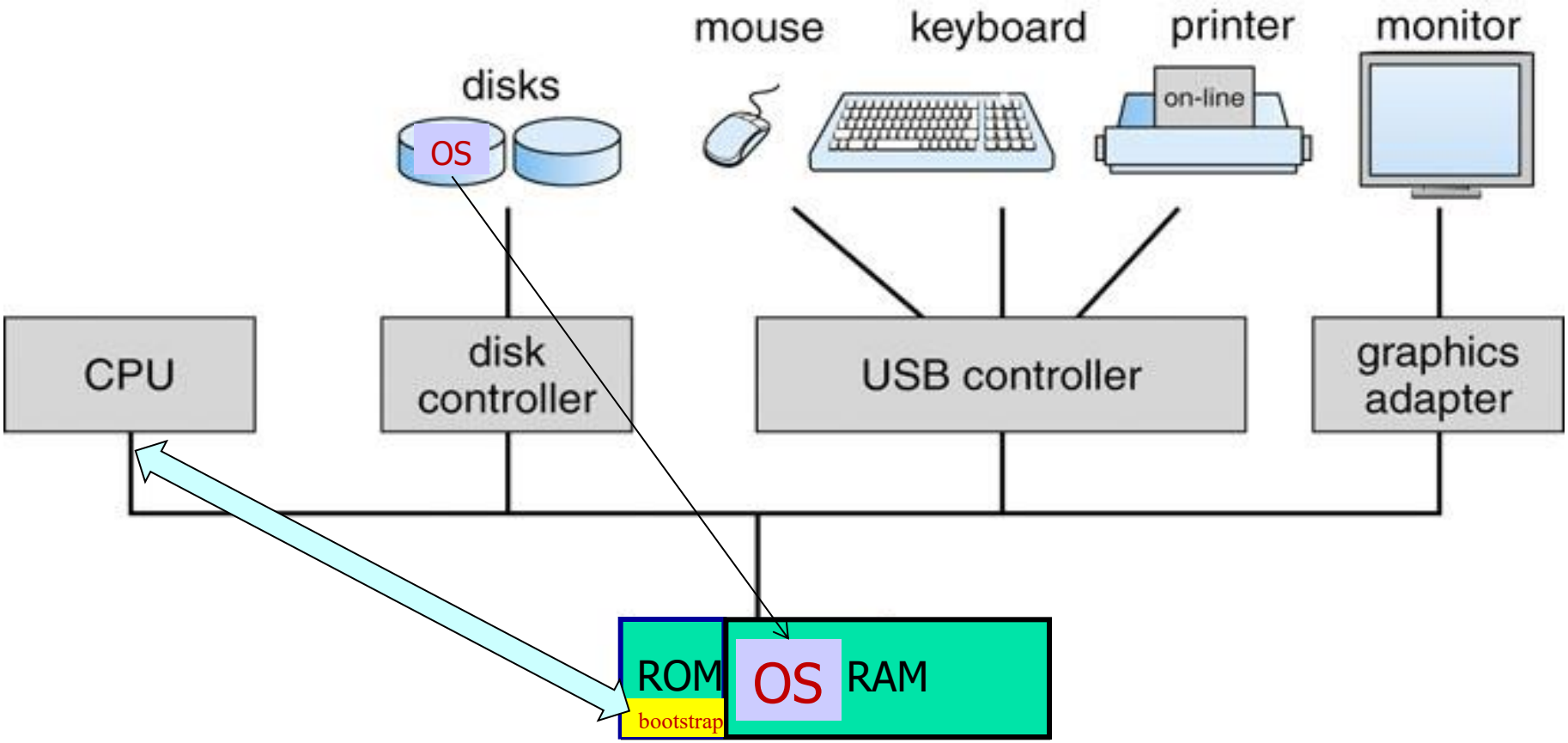
- **A modern computer system**
  - consists of one or more CPUs and devices controllers connected through a common bus that provides access between components and shared memory



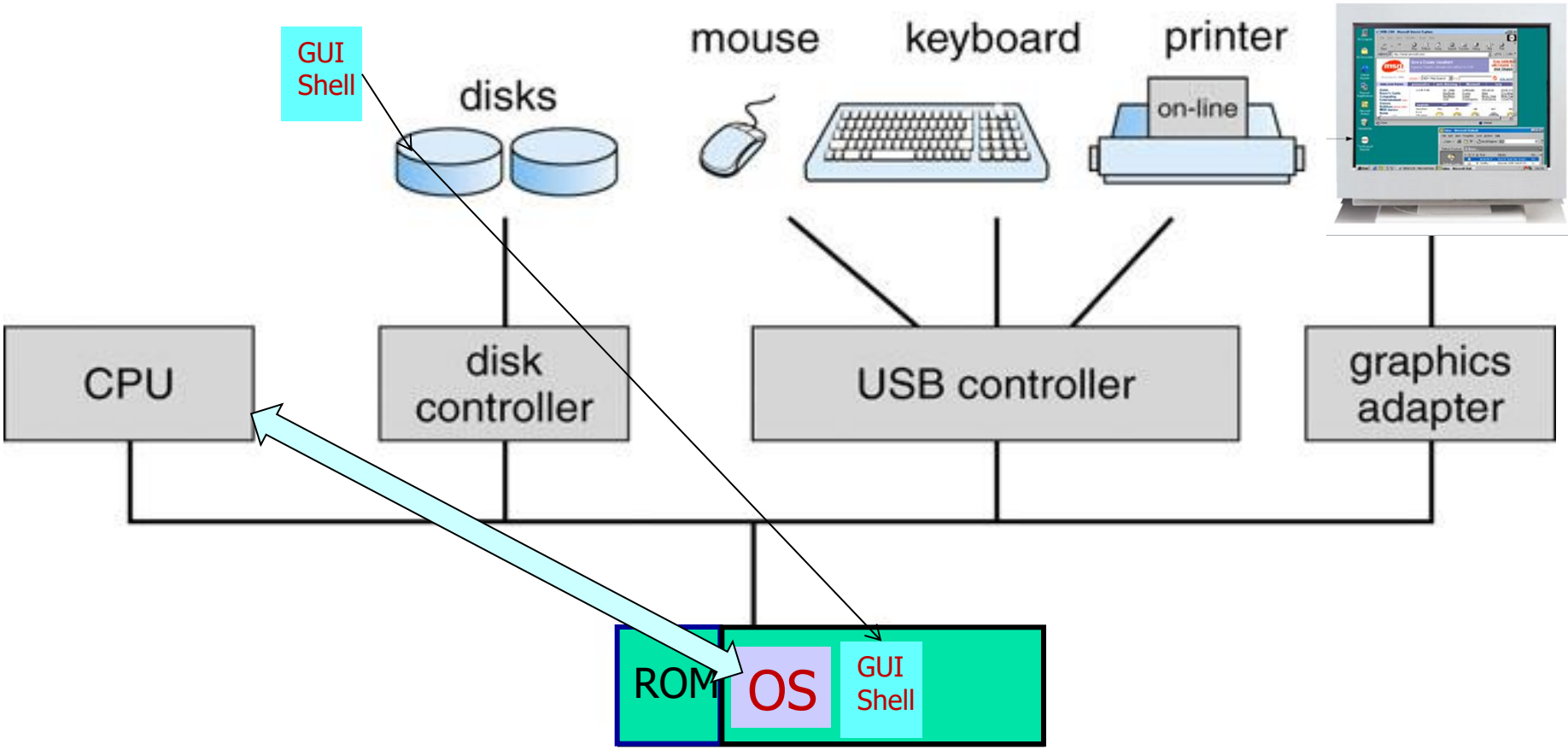
When does an OS run?
(1) system booting



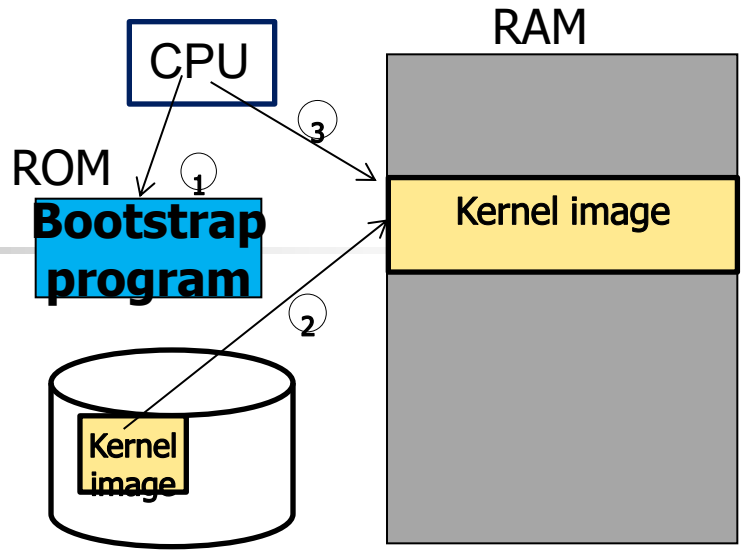
When does an OS run?
(1) system booting



When does an OS run?
(1) system booting

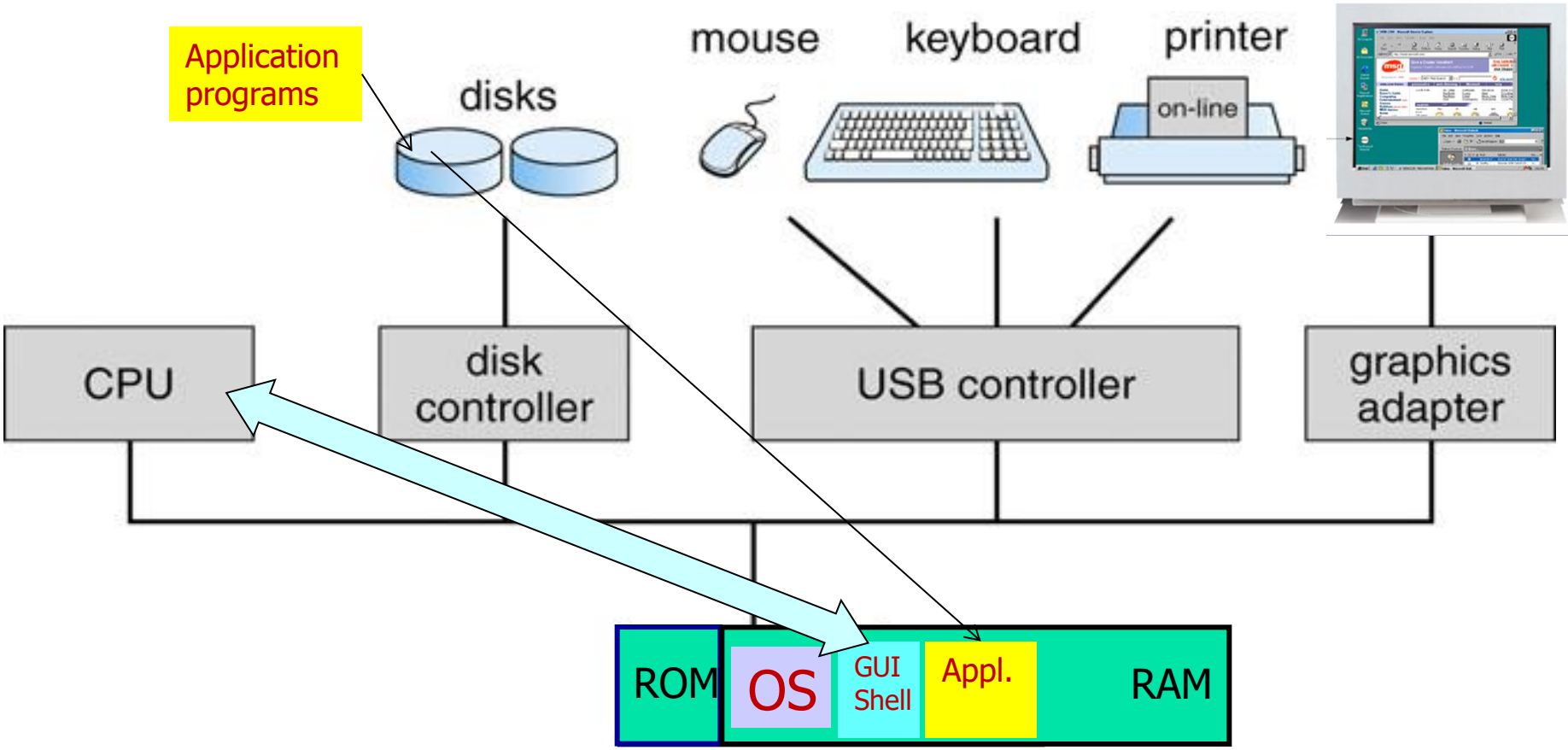


Computer Startup

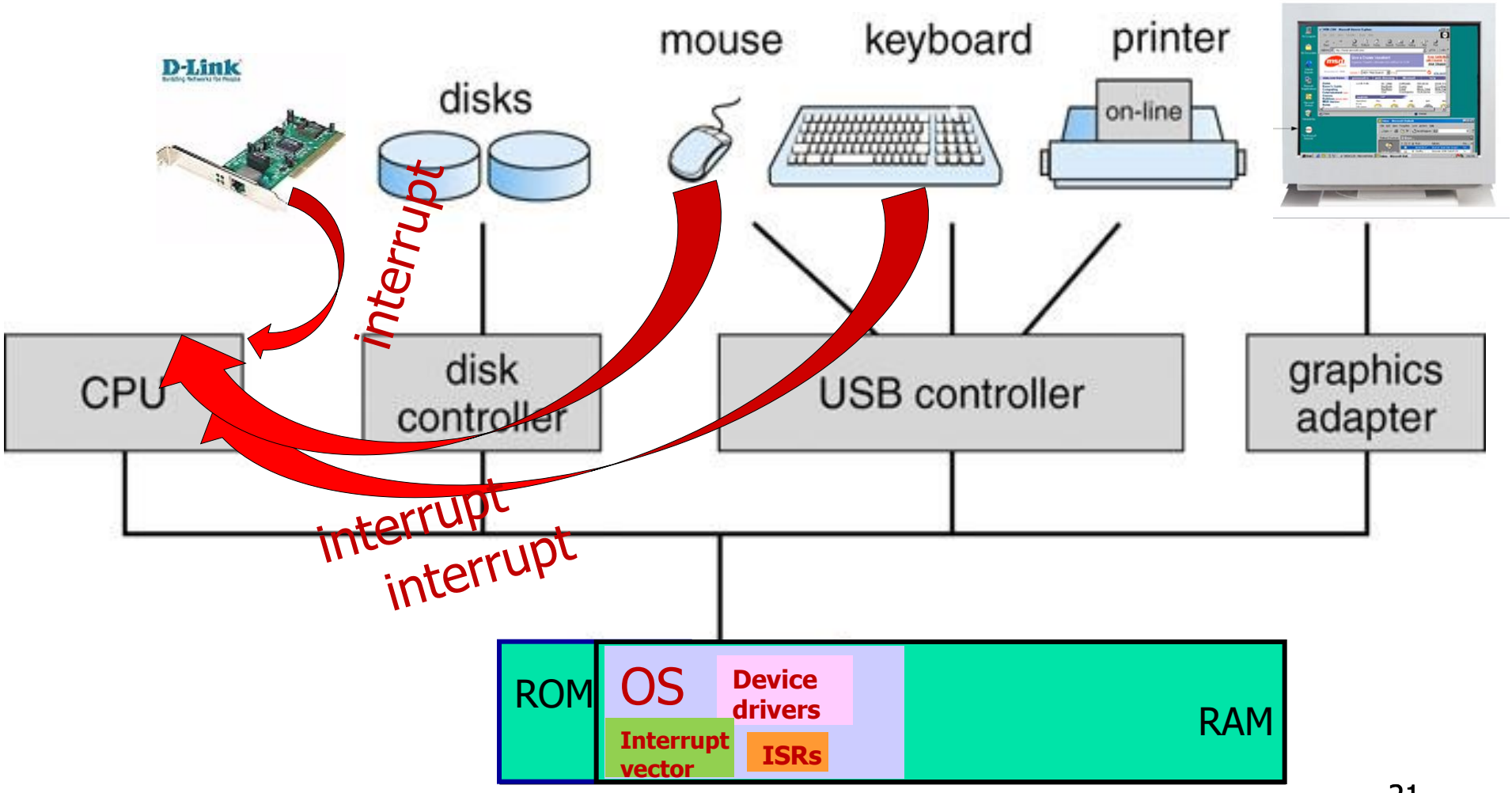


- Needs to have an initial program to run: ***bootstrap program***
  - typically stored in ROM, EEPROM, flash RAM
  - initializes all aspects of the system, loads OS kernel, start executing the OS
- OS then starts executing the first process, such as “init” and waits for some event to occur
- **Interrupts**: alert the CPU to events that require attention

When does an OS run? (2) interrupt occurs

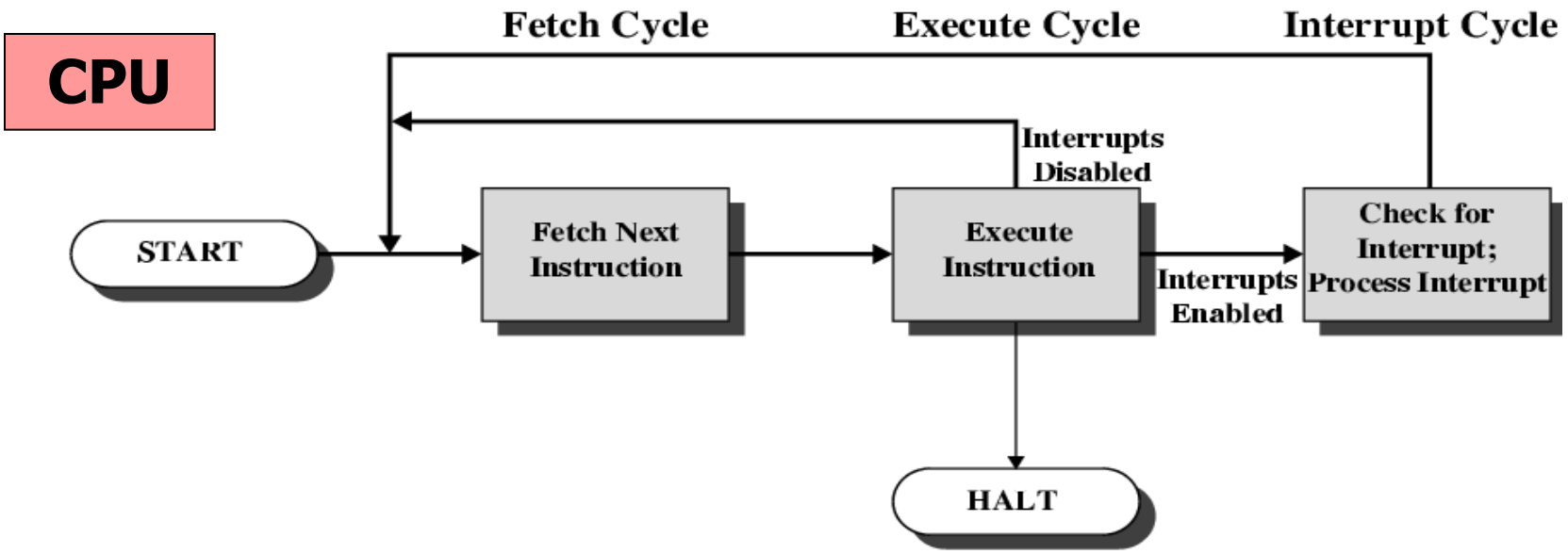


OS is interrupt-driven



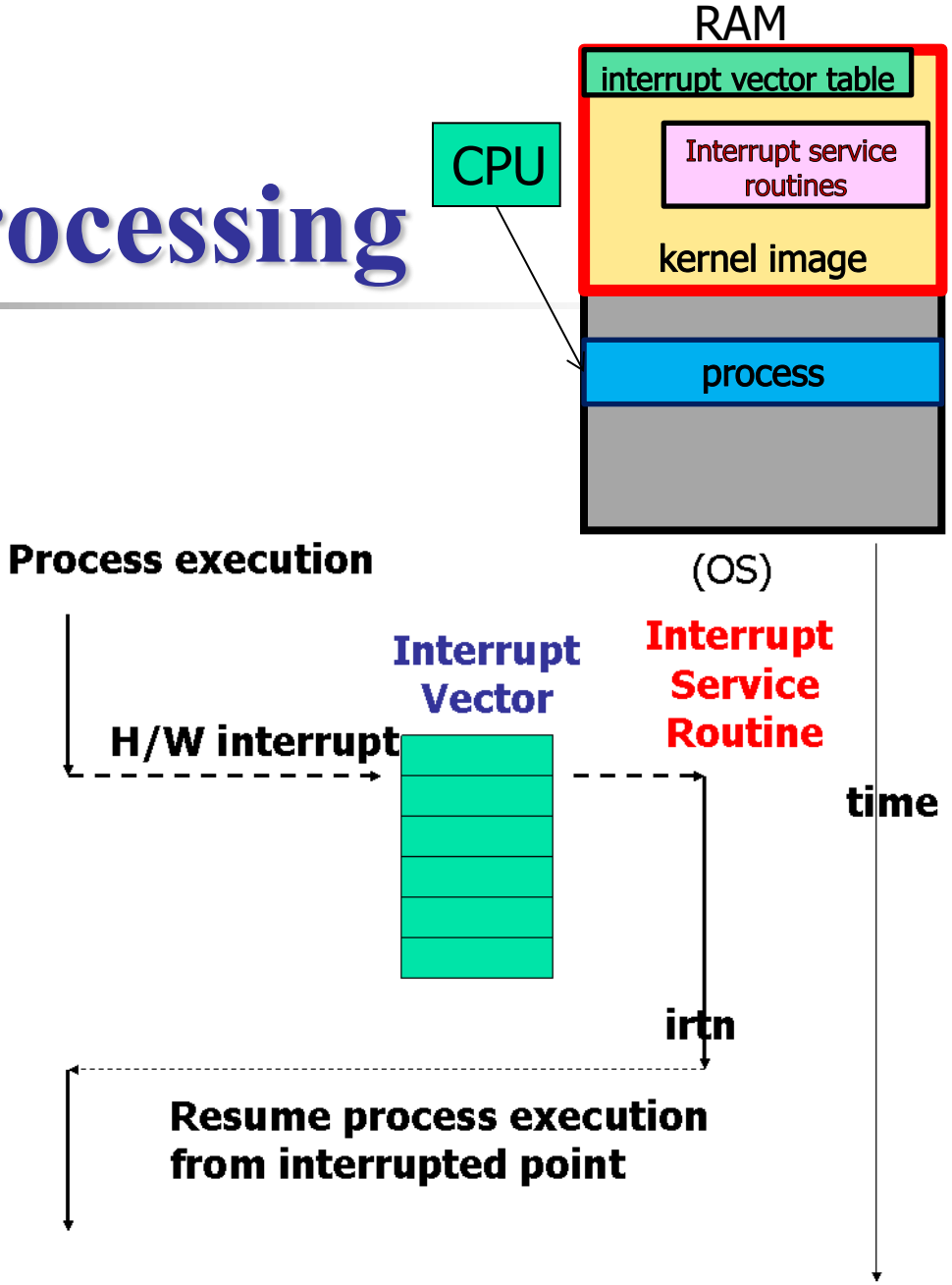
Instruction Cycle with Interrupts

- H/W may trigger an interrupt by sending a hardware signal to CPU
- S/W may trigger an interrupt by executing a system call (monitor call)

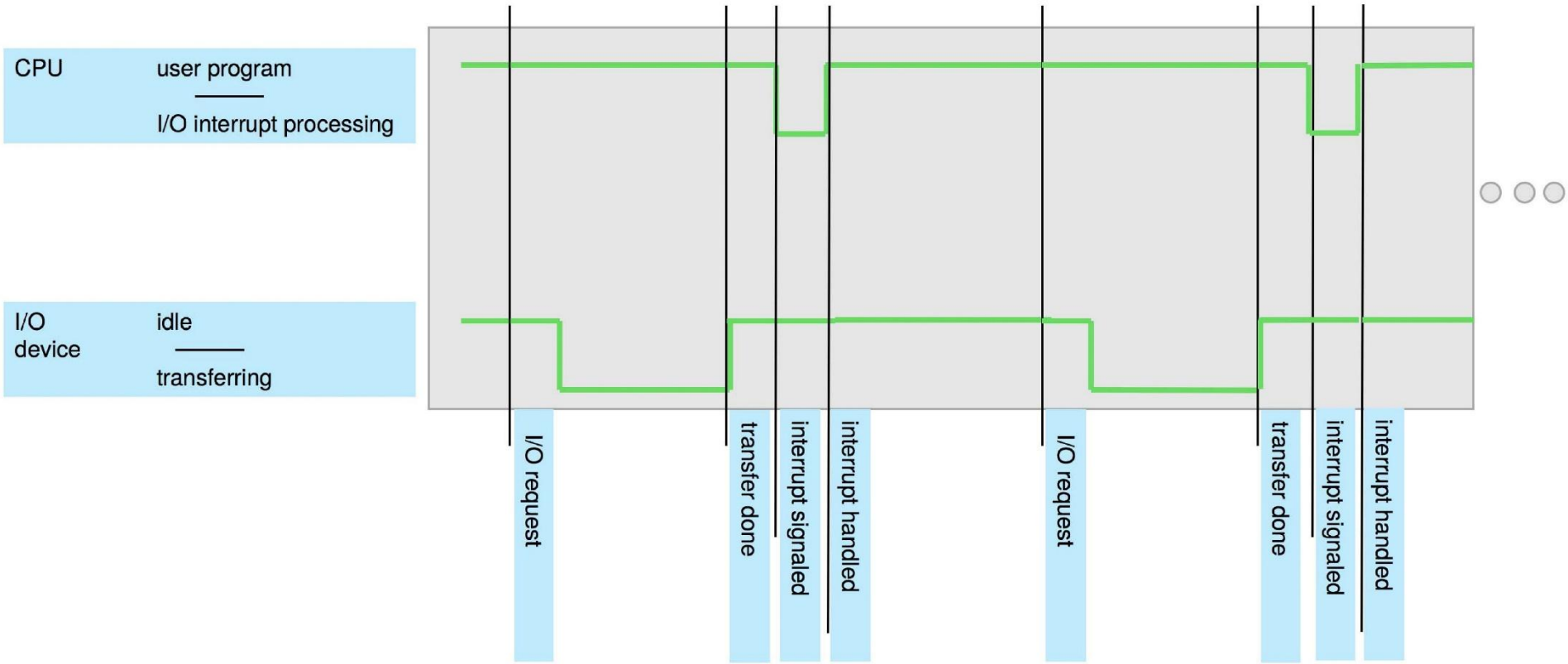


Interrupts Processing

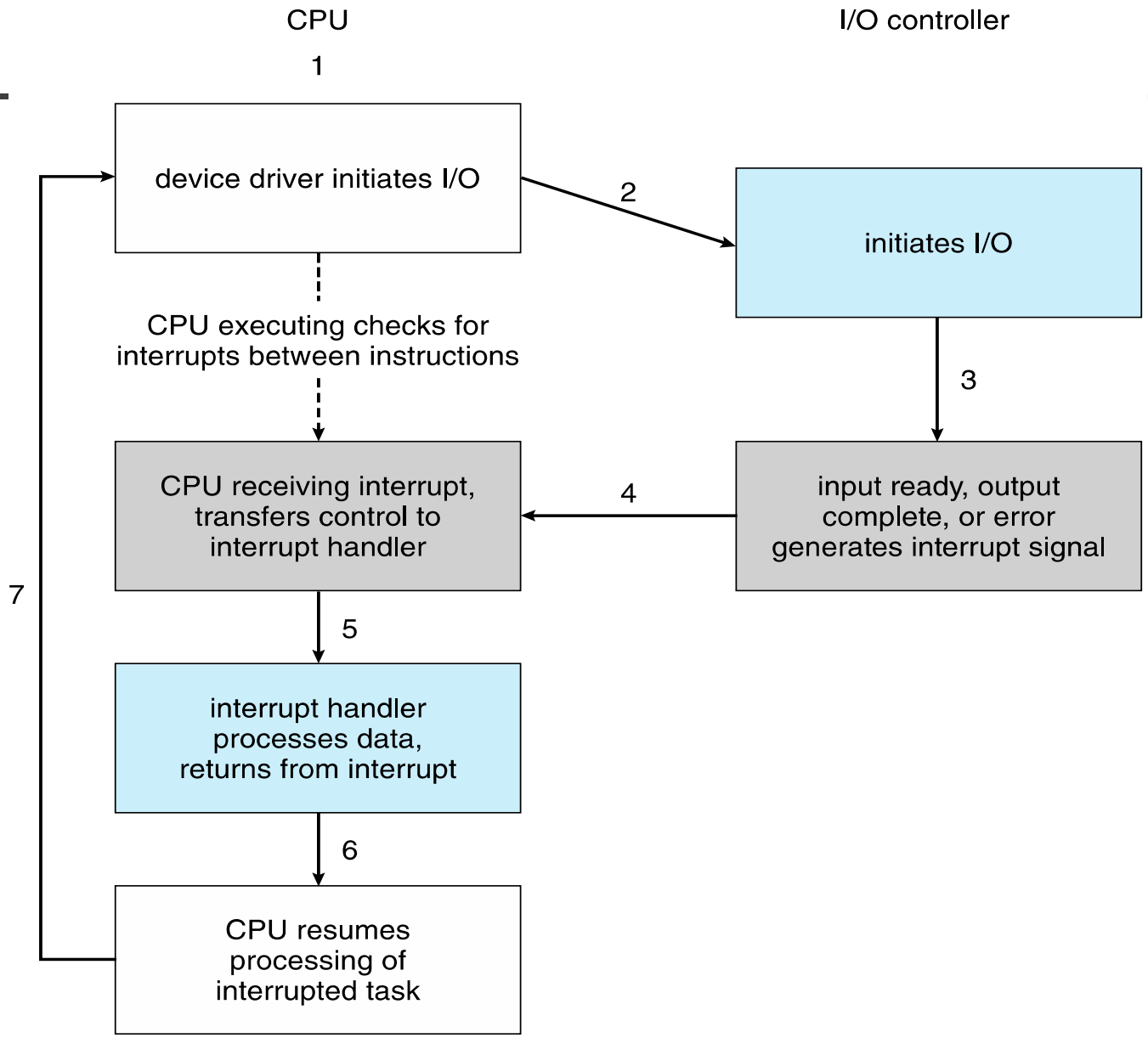
- The interrupt must transfer control to ***interrupt service routine*** (ISR) generally, through *interrupt vector*, which contains the addresses of all service routines.
- Interrupt architecture must save the address of the interrupted instruction for later resuming.
- Used in Windows, UNIX



Interrupt Timeline for a Single Program Doing Output

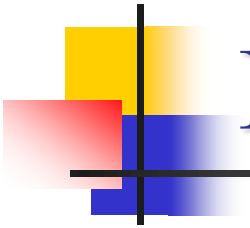


Interrupt-drive I/O Cycle



Intel Processor Event-vector Table

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

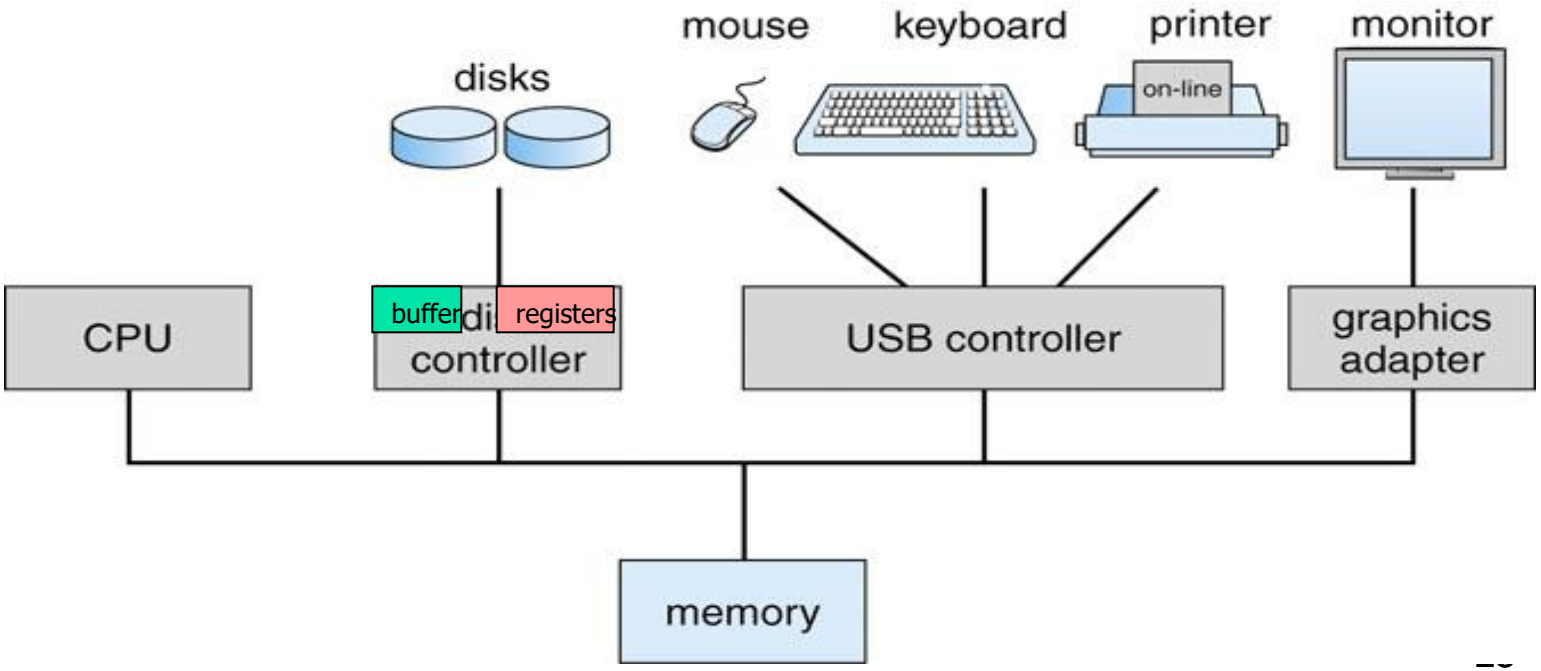


I/O

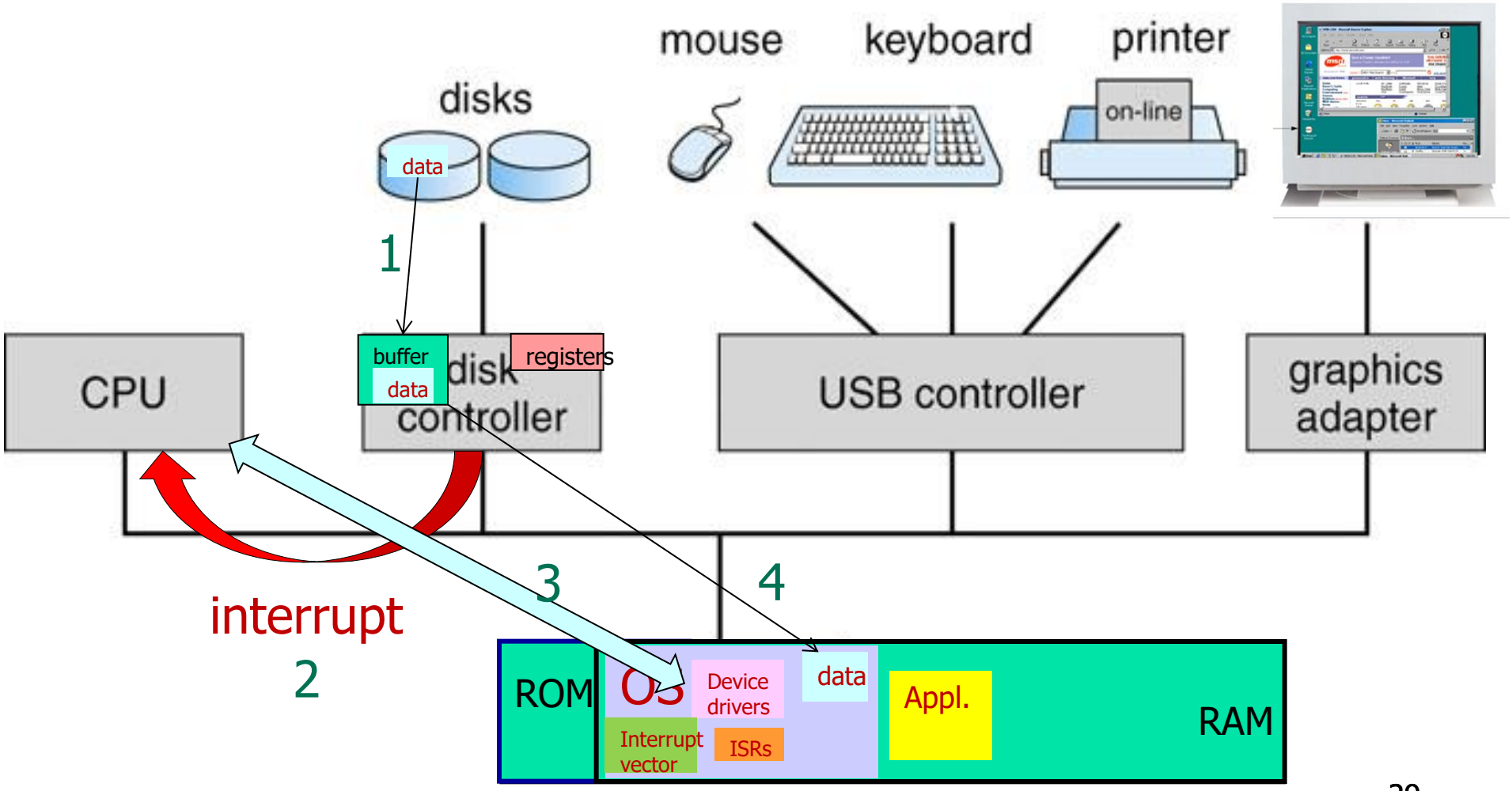
- **Interrupt-driven I/O**
  - requires active CPU intervention to transfer data between memory and an I/O module
  - CPU is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer
- **Direct Memory Access (DMA) I/O**
  - Additional Module (hardware) on system bus
  - DMA module takes over from CPU for I/O

I/O Structure

- **A device controller**
  - maintains local buffer and a set of special-purpose registers
  - responsible for moving data between peripheral devices that it controls and its local buffer
- **OS have a device driver for each device controller.**



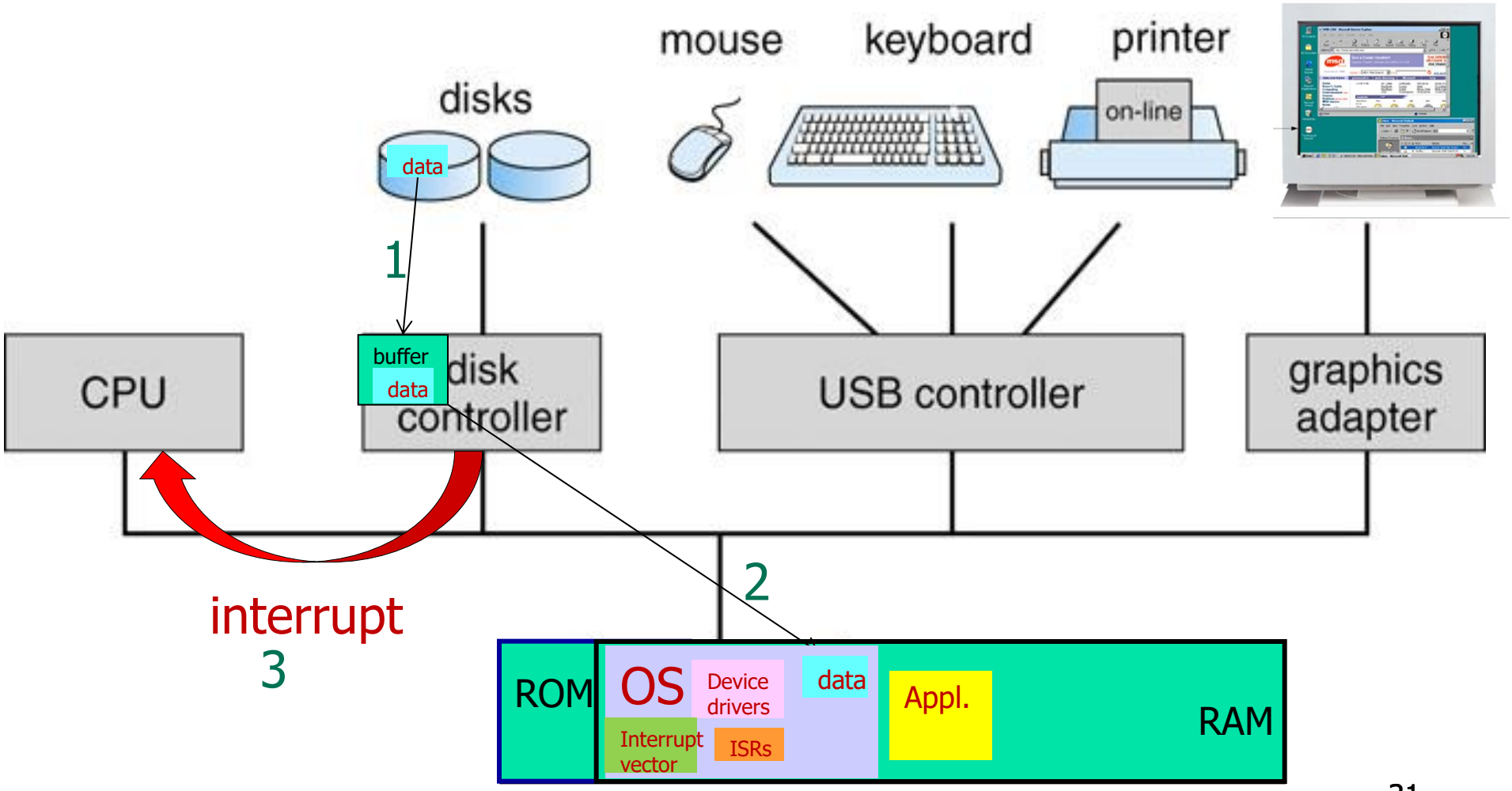
Interrupt-driven I/O

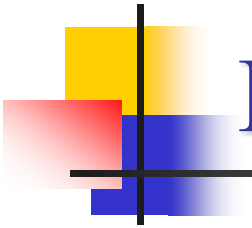


I/O Structure (Cont.)

- **To start an I/O**
  - Device driver loads the appropriate registers within device controller.
  - Device controller examines the registers and does the request I/O. If read is requested, the controller starts to transfer data from device to its local buffer.
- **When the transfer of data is complete**
  - Device controller informs CPU that it has finished its operation by causing an *interrupt*.

DMA I/O

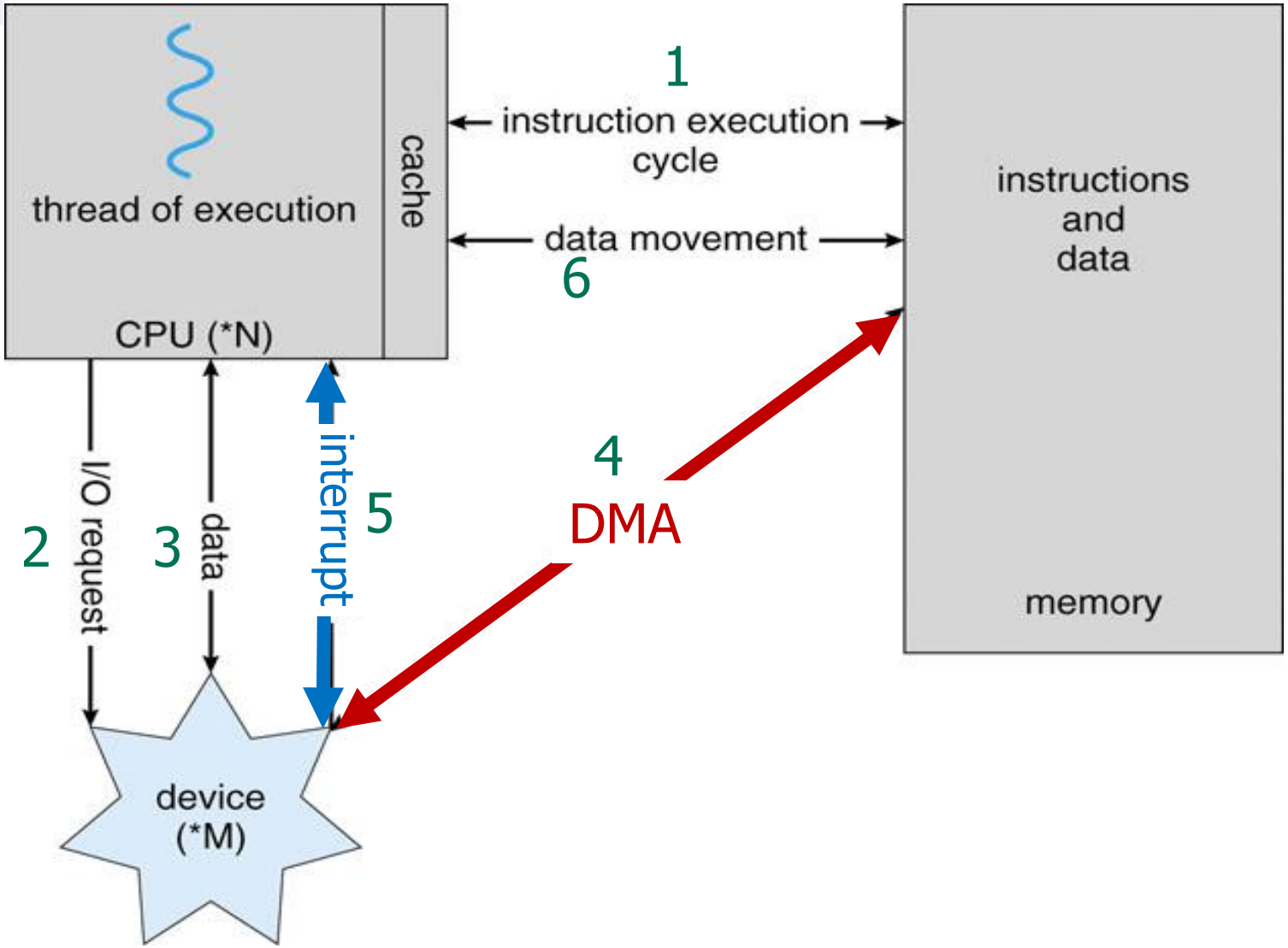

mouse
keyboard
printer
disks
data
on-line
1
buffer
data
CPU
disk controller
USB controller
graphics adapter
interrupt
3
2
ROM
OS
Device drivers
data
Appl.
Interrupt vector
ISRs
RAM



Direct Memory Access

- Device controller transfers an entire block of data directly to or from its own buffer storage to main memory, without CPU intervention.
- Only one interrupt is generated per block, rather than one interrupt per byte generated for low-speed devices.
- While device controller is performing data transfer, CPU is free to perform other tasks.

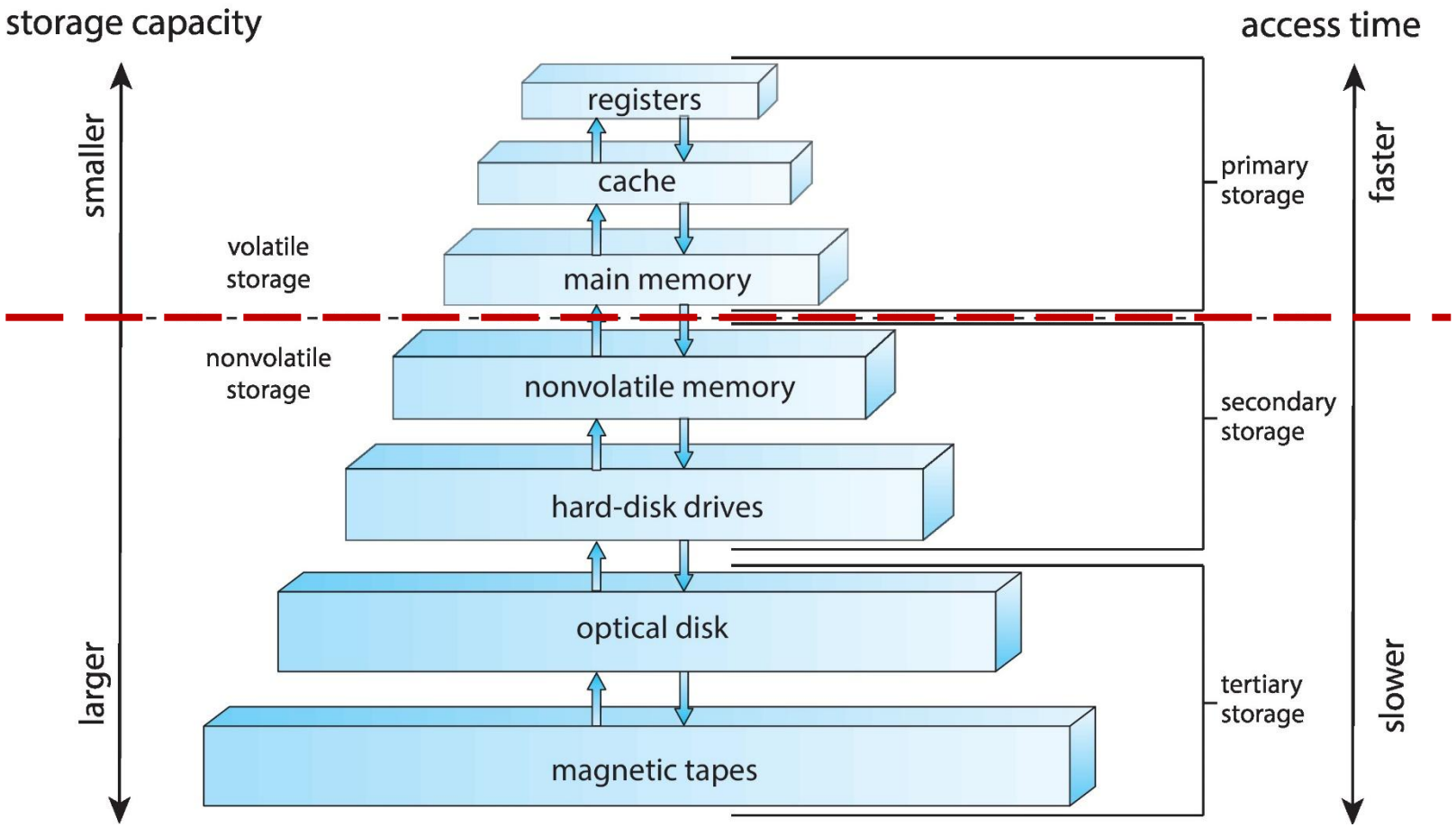
How a modern computer system works



Storage Structure

- **The main differences among various storage systems lie in**
  - speed
  - cost
  - size
  - volatility (揮發性)
- **Caching**
  - copying information into faster storage system; main memory can be viewed as a cache for secondary storage

Storage-Device Hierarchy
(Fig. 1.6)



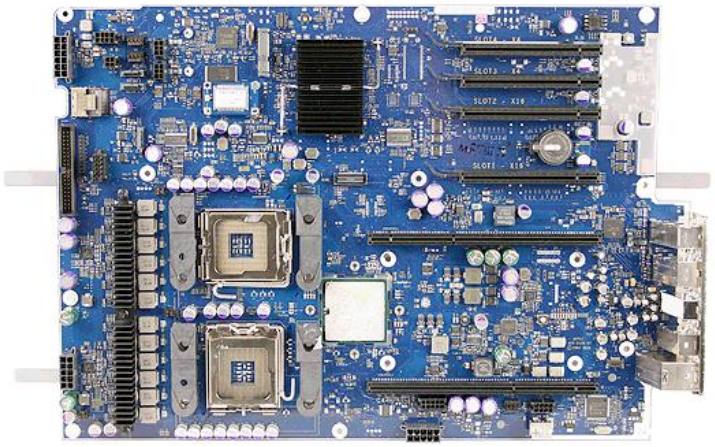
1.3 Computer-System Architecture

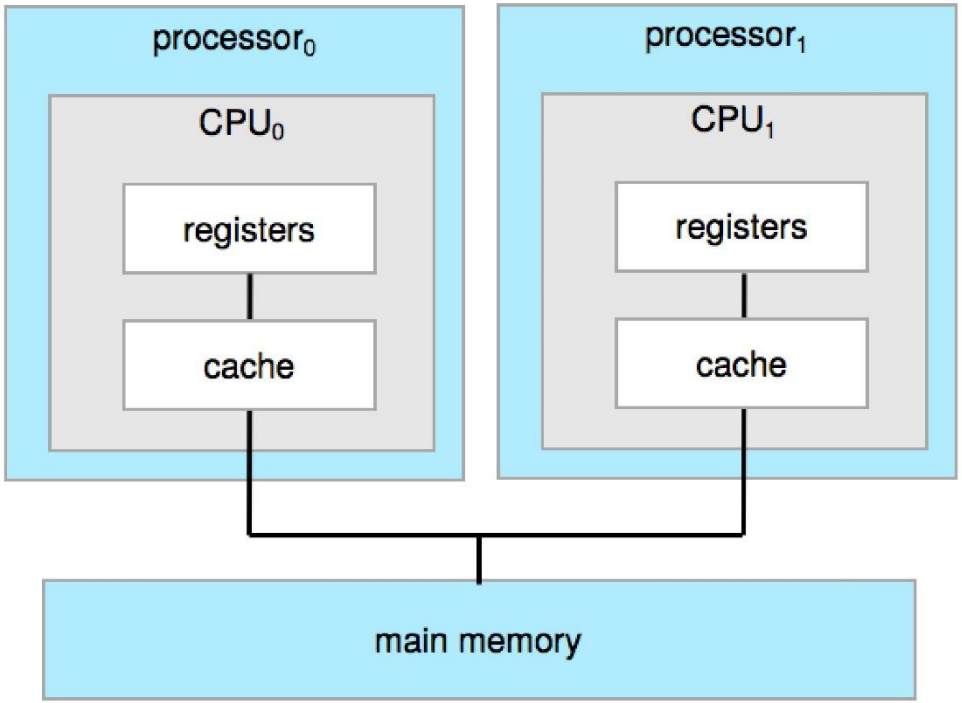
- **Categorize roughly according to the number of general-purpose processors used:**
  - Single-Processor Systems
  - Multiprocessor Systems
  - Clustered Systems
- **Impact on OS design**

Multiprocessor Systems

- ***Multiprocessor systems*** **have two or more processors, sharing the computer bus and sometimes the clock, memory, and peripheral devices.**

Symmetric Multiprocessing Architecture

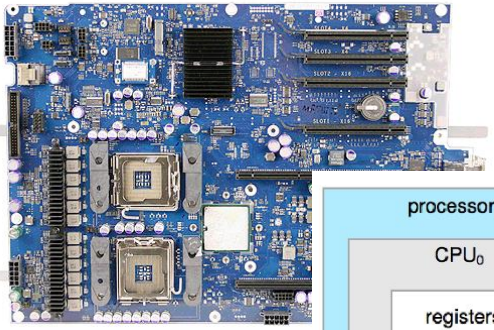


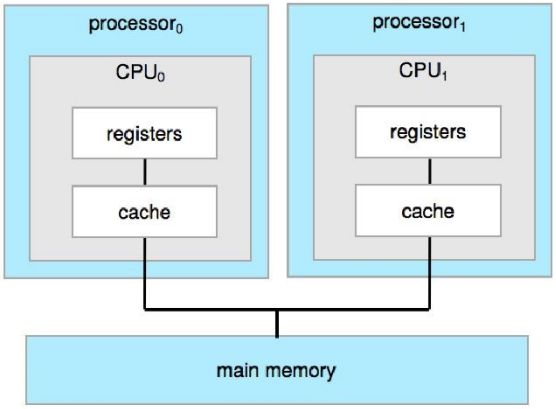


Multiprocessor Systems (Cont.)

- **Main advantage of multiprocessor systems:**
  - Increased *throughput*
- **Most systems use *Symmetric multiprocessing (SMP)***
  - Each peer CPU processor performs all tasks, including OS functions and user processes.
  - Many processes can run simultaneously without significant performance deterioration.
  - Modern OSs (Windows, macOS, Linux, Android and OS,…) all support SMP
- ***Asymmetric multiprocessing***
  - Each processor is assigned a specific task.
  - eg., boss processor schedules and allocates work to worker processors.

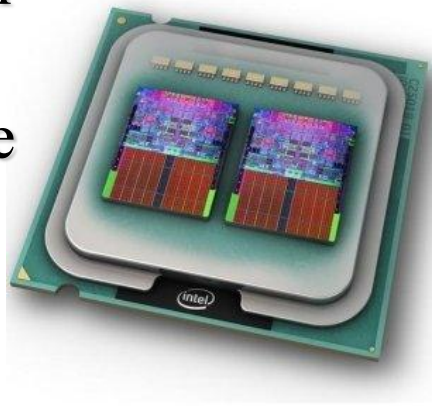
Multiprocessor Systems (Cont.)

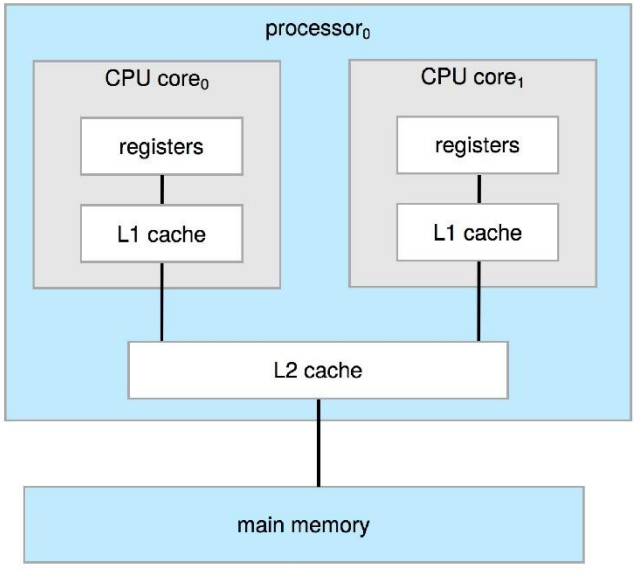




- **Multiple chips**
- **Multicore**
  - Multiple computing cores on a single chip
  - Uses less power, more efficient than multiple single-core chips
    - on-chip communication is faster
  - These multicore CPUs appear to the OS as N standard processors

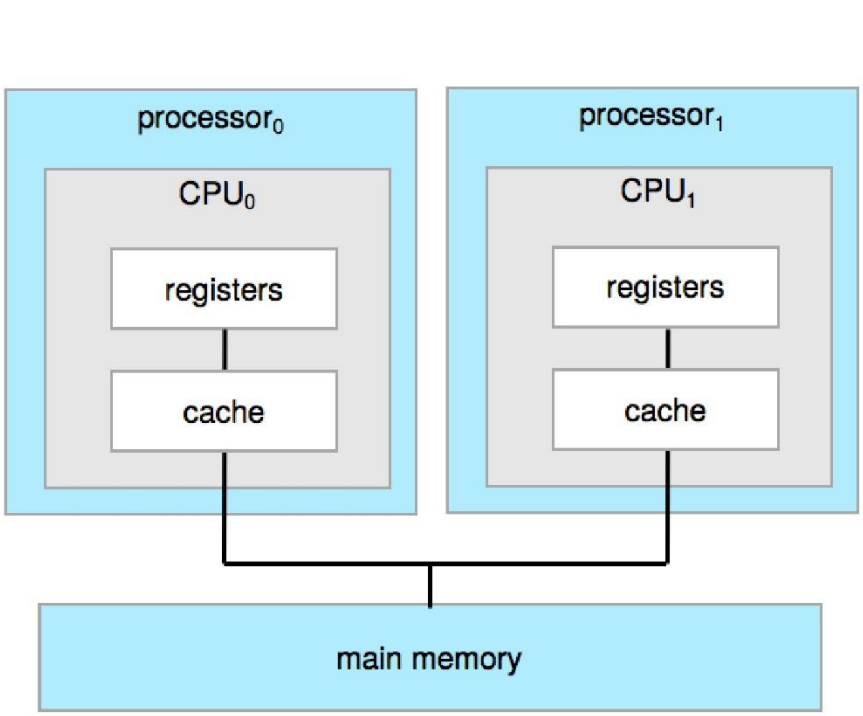
A dual-core design with two cores placed on the same chip

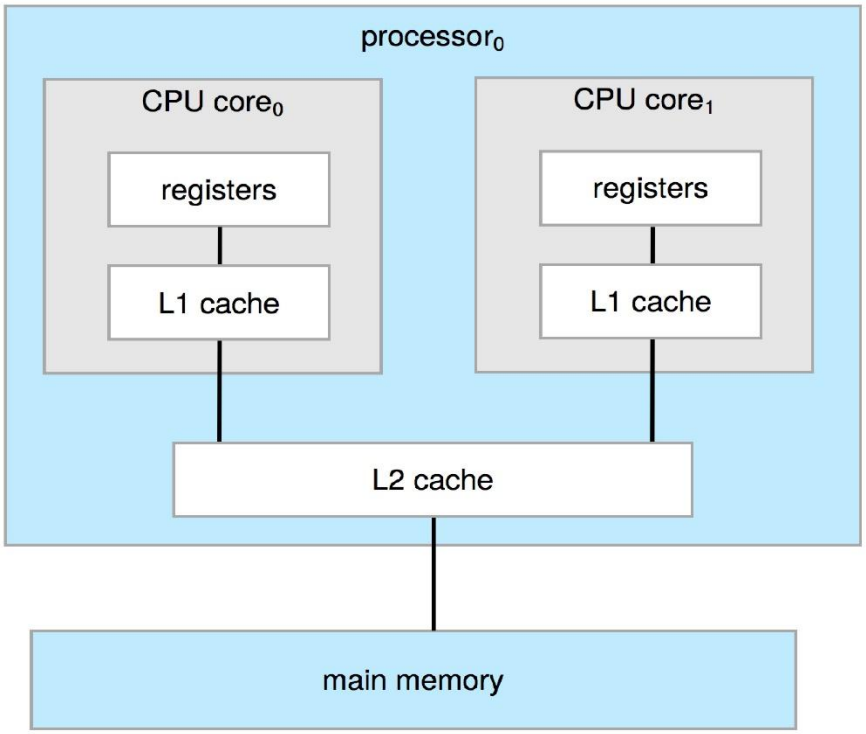




Memory Access Model

- **Uniform Memory Access (UMA)**
  - Access to any RAM from any CPU takes the same time
- ***Symmetric multiprocessing (SMP) architecture***

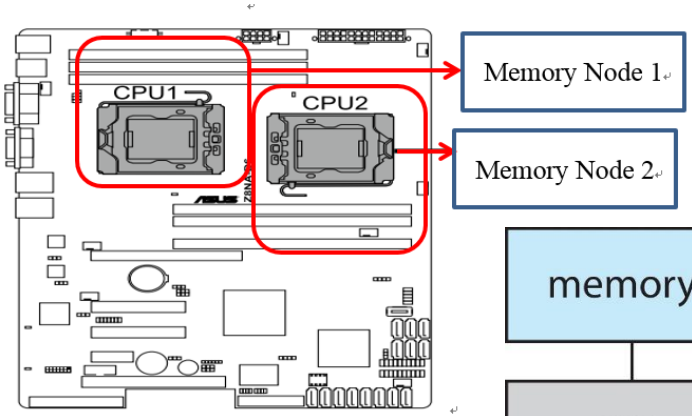


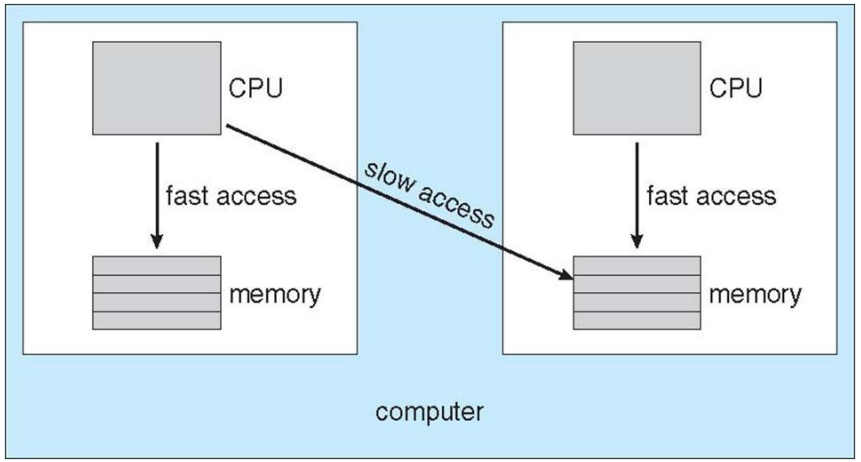


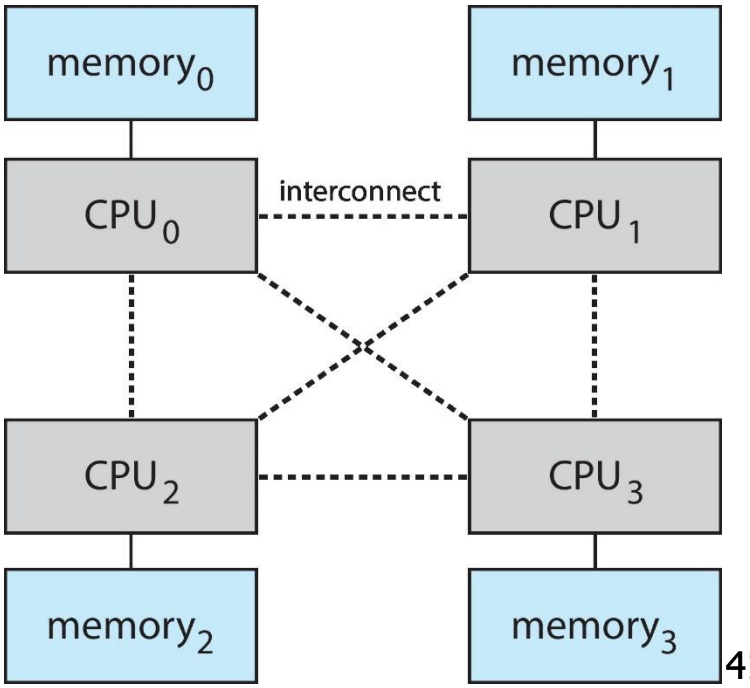
Memory Access Model (Cont.)

Non-uniform Memory Access (NUMA) Systems

- A CPU has faster access to some parts of memory than to other parts





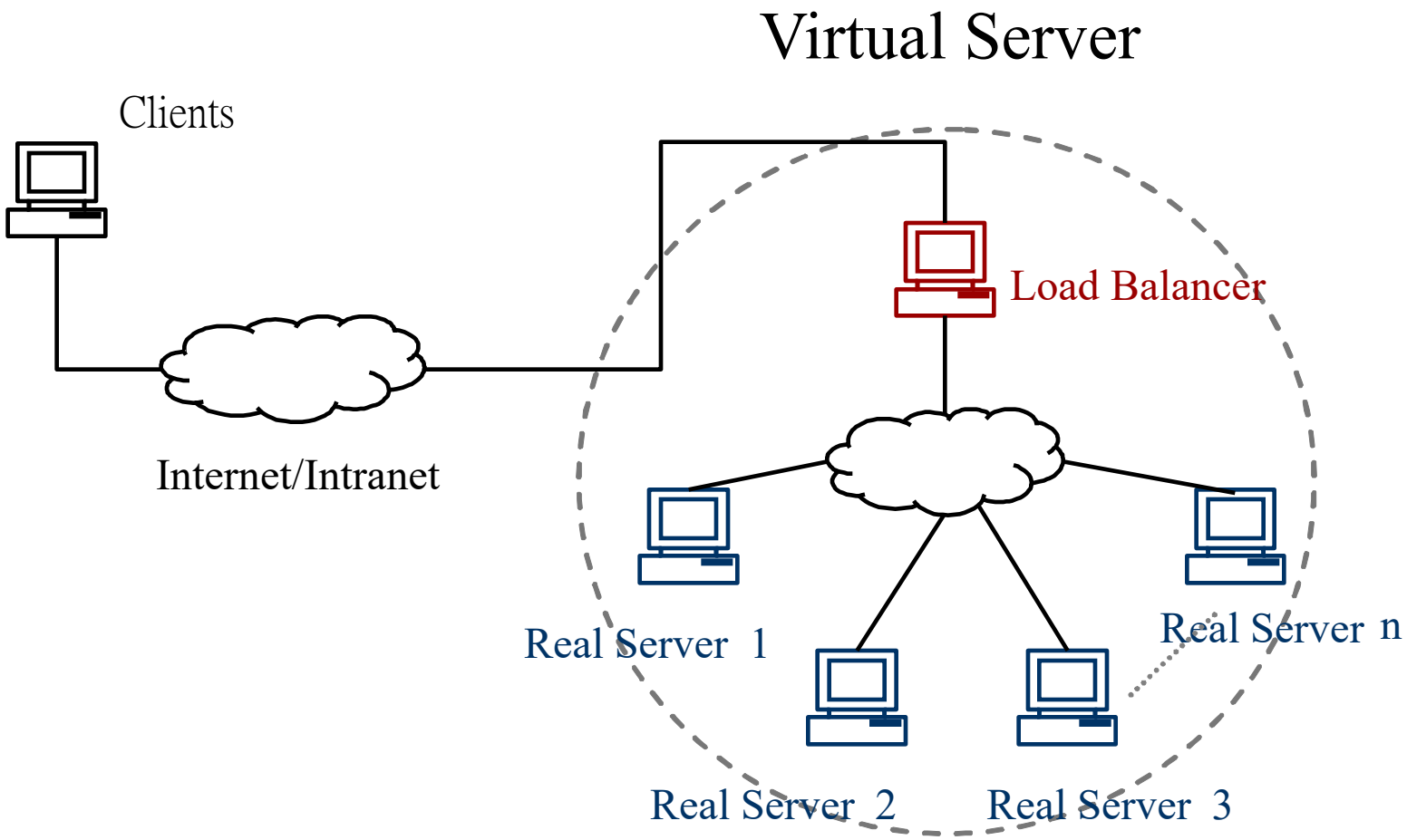


叢集

Clustered Systems

- **Composed of two or more individual systems or nodes joined together**
  - e.g., web cluster, game server cluster, computing cluster
- **The generally accepted definition:**
  - clustered computers share storage and are closely linked via LAN networking or a faster interconnect such as InfiniBand.

Example: LVS Cluster Architecture



Clustered Systems (Cont.)

可利用性

- **Usually used to provide high availability**
  - Service will continue even if one or more systems in the cluster fail
  - Increased *reliability* (可靠度)
  - The ability to continue providing service proportional to the level of surviving hardware is called graceful degradation.

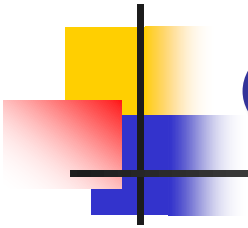
倖存；殘留

- ***Fault-tolerant* systems**

容錯

  - Systems that can suffer a failure of any single component and still continue operation
  - e.g., the HP NonStop system (formerly Tandem) system uses both H/W and S/W duplication to ensure continued operation despite faults.

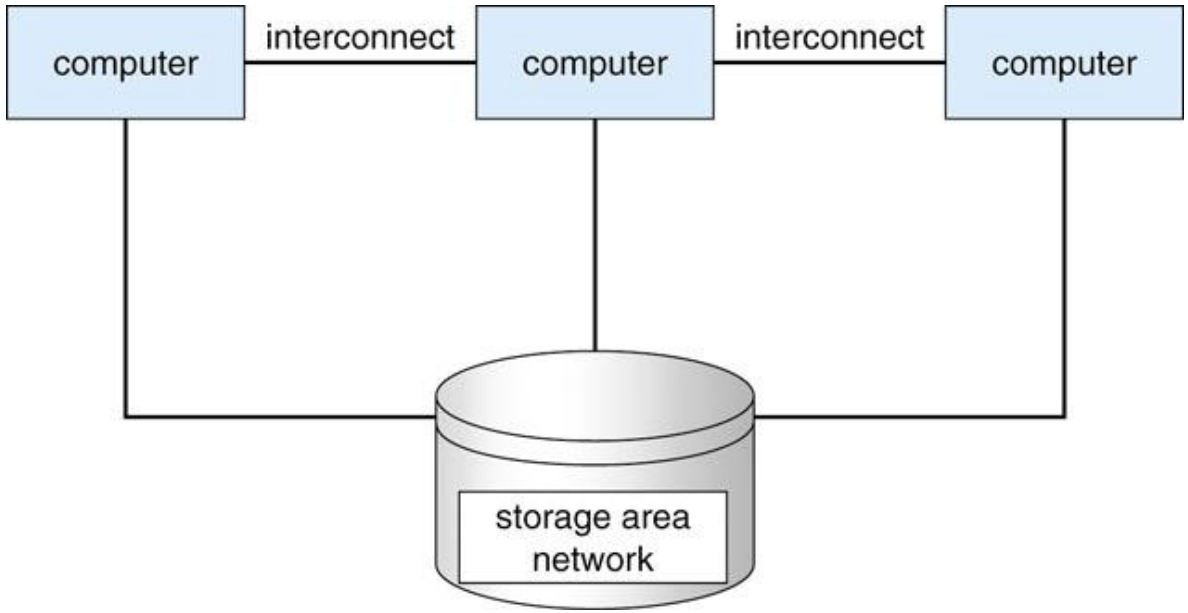
Clustered Systems (Cont.)



- **Can be structured asymmetrically or symmetrically**
  - In *asymmetric clustering*, one machine is in hot standby mode while the other is running the applications.
  - In *symmetric clustering*, two or more hosts are running applications, monitoring each other
- **Clusters can also be used to provide high-performance computing environments**
  - Applications must be written to use parallelization
  - Distributed lock manager (DLM)
    - supply access control and locking to avoid conflicting operations on shared access
- **Cluster technology is rapidly changing.**

General structure of a clustered system (Fig. 1.8)

- Usually sharing storage via a storage-area network (SAN), which allow many systems to attach to a pool of storage

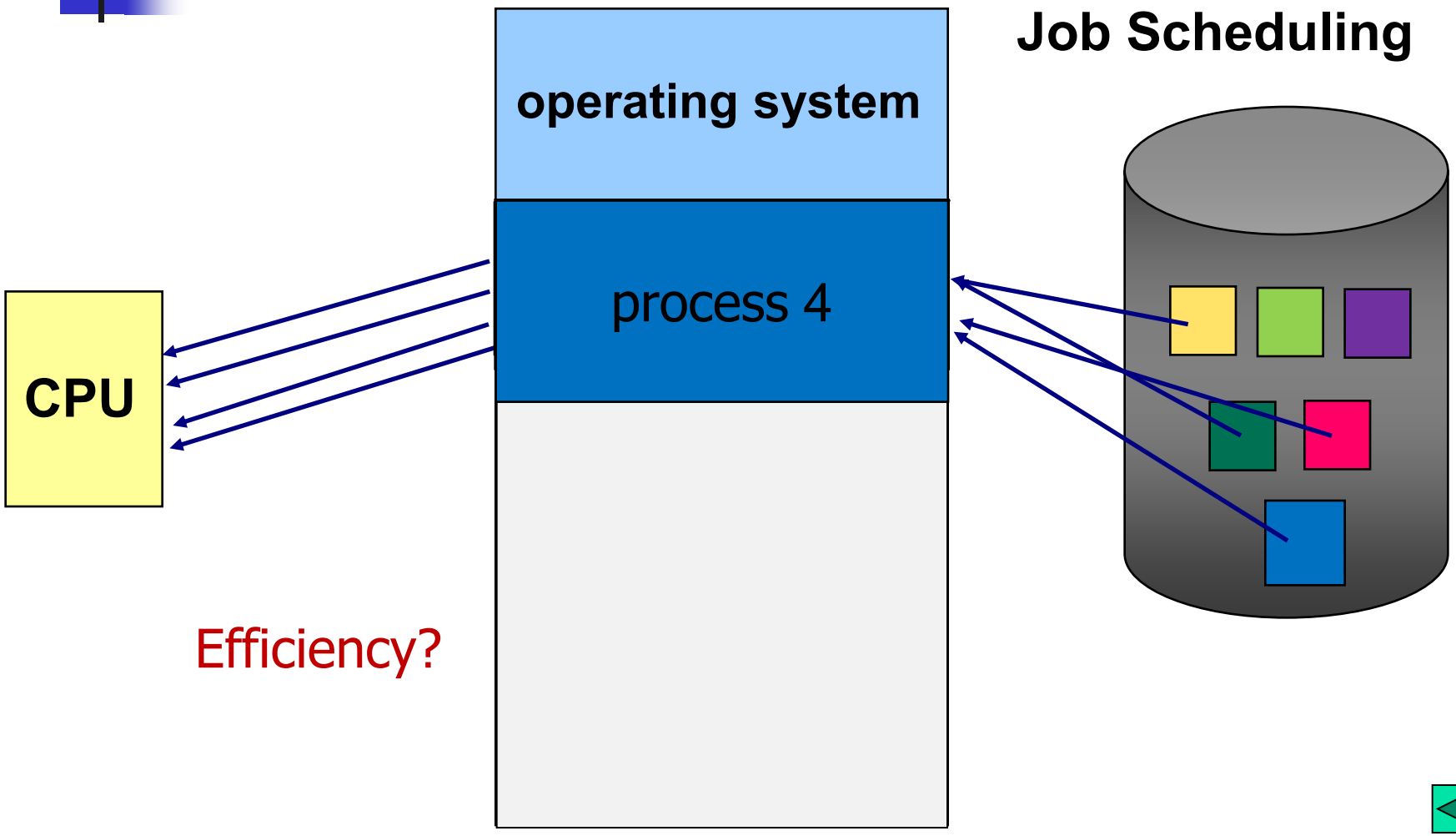


1.4 Operating-System Operations

1.4.1 Multiprogramming & Multitasking

- **Single tasking**
  - Single program cannot keep either CPU or I/O devices busy at all times
- **Multiprogramming system**
- **Multitasking**

Single Tasking

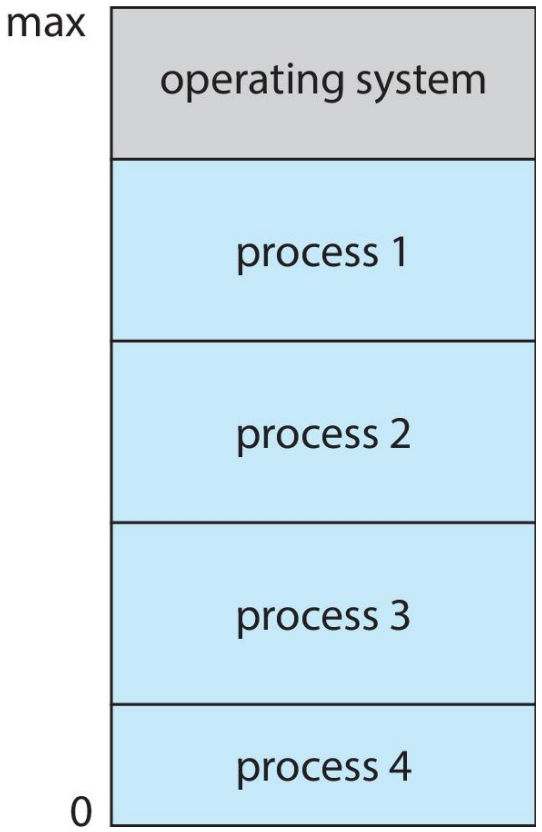


Multiprogramming System

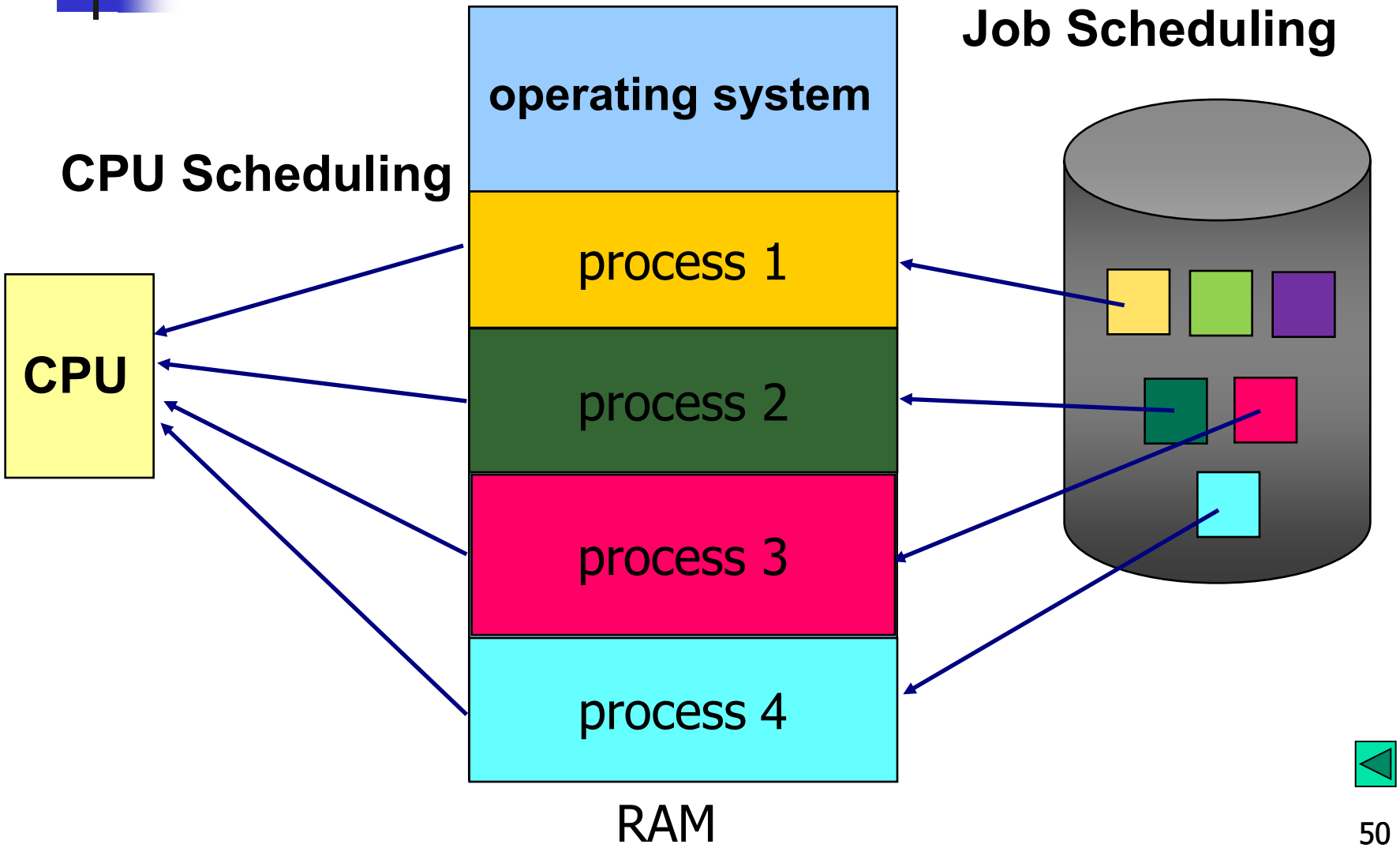
- ***Multiprogramming*** **increases CPU utilization by organizing programs so that CPU always has one to execute.**
  - A program in execution is termed a process.
  - OS keeps several processes in memory simultaneously.
  - When one process being executed needs to wait, the OS switches to and executes another process.

Efficiency? Fair?

Memory Layout for a Multiprogramming System



Multiprogramming's impact on OS design?



Multitasking Systems

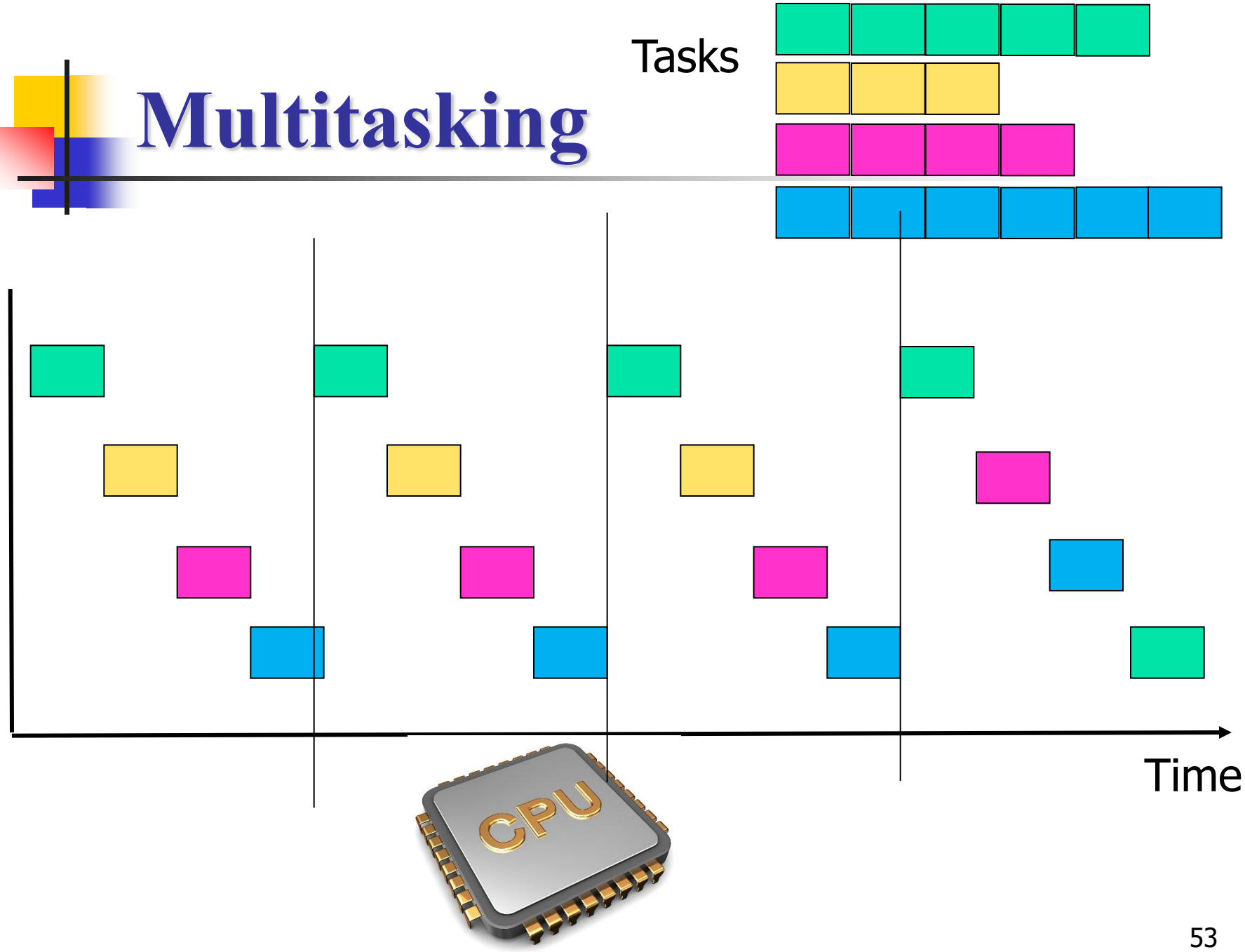
- ***Multitasking*** is a logical extension of multiprogramming.
- The CPU executes multiple processes by switching among them, but the switches occur so frequently, providing the user with a fast response time.
- A time-shared OS uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer.

Time-Sharing Systems (Cont.)

- ***Time-sharing***
  - The technique of dividing time into intervals, or time slices, and then restricting the execution of a job to only one time slice at a time.
  - At the end of each time slice, the current job is set aside and another is allowed to execute during the next time slice.
  - By rapidly shuffling the jobs in this manner, the illusion of several jobs executing simultaneously is created.

洗牌; 攪亂; 改組

Multitasking

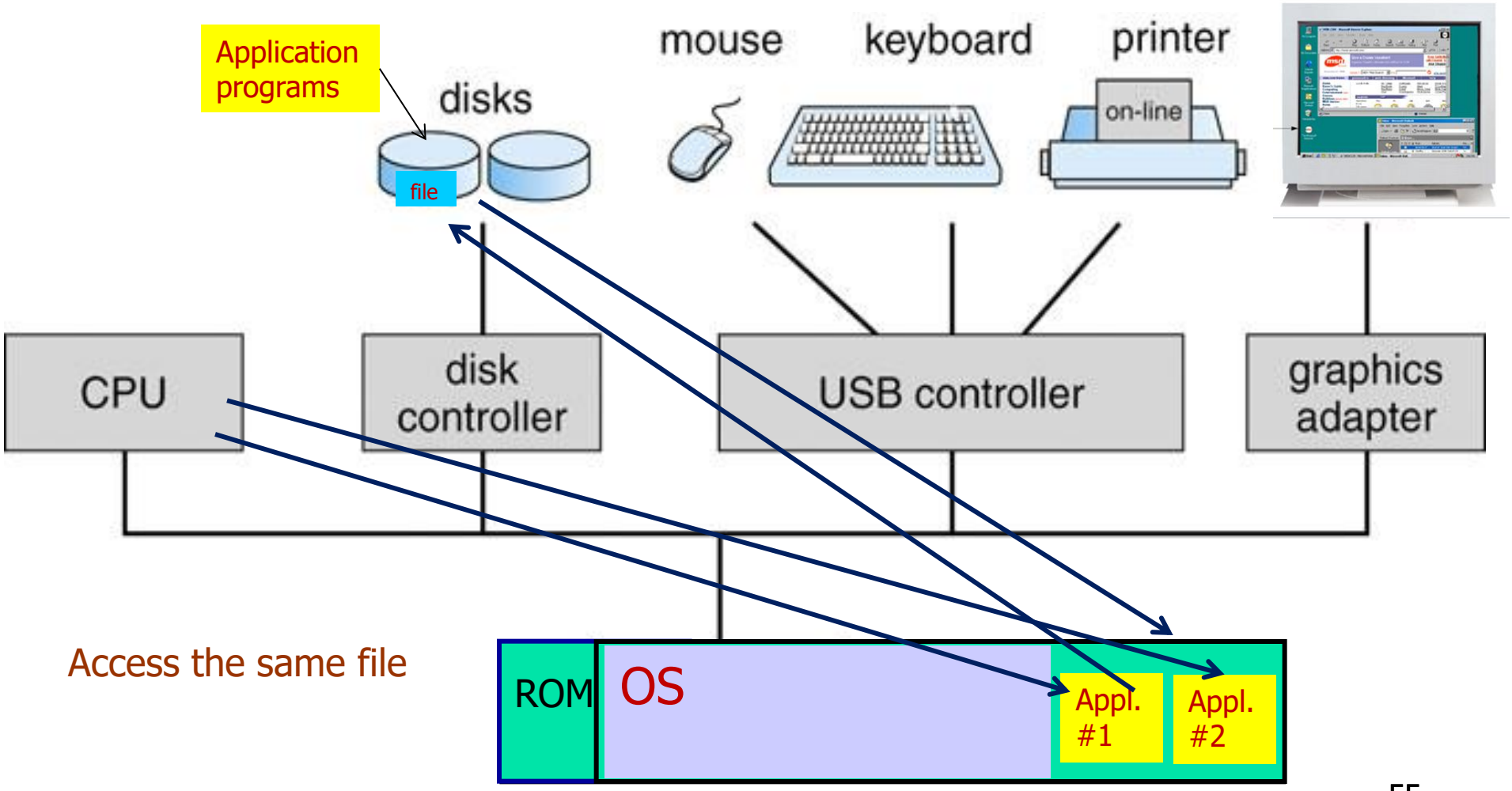


OS Features Needed for Multiprogramming Systems

- ***Memory management***
- ***CPU scheduling***
  - The system must choose next job that are ready to run in memory to be run next by CPU
- **Concurrency control**
  - Running multiple jobs concurrently requires that their ability to affect one another be limited in all phases of the OS, including process scheduling, disk storage, and memory management.

同時發生

Concurrent Execution



Access the same file

Other OS Features also Needed for Multitasking Systems

- **OS must ensure reasonable response time**
- ***Virtual memory***
  - A technique that allows execution of processes not completely in memory
- ***File system*** **and** ***storage management***
- ***Protection***
- ***Process synchronization*** **and** ***communication***
  - To ensure orderly execution
- ***Deadlock avoidance***

Operating-System Operations

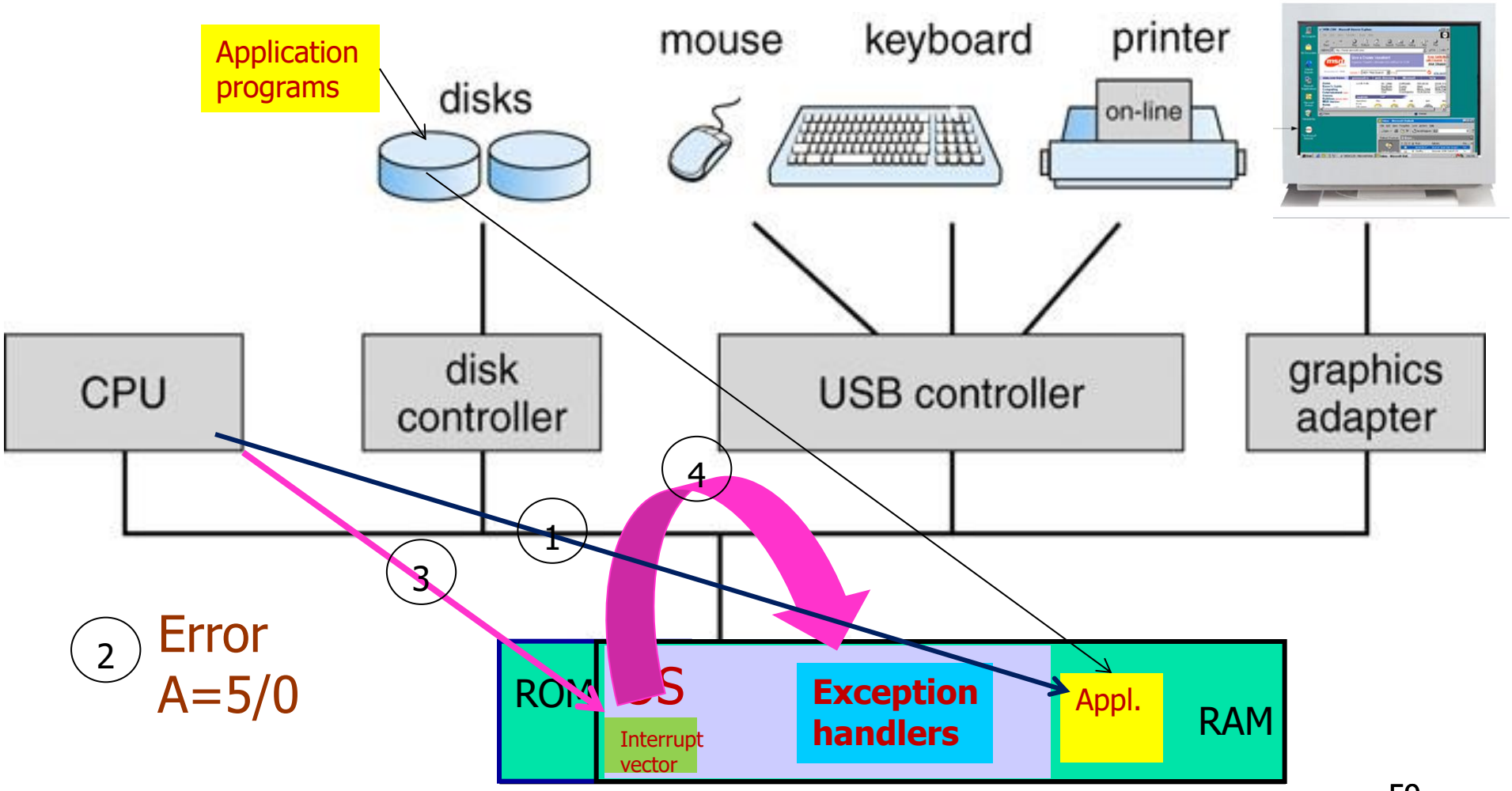
- **Modern OS are *interrupt* driven:**
  - If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an OS will sit quietly, waiting for something to happen.
  - Events are almost always signaled by the occurrence of an interrupt or a trap.
- **A *trap* (or *exception*) is a S/W generated interrupt caused either by**
  - an error (e.g., divide by zero, invalid memory access) or
  - a specific request from a user program that an OS service be performed (i.e., system call).

Exceptions and Interrupts (Intel CPU)

Table 6-1. Exceptions and Interrupts

Vector	Mnemonic	Description	Source
0	#DE	Divide Error	DIV and IDIV instructions.
1	#DB	Debug	Any code or data reference.
2		NMI Interrupt	Non-maskable external interrupt.
3	#BP	Breakpoint	INT 3 instruction.
4	#OF	Overflow	INTO instruction.
5	#BR	BOUND Range Exceeded	BOUND instruction.
6	#UD	Invalid Opcode (UnDefined Opcode)	UD2 instruction or reserved opcode.[1]
7	#NM	Device Not Available (No Math Coprocessor)	Floating-point or WAIT/FWAIT instruction.
8	#DF	Double Fault	Any instruction that can generate an exception, an NMI, or an INTR.
9	#MF	CoProcessor Segment Overrun (reserved)	Floating-point instruction.[2]
10	#TS	Invalid TSS	Task switch or TSS access.
11	#NP	Segment Not Present	Loading segment registers or accessing system segments.
12	#SS	Stack Segment Fault	Stack operations and SS register loads.
13	#GP	General Protection	Any memory reference and other protection checks.
14	#PF	Page Fault	Any memory reference.
15		Reserved	
16	#MF	Floating-Point Error (Math Fault)	Floating-point or WAIT/FWAIT instruction.
17	#AC	Alignment Check	Any data reference in memory.[3]
18	#MC	Machine Check	Error codes (if any) and source are model dependent.[4]
19	#XM	SIMD Floating-Point Exception	SIMD Floating-Point Instruction[5]
20-31		Reserved	
32-255		Maskable Interrupts	External interrupt from INTR pin or INT *n* instruction.

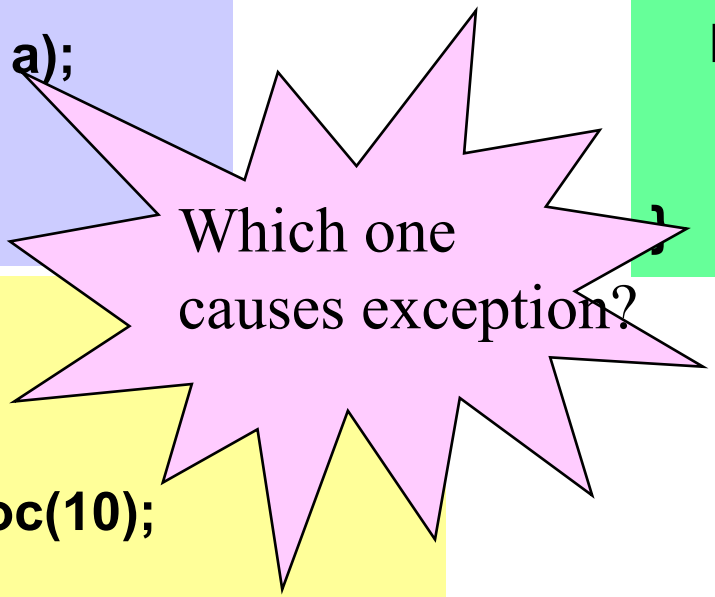
When does an OS run?
(3) exception occurs



Program Misbehavior

```
int main() {
    int a;

    a=5/0;
    printf("a=%d", a);

    return 0;
}
```

```
int main() {
    char *b;

    b=(char *)0;
    b[10]='A';

    printf("b[10]=%c", b[10]);

    return 0;
}
```

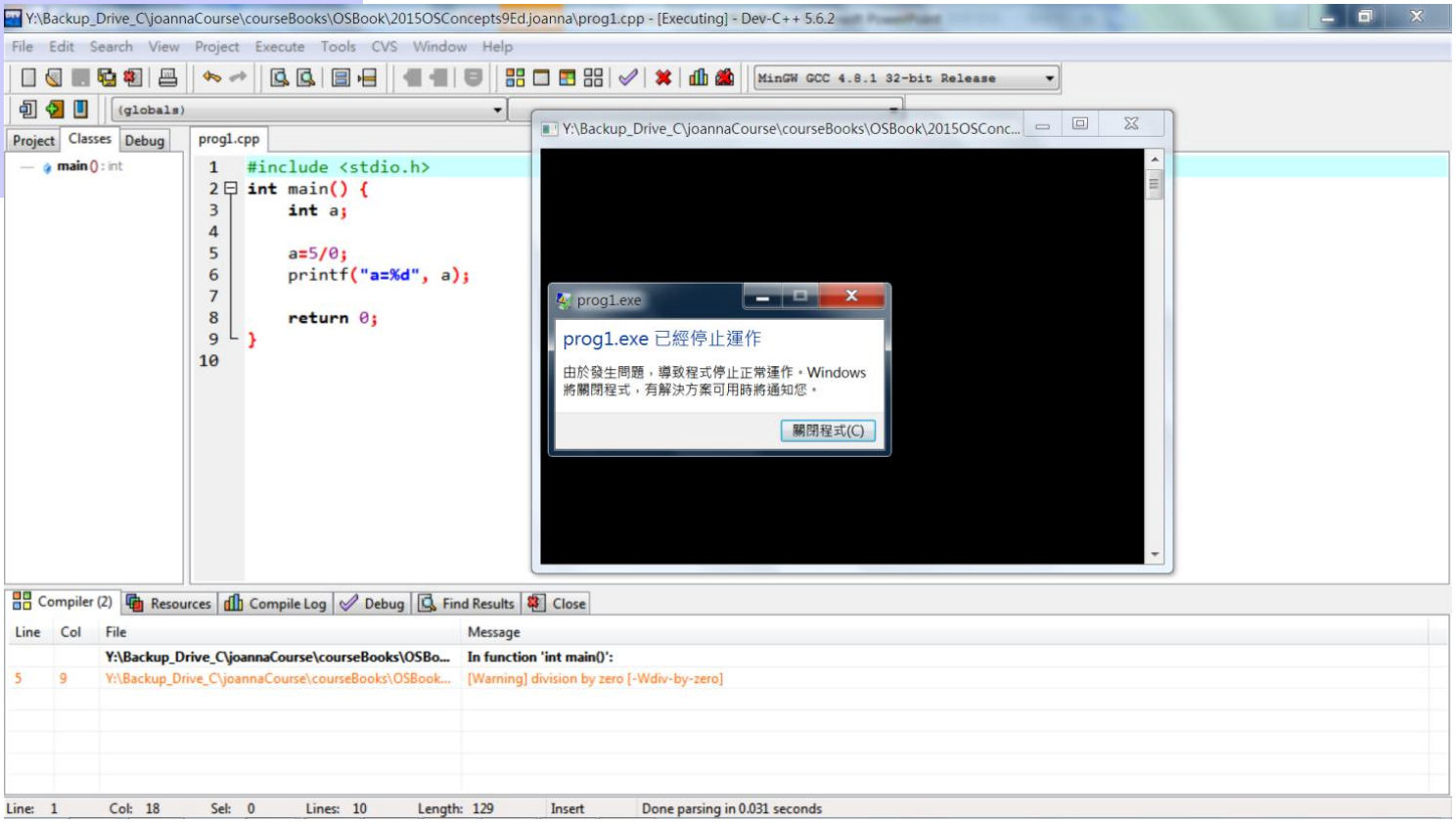



```
int main() {
    char *b;

    b=(char *)malloc(10);
    b[10000]='A';

    printf("b[10000]=%c", b[10000]);

    return 0;
}
```

```
int main() {
    char *b;

    while (1) {
    };

    printf("GG");

    return 0;
}
```

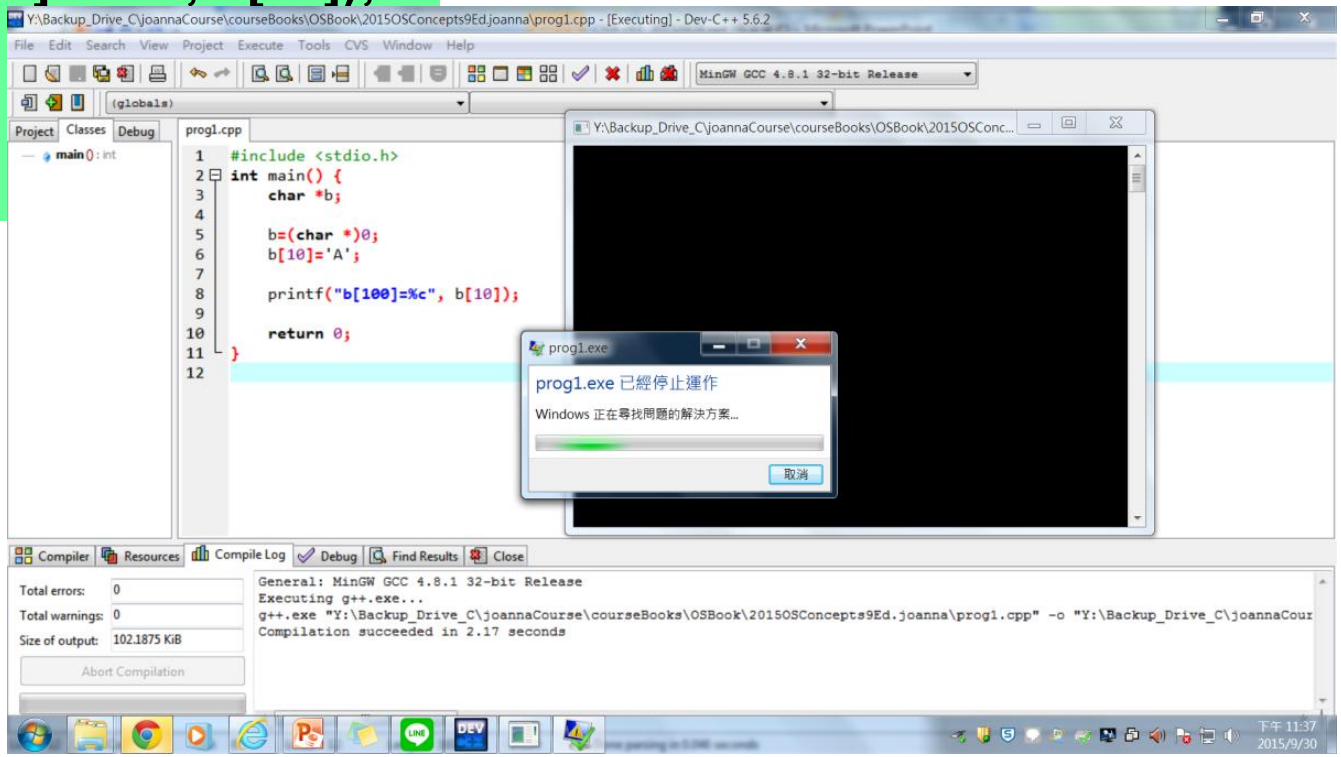
Program Misbehavior

```
#include <stdio.h>
int main() {
    int a;

    a=5/0;
    printf("a=%d", a);

    return 0;
}
```



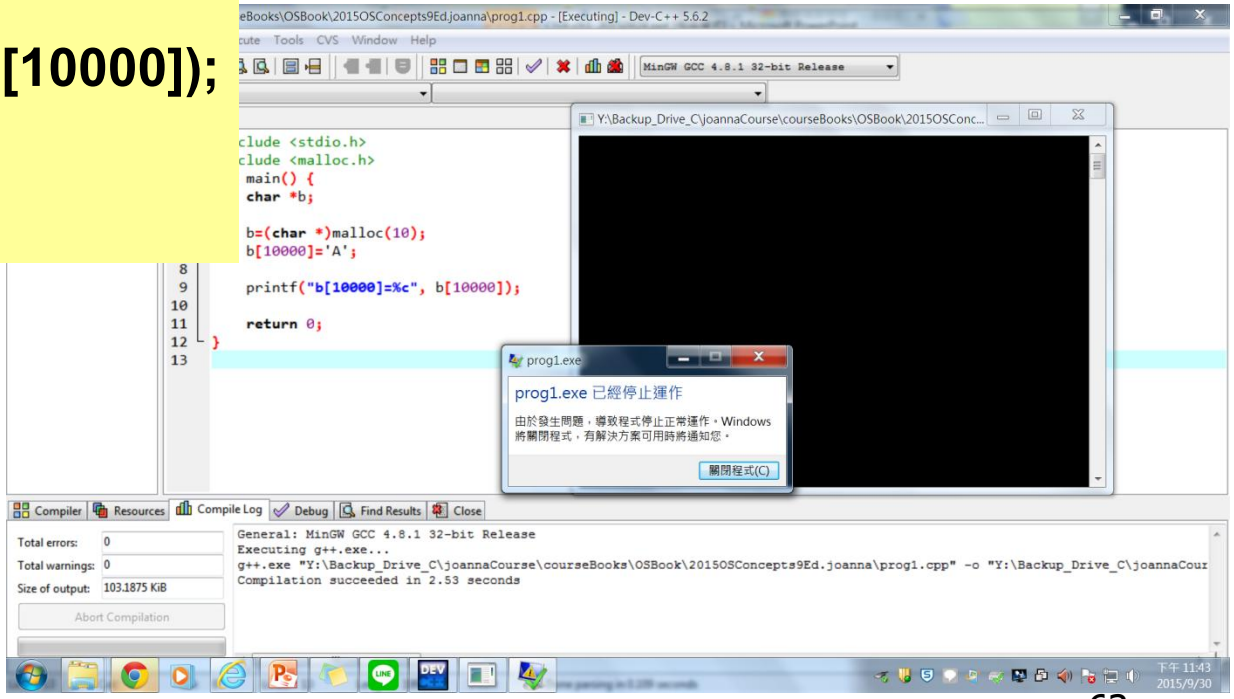
Program Misbehavior

```
#include <stdio.h>
int main() {
    char *b;

     b=(char *)0;
    b[10]='A';

    printf("b[10]=%c", b[10]);

     return 0;
}
```



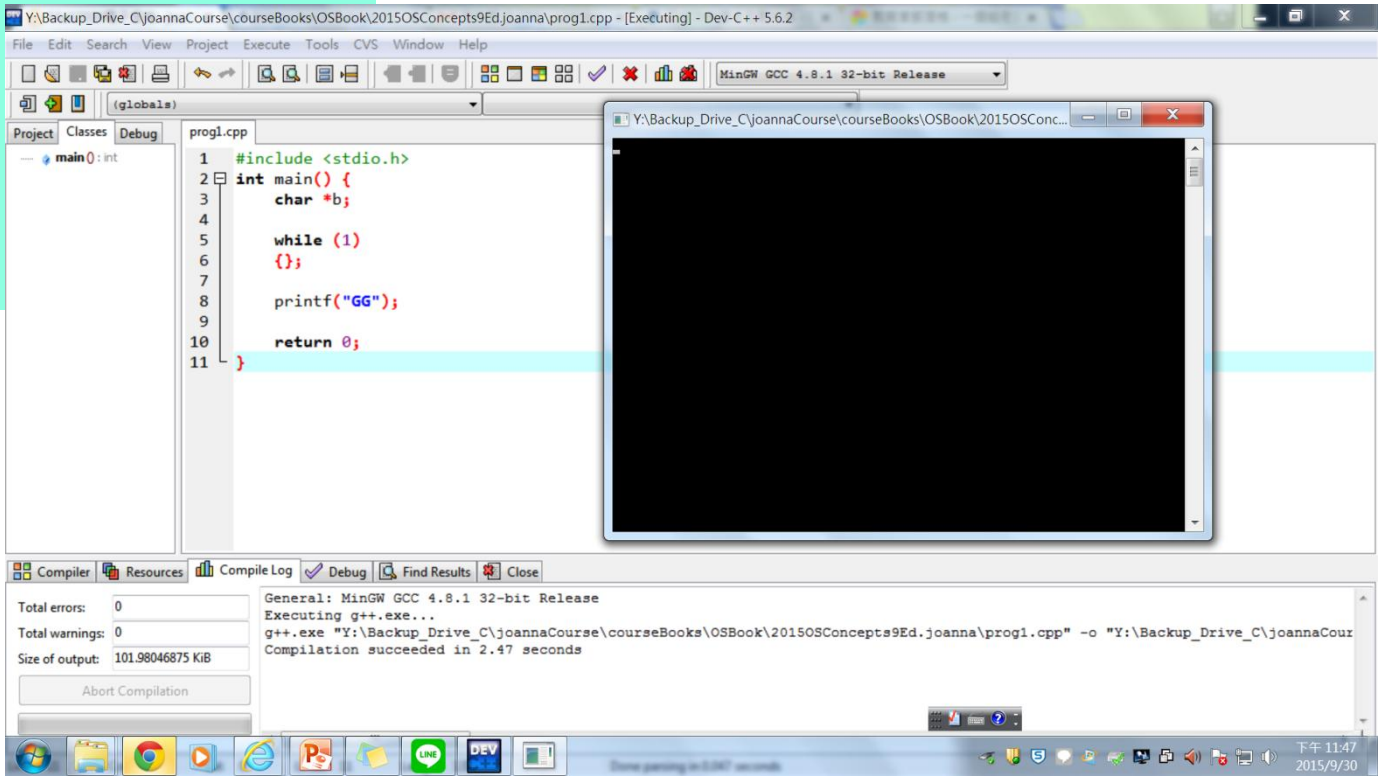
Program Misbehavior

```
#include <stdio.h>
#include <malloc.h>
int main() {
    char *b;

     b=(char *)malloc(10);
    b[10000]='A';

    printf("b[10000]=%c", b[10000]);

     return 0;
}
```

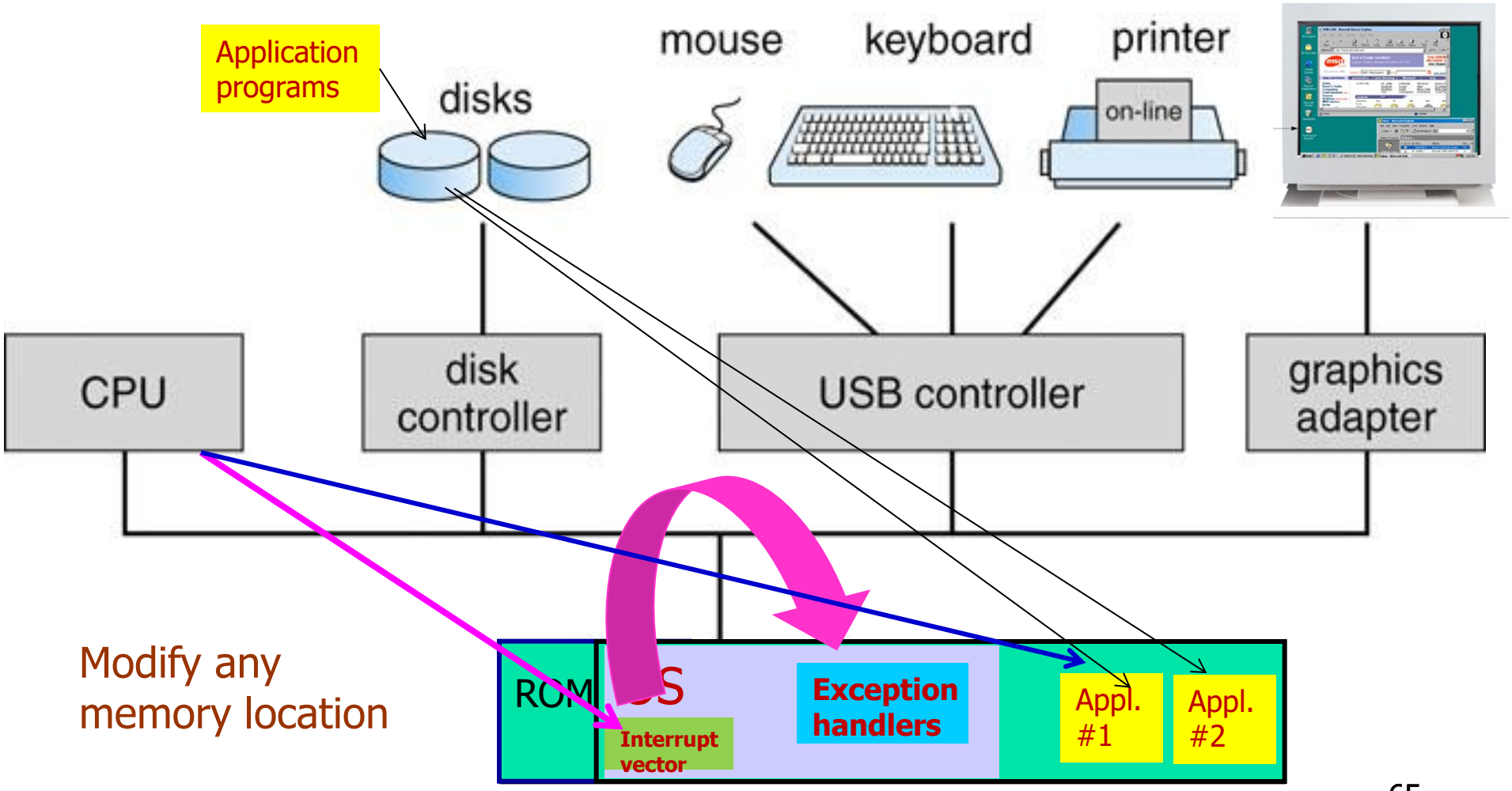


Program Misbehavior

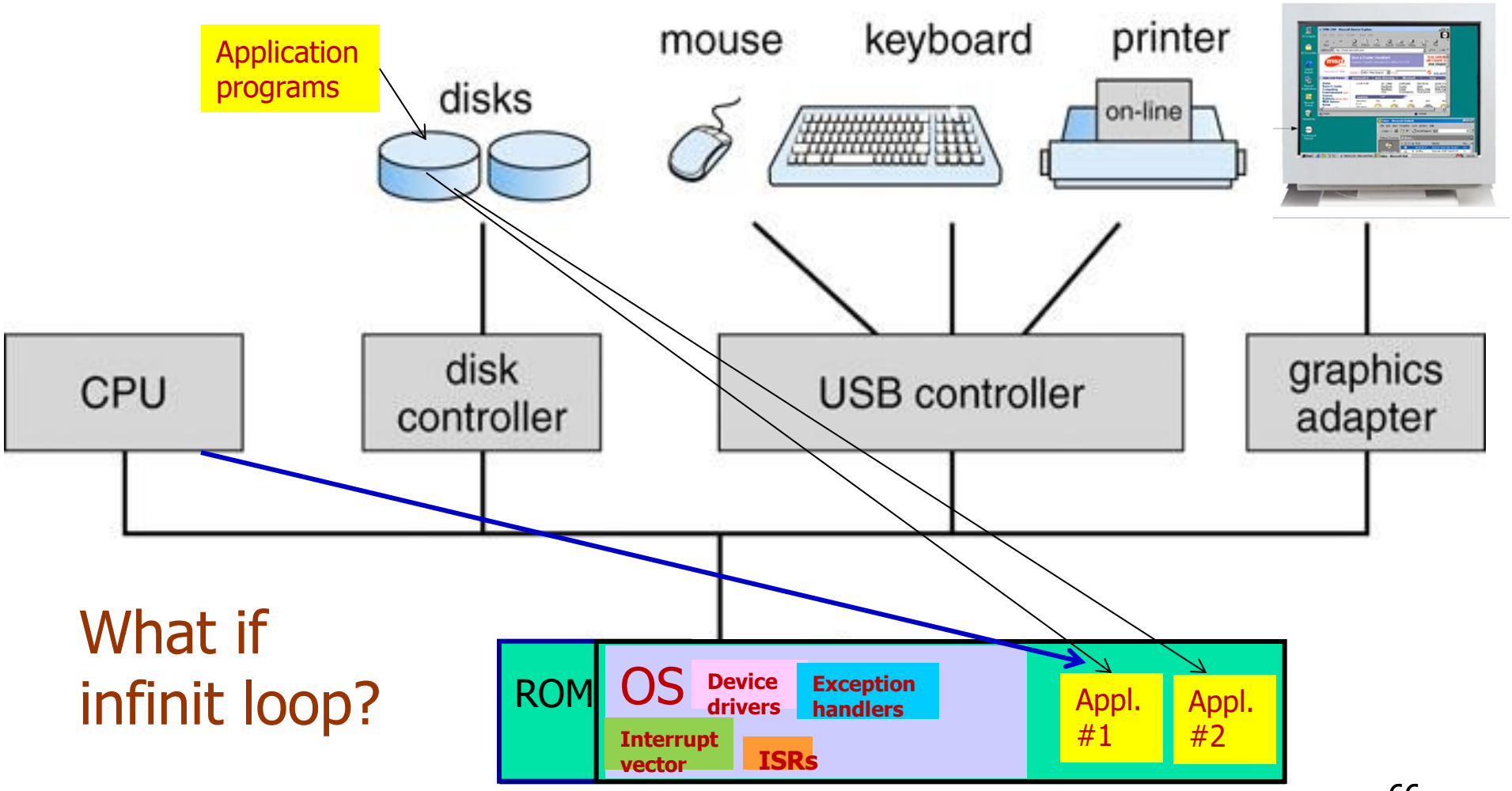
```
#include <stdio.h>
int main() {
    char *b;

     while (1)
     {};

    printf("GG");

     return 0;
}
```



Program misbehavior



Program misbehavior



What if infinit loop?

Operating-System Operations (Cont.)

Interrupt handling

Process misbehavior:

Infinite loop

Modify each other modify OS ….

- For each type of interrupt, separate segments of code (ISR) in OS determine what action should be taken
- Other process problems include infinite loop, processes modifying each other or the OS
- A properly designed OS must ensure that an incorrect (or malicious) program cannot cause other programs or the OS itself to execute incorrectly.

恶意的

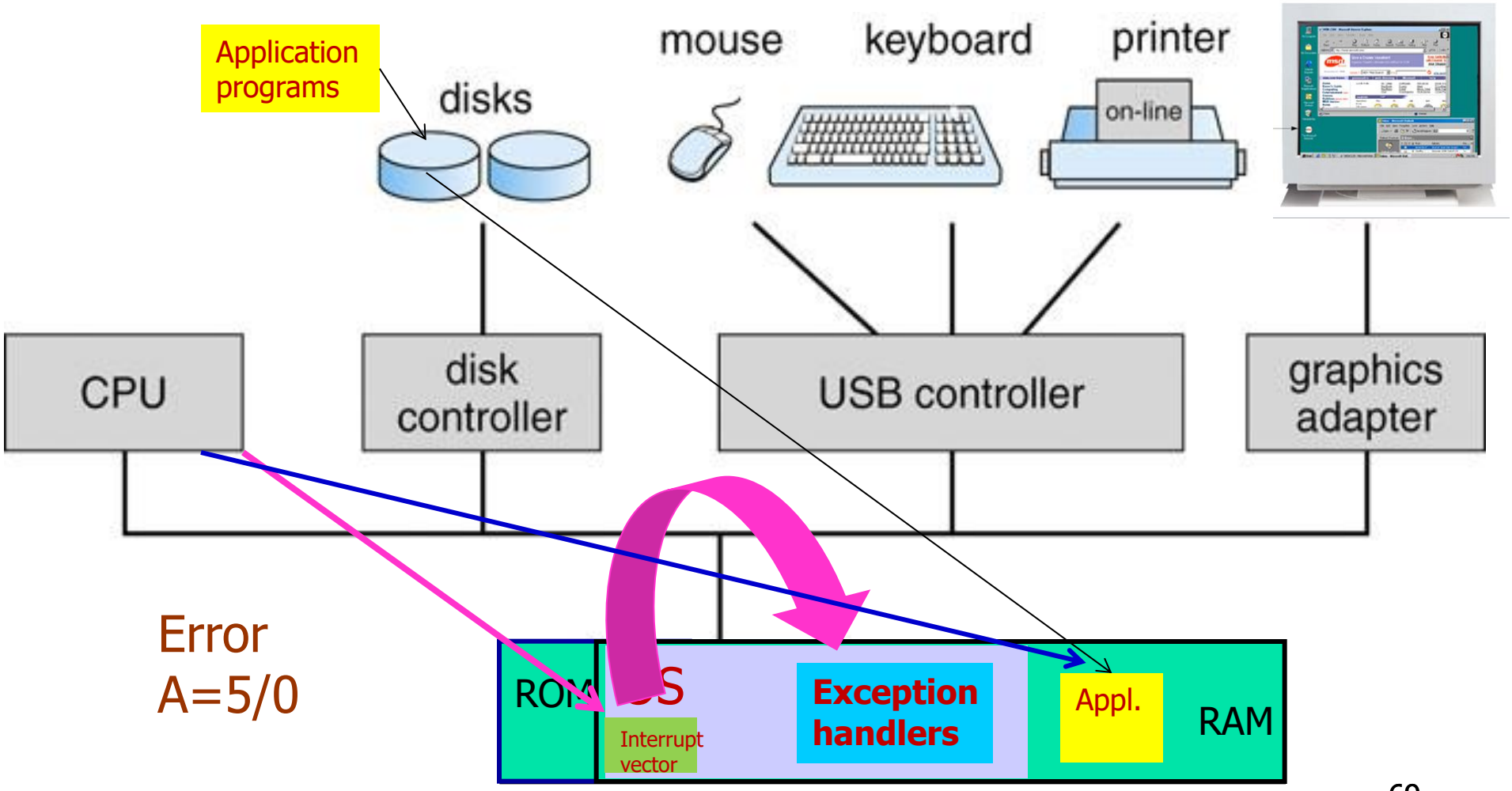
  - 1.4.2 Dual-Mode and Multimode Operation
  - 1.4.3 Timer

1.4.2 Dual-Mode & Multimode Operation

- **Must distinguish between the execution of OS code and user-defined code.**
- **Most computer systems provide H/W support to differentiate at the very least two modes of operation:**
  - *Mode bit* added to computer hardware to indicate the current mode.
  - ***User mode*** **(1)**
    - execution on behalf of a user
  - ***Kernel mode*** **(0)**

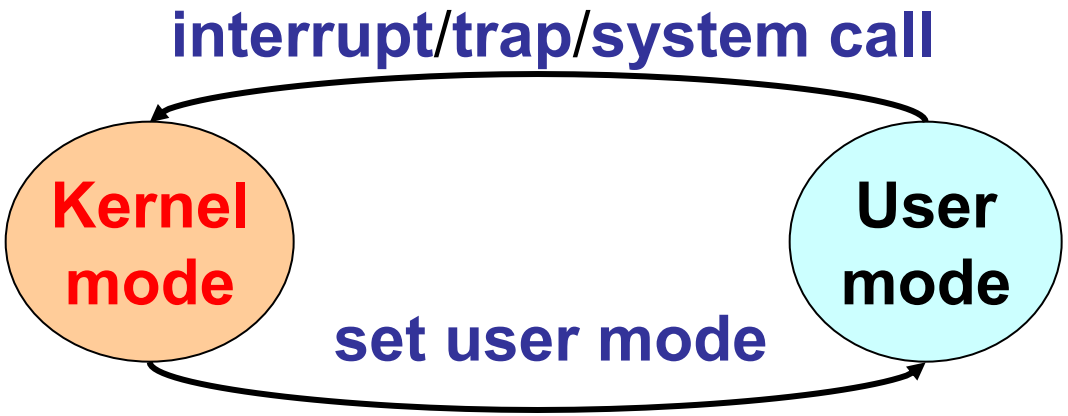
    (also called *supervisor mode*, or *system mode*, *or privileged mode*) 特權模式
    - execution on behalf of OS

When does an OS run?
(3) exception occurs

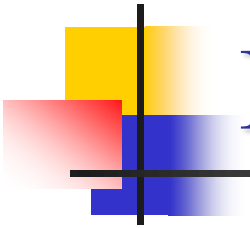


Dual-Mode & Multimode Operation (Cont.)

- When an **interrupt** (H/W interrupt or S/W interrupt) or **trap** occurs, H/W switches to kernel mode to execute OS codes.
  - After completion, the system switches to user mode and control is transferred back to execute user code.

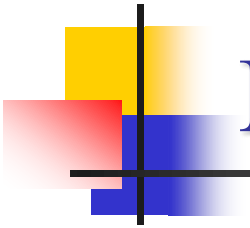


- The dual-mode of operation provides us with the means for protecting the OS from errant users, and errant users from one another. 走入歧途的



Privileged Instructions

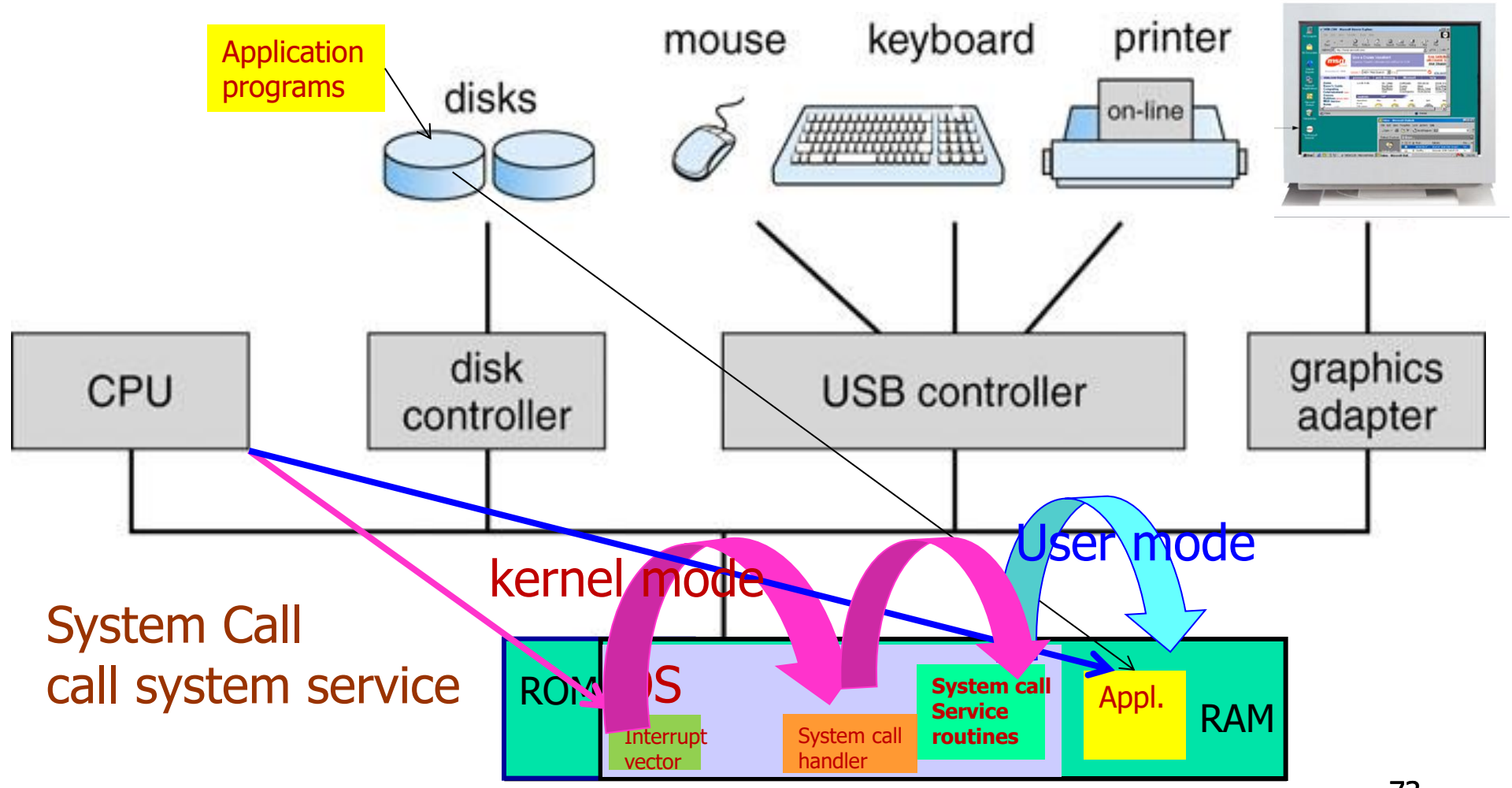
- **Instructions that may cause harm, and can be executed only in kernel mode.**
  - e.g., the instruction to switch to kernel mode, I/O control, timer management, interrupt management, …
- **If an attempt is made to execute a privileged instruction in user mode, hardware treats it as illegal and traps it to the OS.**



Multimode Operation

- **Most contemporary OS, such as Windows, UNIX and Linux, take advantage of this dual-mode feature and provides greater protection for OS.**
- **Increasingly CPUs support multi-mode operations**
  - e.g., some CPUs support virtualization
    - virtual machine manager (VMM) mode for guest VMs
  - Different modes used by various kernel components
  - e.g., Intel 64 family of CPUs supports 4 *privilege levels* (protection rings) and supports virtualization

When does an OS run?
(4) system call occurs



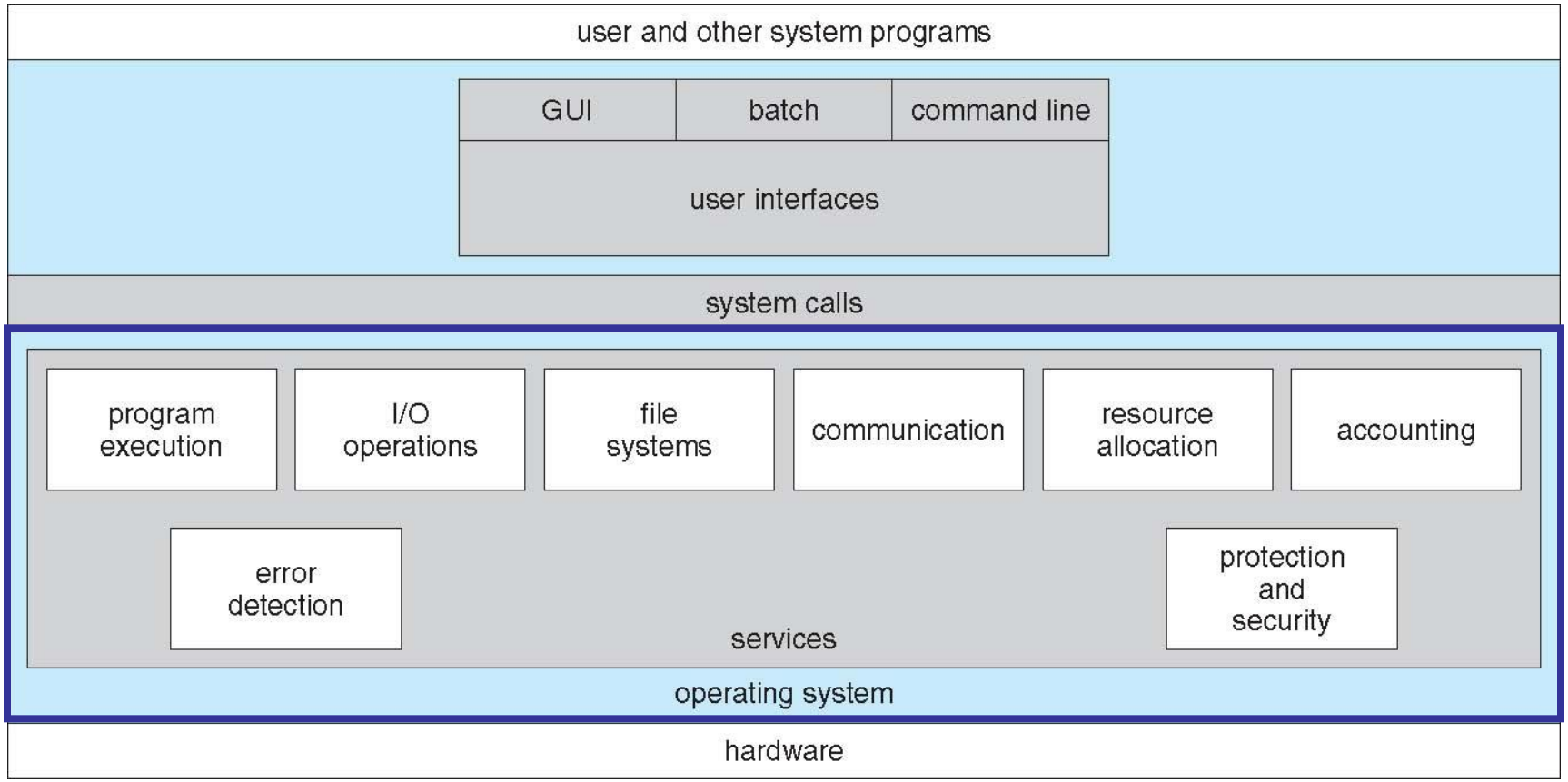
System Call

- ***System call*** **– provide the means for a user program to ask OS to perform OS tasks on the user program's behalf.**
  - Usually takes the form of a **trap** to a specific location in the interrupt vector.
  - This trap can be executed by a generic ***trap*** instruction, though some systems have a specific **syscall** instruction to invoke a system call.
- **When a system call is executed, it is treated by H/W as a S/W interrupt.**

User program invokes system call

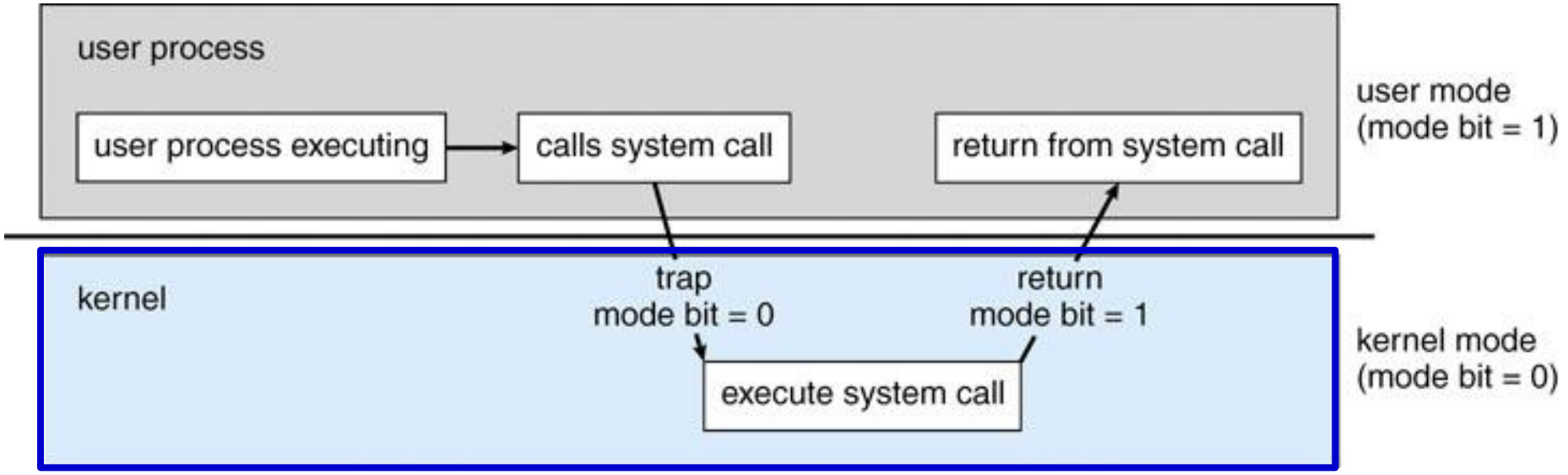
```
int main() {
    int pid=1,i;
    char c[100];

    while (scanf("%s",c)) {
        if (strcmp("exit",c) == 0) {
            exit(0);
        } else {
            pid = fork();    // call C Library in fork.S
            if (pid == 0) {
                printf("%s\n",c);
                i=execlp(c,c,NULL);
                if (i == -1)
                    printf("Command Error!!\n");
                printf("Child terminal.\n");
                exit(0);
            } else {
                printf("Parent had fork a child with PID %d.\n",pid);
            }
        }
    }
}
```

A View of OS Services (Fig. 2.1)

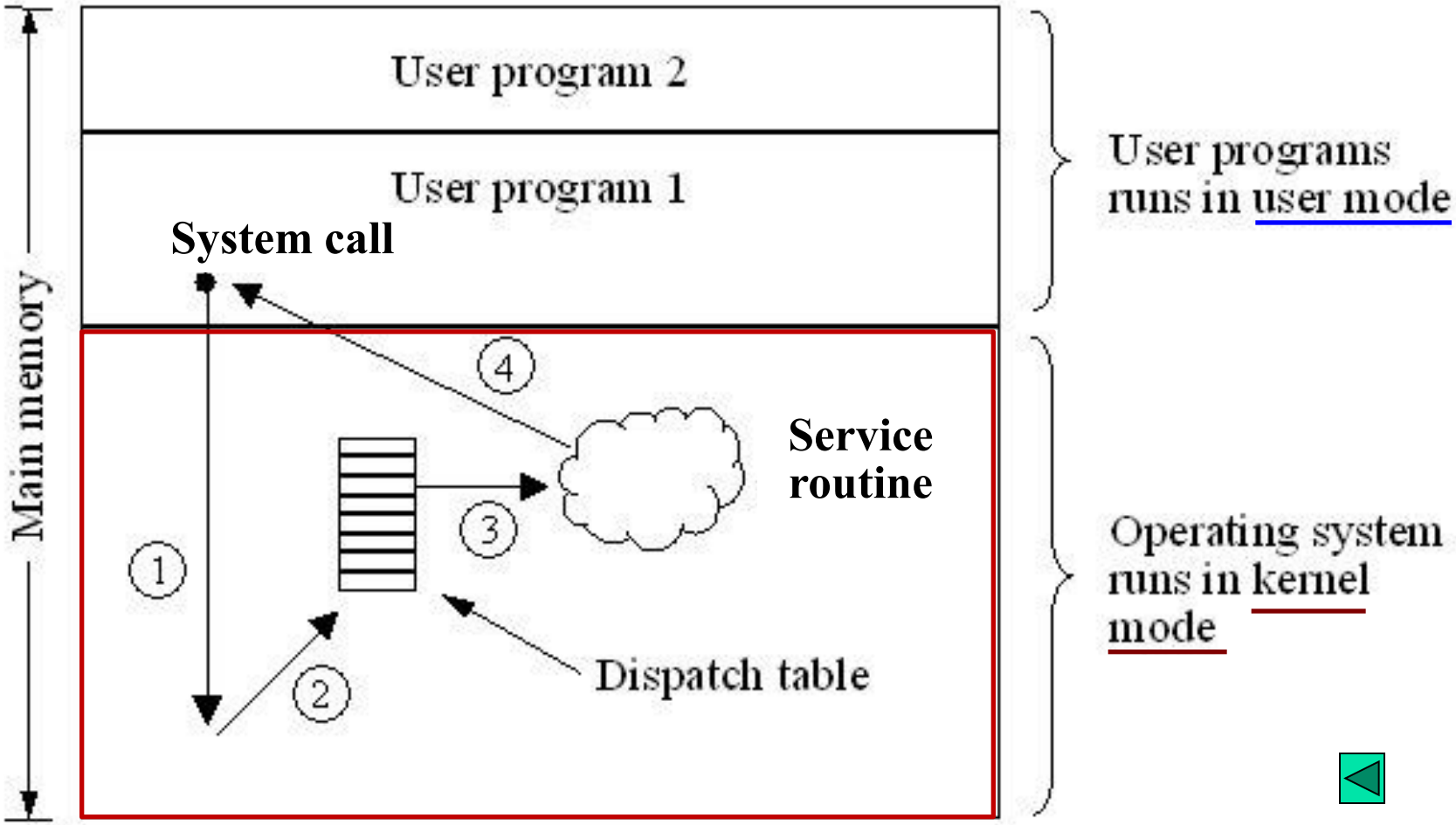


Transition from User to Kernel Mode (Fig. 1.13)

- **System call** changes mode to **kernel mode**, return from call resets it to **user mode**



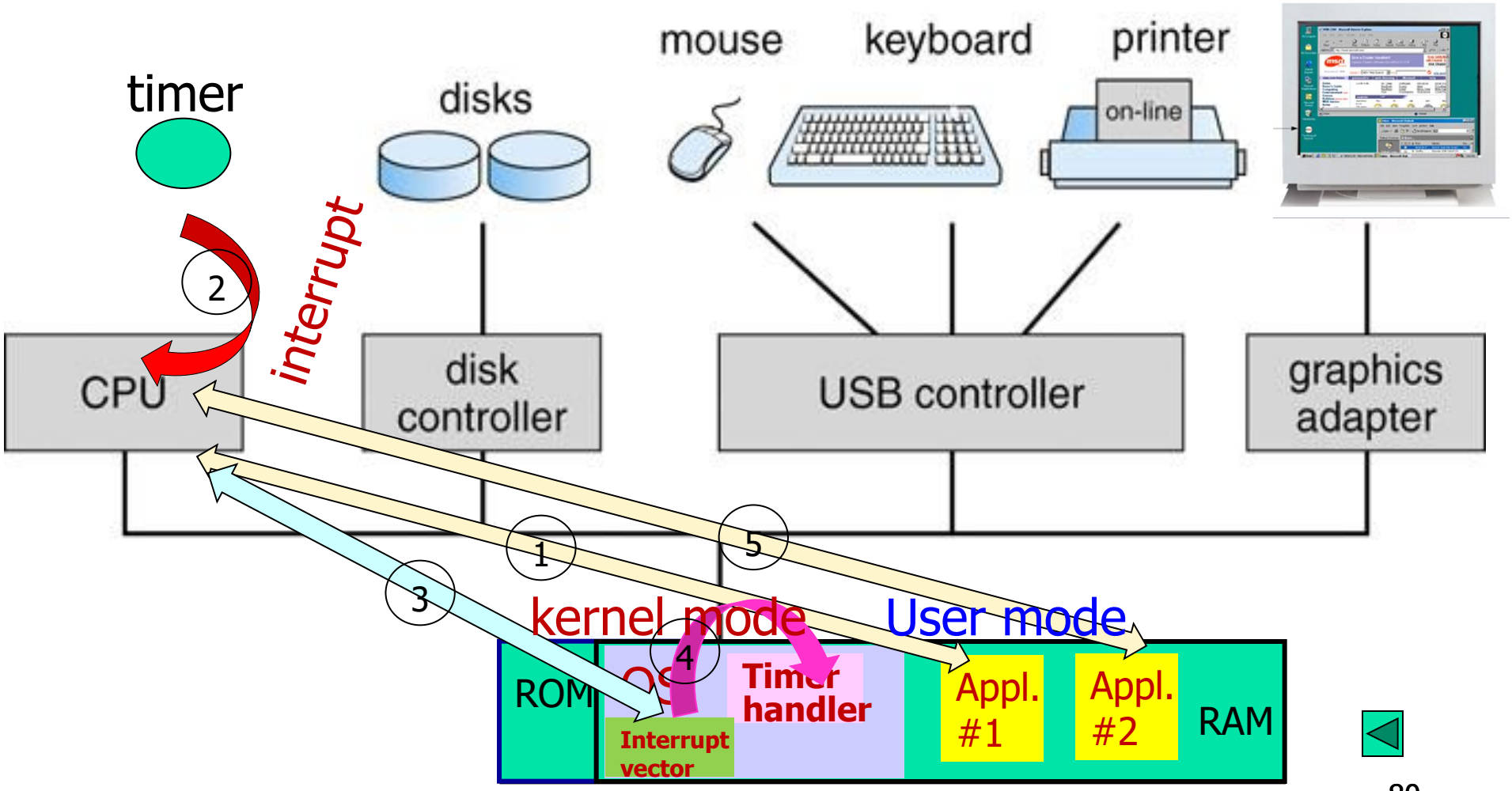
How A System Call Can Be Made



1.4.3 Timer

- **Protect CPU resource from getting stuck by a user program (i.e., in an infinite loop)**
- **To ensure OS maintains control over the CPU, a timer can be set to interrupt the computer after a specified period.**
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time
  - The period may be fixed or variable.
- **Use the timer to prevent a user program from running too long.**
  - Initialize a counter with the allowed execution time

OS is interrupt-driven
timer interrupt processing



1.5 Resource Management

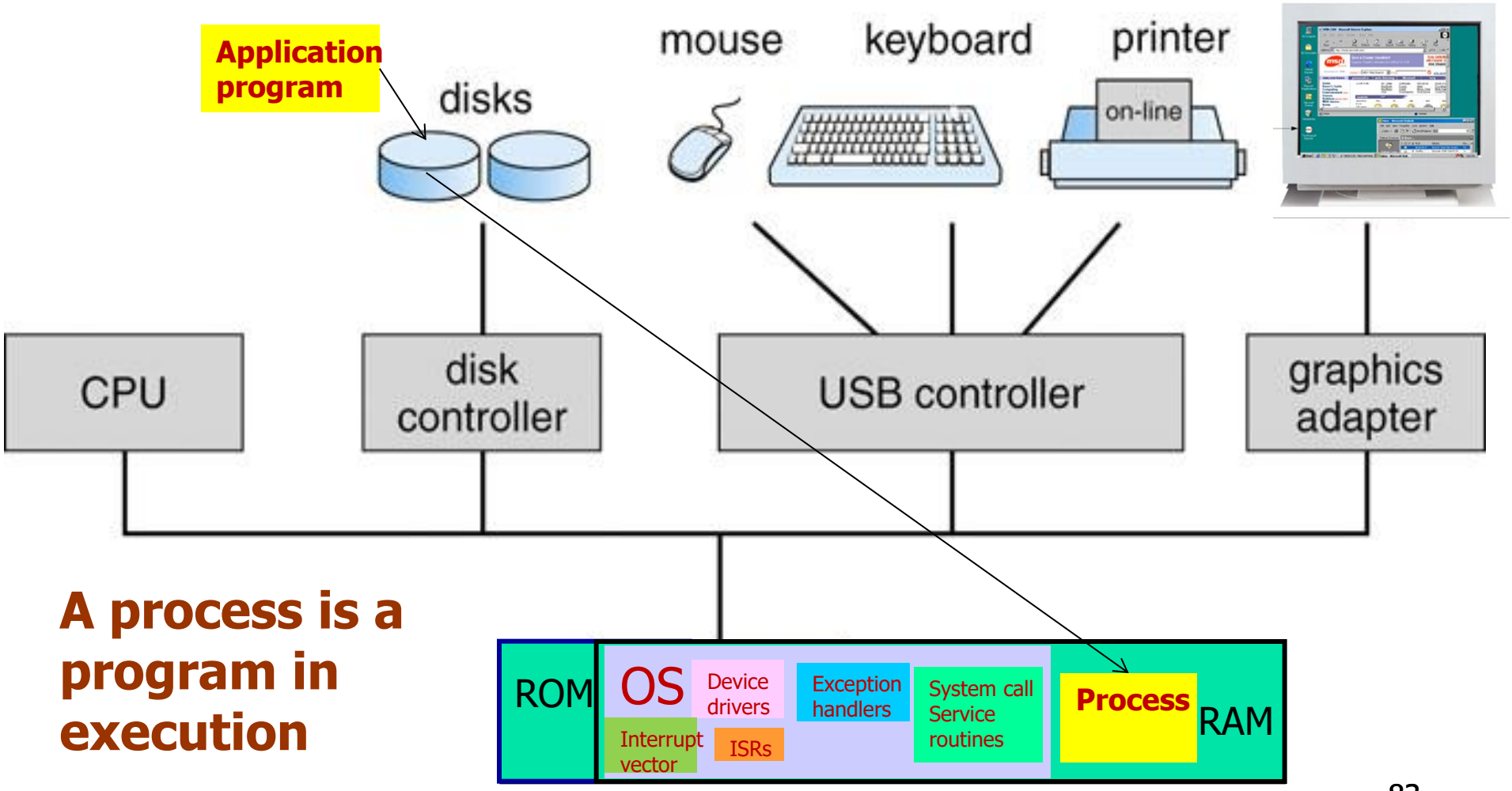
- **Basic OS System Components**
  - 1.5.1 Process Management (ch3-8)
  - 1.5.2 Memory Management (ch9-10)
  - 1.5.3 File-System Management (ch13-15)
  - 1.5.4 Mass-Storage Management (ch11)
  - 1.5.5 Cache Management (ch19)
  - 1.5.6 I/O System Management (ch12)

1.5.1 Process Management

- A ***process*** is a program in execution.
- A program by itself is not a process.
- A ***program*** is a *passive* entity, like the contents of a file stored on disk, whereas a ***process*** is an *active* entity.
- A process needs resources to accomplish its task
  - CPU time, memory, I/O, files
  - Initialization data
- Process termination requires reclamation of any reusable resources. 回收利用

A process is a program in execution

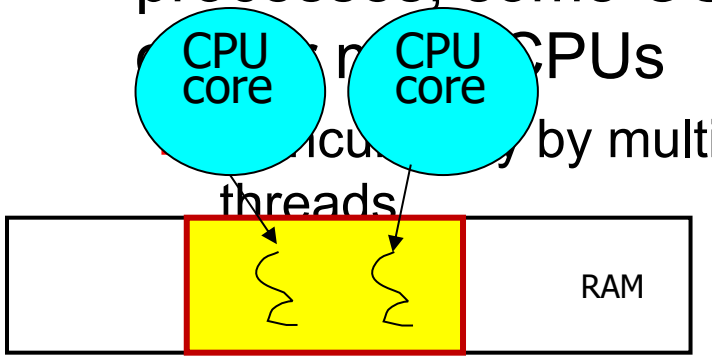
Program vs. Process

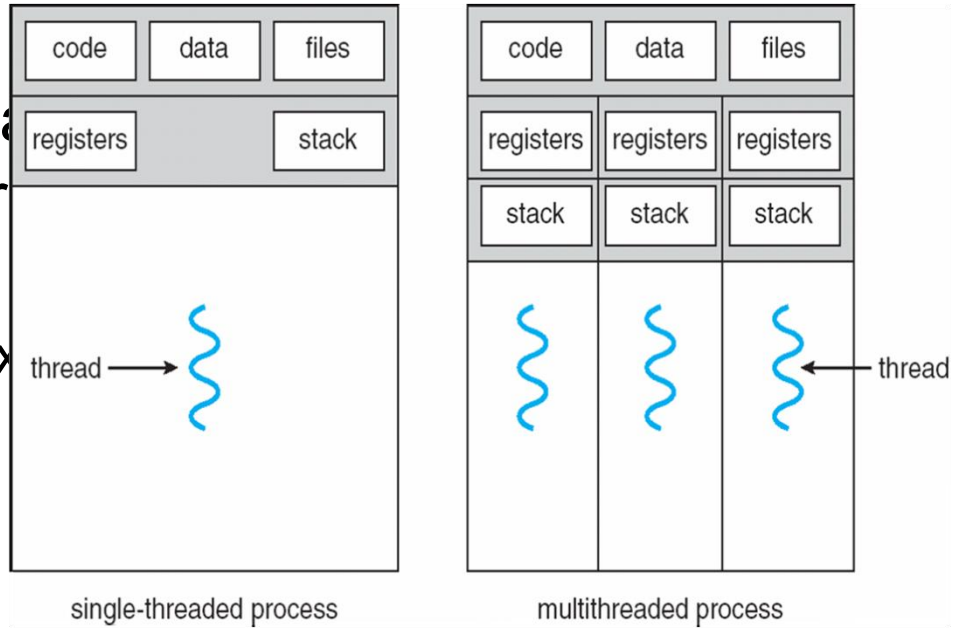


A process is a program in execution

Process Management (Cont.)

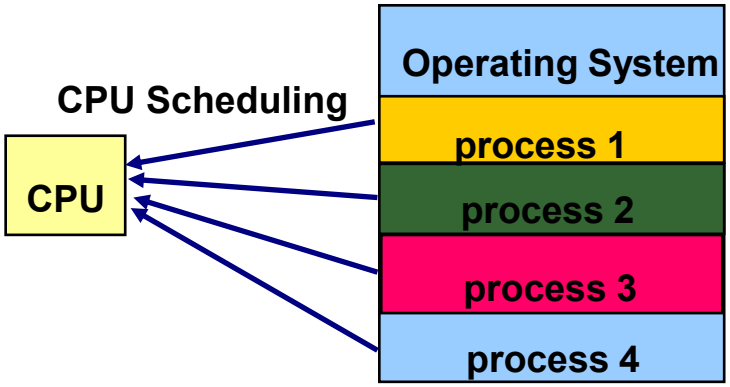
- A **single-threaded process** has one **program counter** specifying the next instruction to execute.
- A **multi-threaded process** has multiple program counters, each pointing to the next instruction to execute for a given thread.
- Typically system has ma
  processes, some OS pr
  CPUs
  by multiplex
  threads

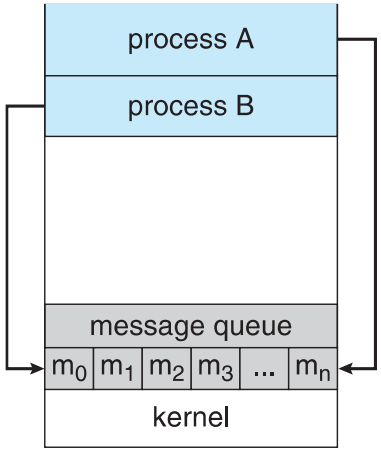




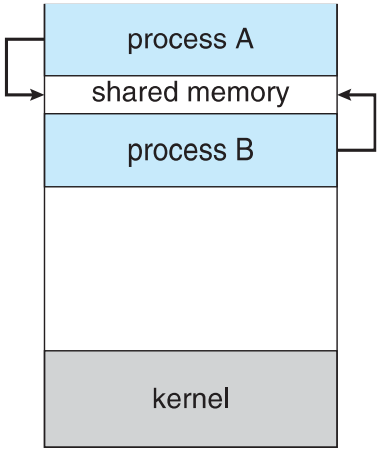
Process Management Activities

- Scheduling processes and threads on the CPUs
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for
  - process synchronization
  - communication





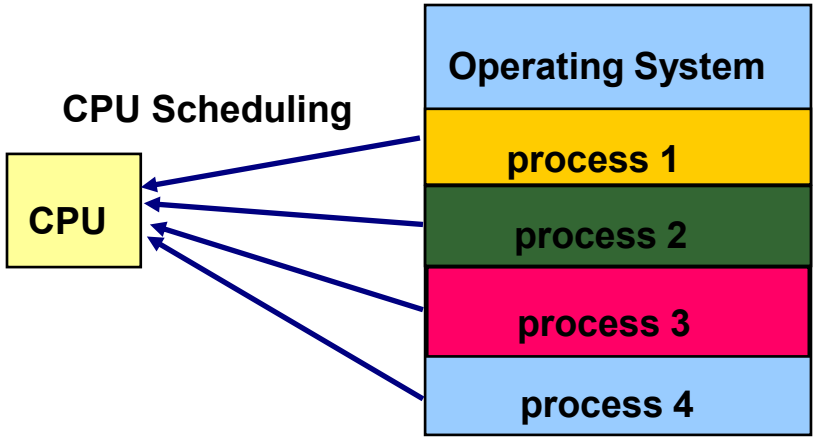
(a)



(b)

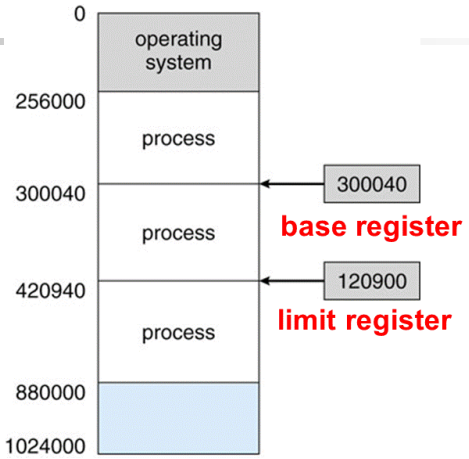
1.5.2 Memory Management

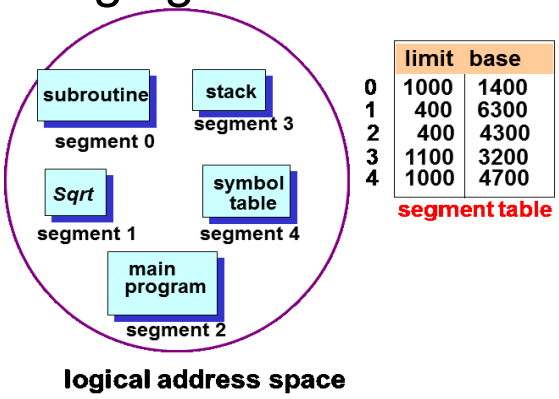
- **To improve CPU utilization and responsive time**
  - keep several programs in memory, creating a need for memory management.
- **In selecting a memory-management scheme for a specific system, we must consider *H/W* design of the system.**
- **Each algorithm requires its own *H/W* support.**



Memory-Management Scheme

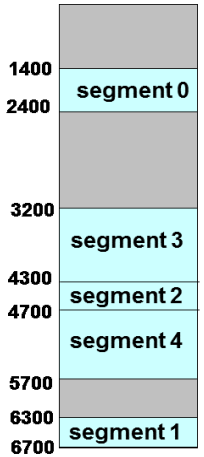
- **Must consider *H/W* design of the system.**
  - Contiguous allocation
  - Segmentation
  - Paging

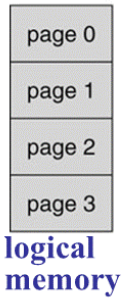


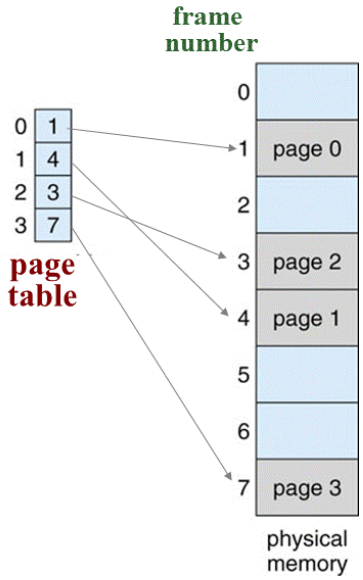


	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment table

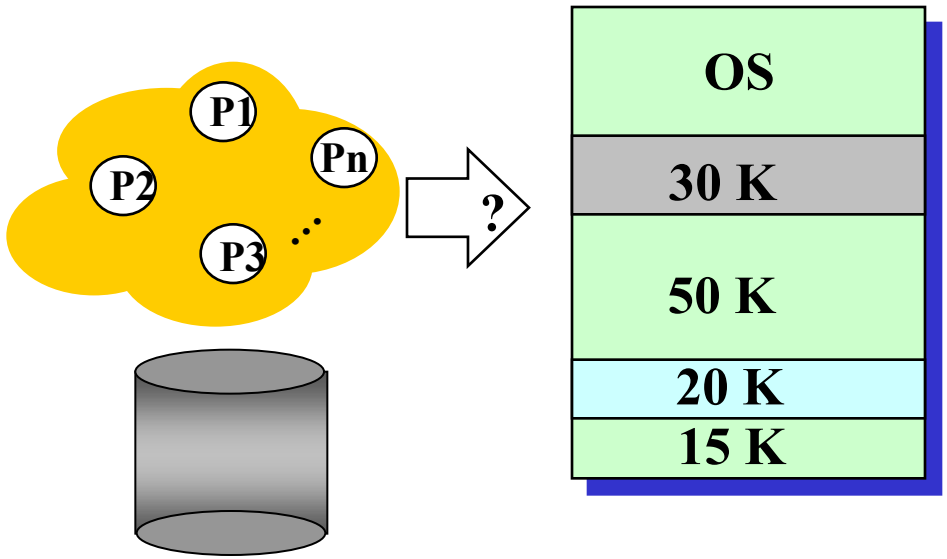






Main-Memory Management Activities

- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes (or parts of processes) and data to move into and out of memory
- Allocating and deallocating memory space as needed

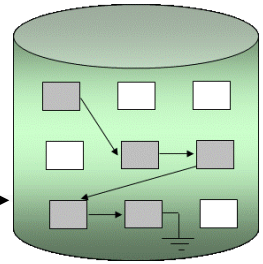


- Placement Problem
- Replacement Problem

1.5.3 File-System Management

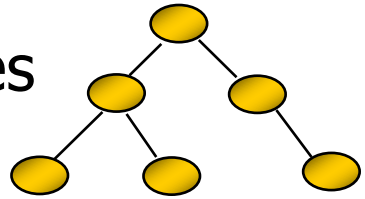
- **OS provides a uniform, logical view of information storage**
  - **abstracts from the physical properties of its storage devices to define a logical storage unit, the *file***
  - **maps files onto physical media and accesses these files via the storage devices**
  - **Each medium is controlled by device (i.e., disk drive)**
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- **Files are normally organized into directories**
- **Access control on most systems to determine who can access what**

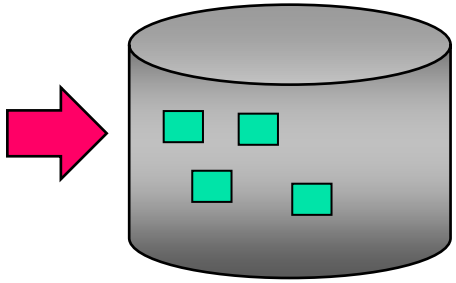
Storage Management



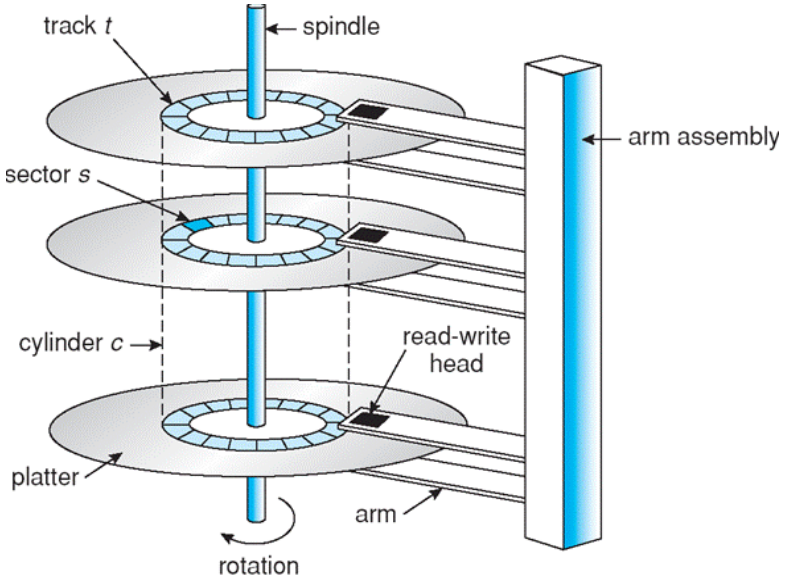
R/W requests

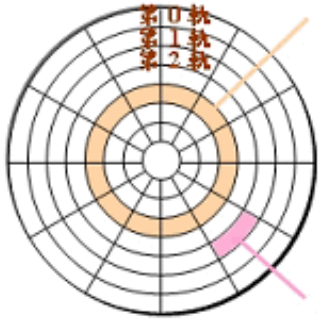
directories
files





blocks



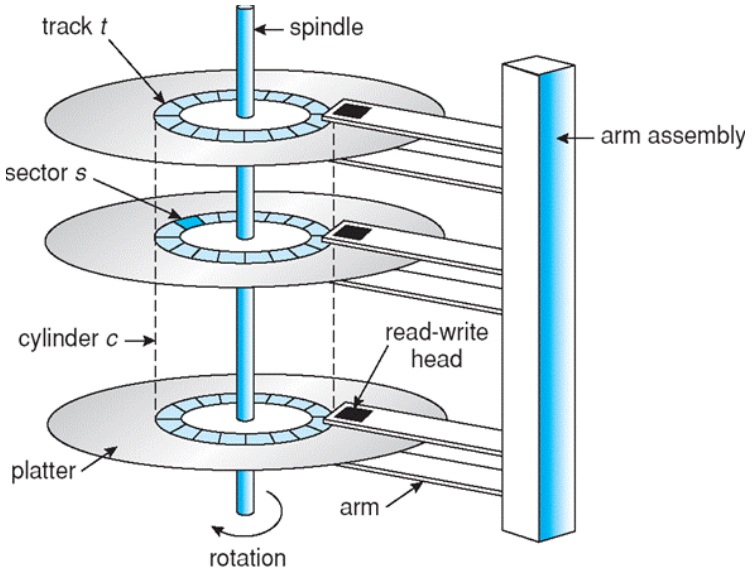


磁軌（track）
磁軌由外至內依序為第 0 軌、第 1 軌、第 2 軌…。

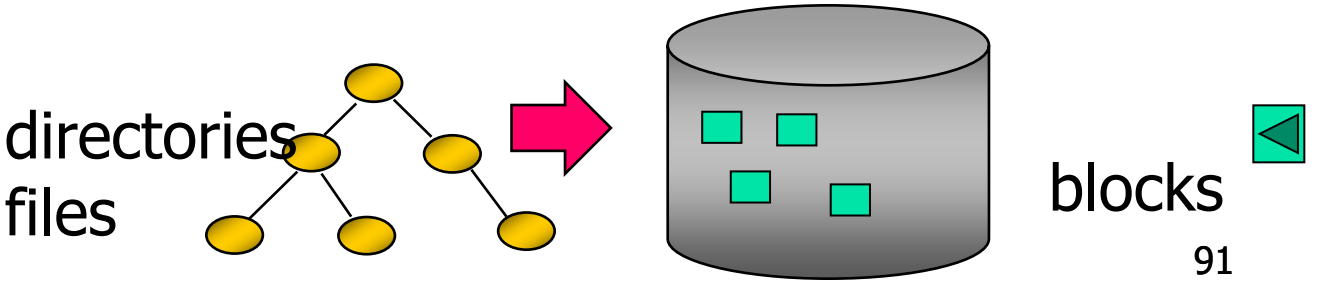
磁區（sector）
每一磁軌再劃分為許多更小的扇形磁區，而磁區是磁碟中基本的儲存單位（通常為 512 bytes）。

File Management Activities

- **Creating and deleting files and directories**
- **Primitives to manipulate files and directories**
- **Mapping files onto mass storage**
- **Backup files on stable (non-volatile) storage media**



Users see files and directories rather than storage blocks



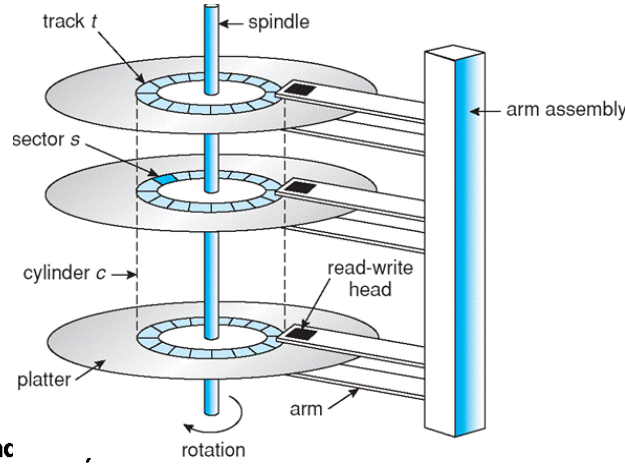
1.5.4 Mass-Storage Management

- **Most modern computer systems use HDDs and NVM devices as the principle on-line storage medium, for both programs and data.**
- **Proper management of secondary storage is of central importance to a computer system.**
- **Entire speed of computer operation may hinge on secondary storage subsystem and its algorithms**
- **Some storage need not be fast**
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications

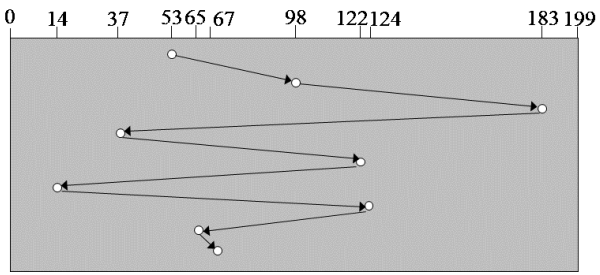
決定於

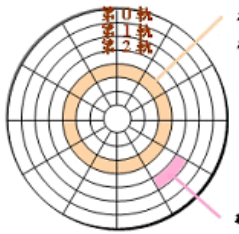
Secondary Storage Management

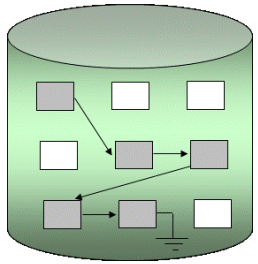
- **OS is responsible for the following activities about disk management:**
  - Mounting / unmounting
  - Free-space management
    - Bit-mapping
    - Link-list
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection



Queue= 98, 183, 37, 122, 14, 124, 65, 67 (cylinc
head starts at 53





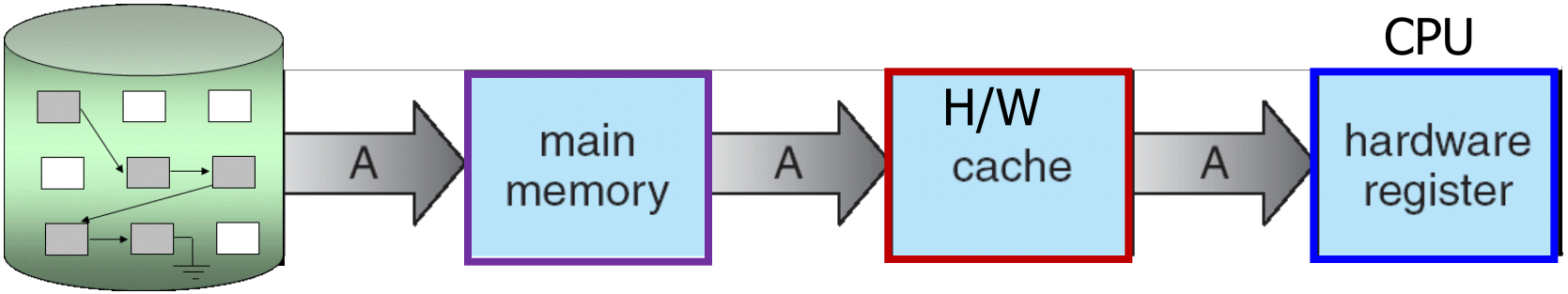


R/W requests

快取

15.5 Caching Management

- **Important principle, performed at many levels in a computer (in hardware, OS, software)**
- **Information in use copied from slower to faster storage temporarily**
- **Faster storage (cache) is checked first to determine if information is there**
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there



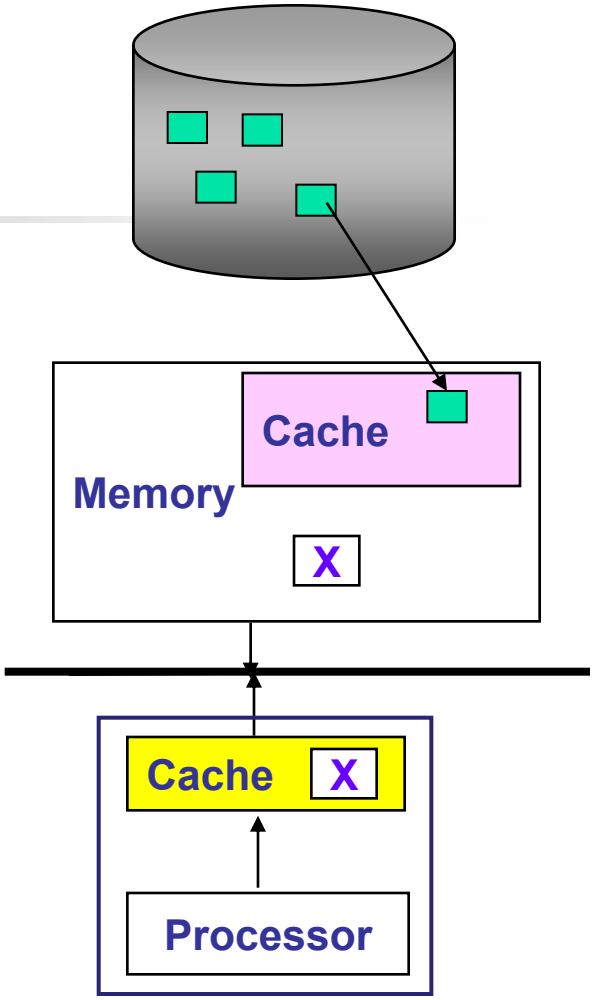
Performance of Various Types of Storage (Fig. 1.14)

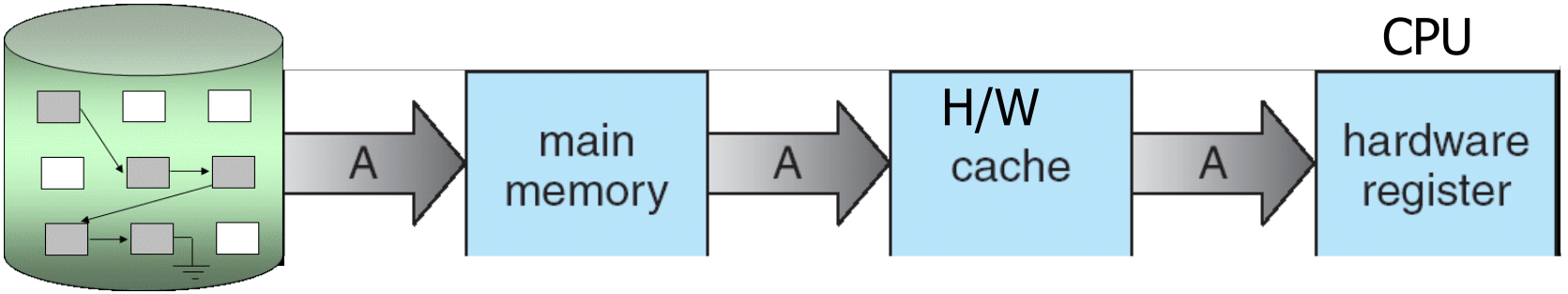
- **Movement between levels of storage hierarchy can be explicit or implicit**

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

Caching

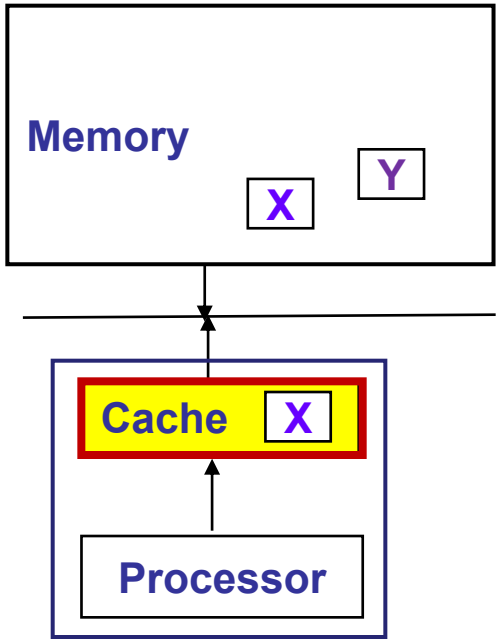
- **Hardware cache**
  - Controlled by H/W, with no OS intervention
- **Software-controlled cache**
  - Controlled by OS

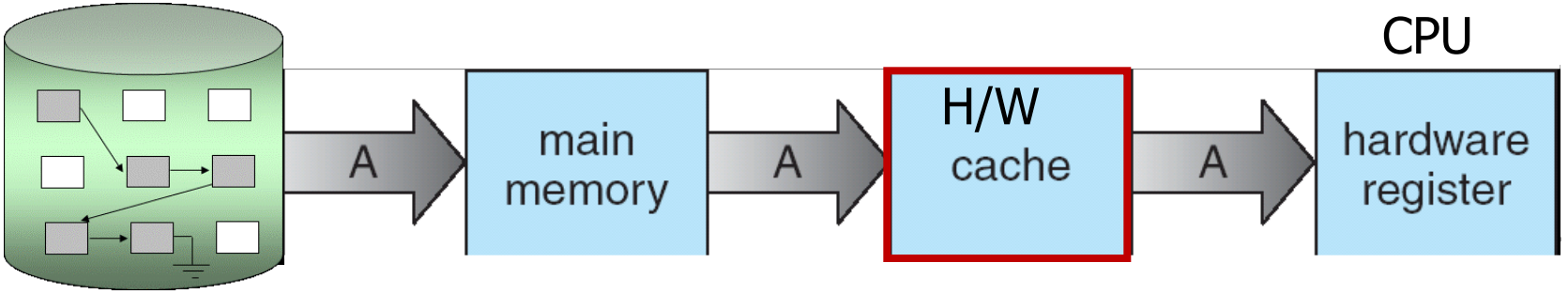




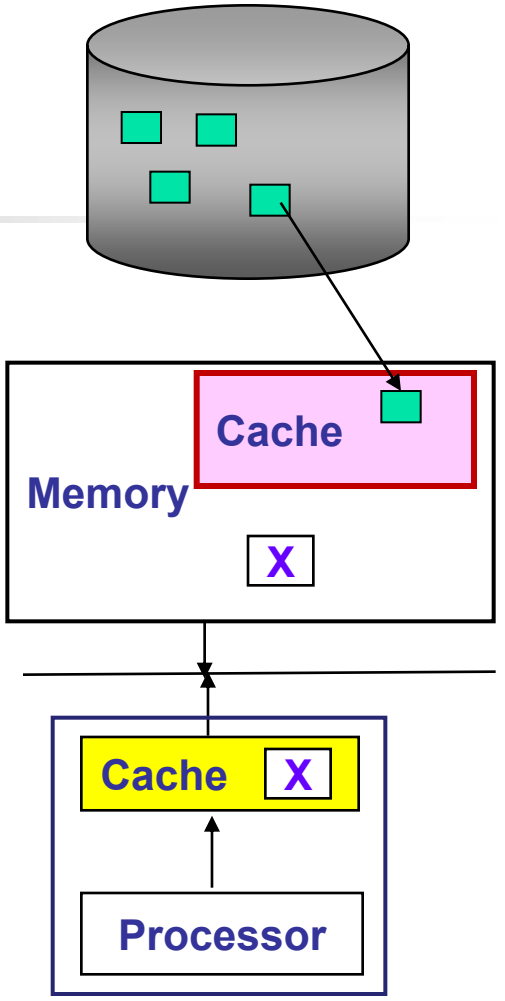
H/W Cache

- **Controlled by H/W, with no OS intervention**
  - Instruction cache and data cache
  - Data transferred from cache to CPU and registers is usually a hardware function, with no OS intervention.

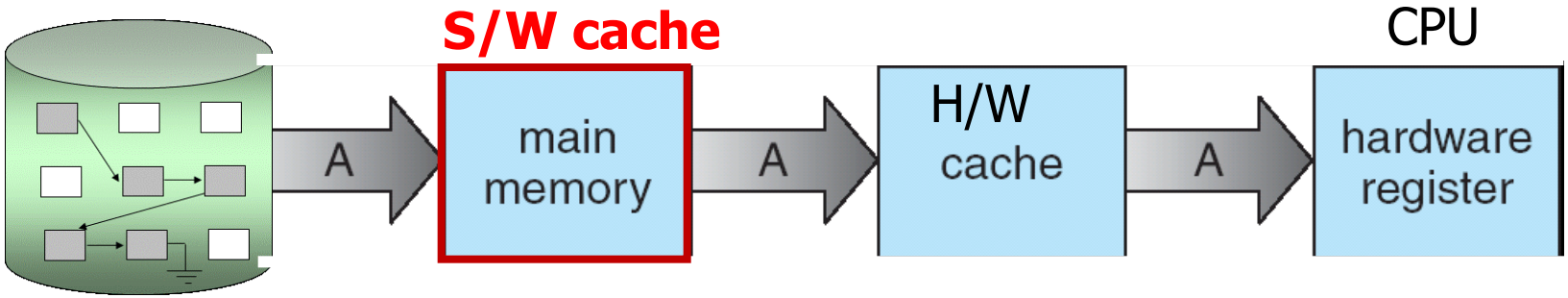




S/W Cache

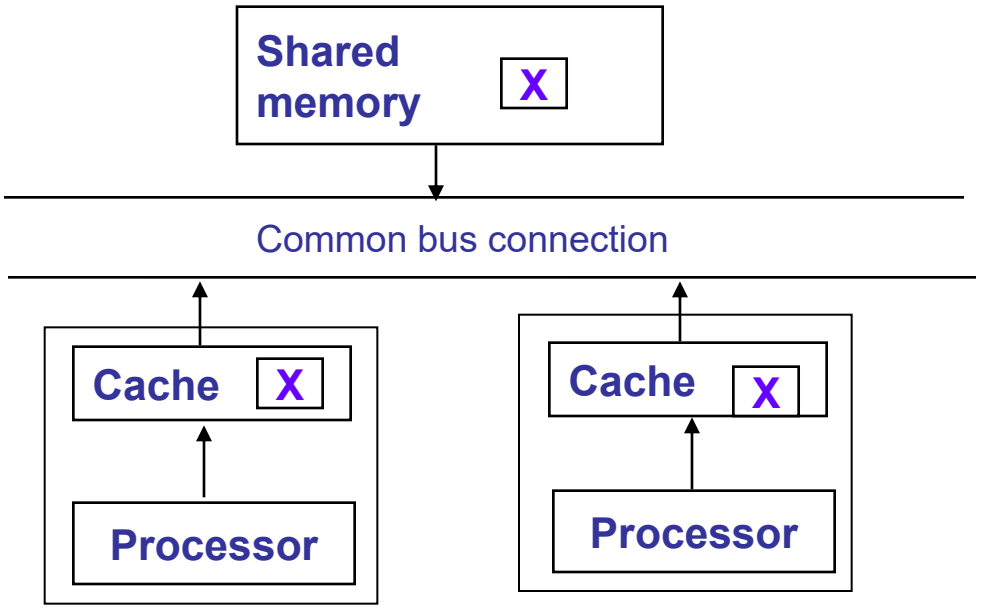


- **Controlled by OS**
  - ***Cache management*** (cache size and replacement policy) is an important design problem.
- **Main memory can be viewed as a fast *cache* for secondary storage.**
  - **OS** may maintain **a cache of file-system data** in main memory
  - Transfer of data from disk to memory is usually controlled by OS



Cache Coherency in Multiprocessor Environment

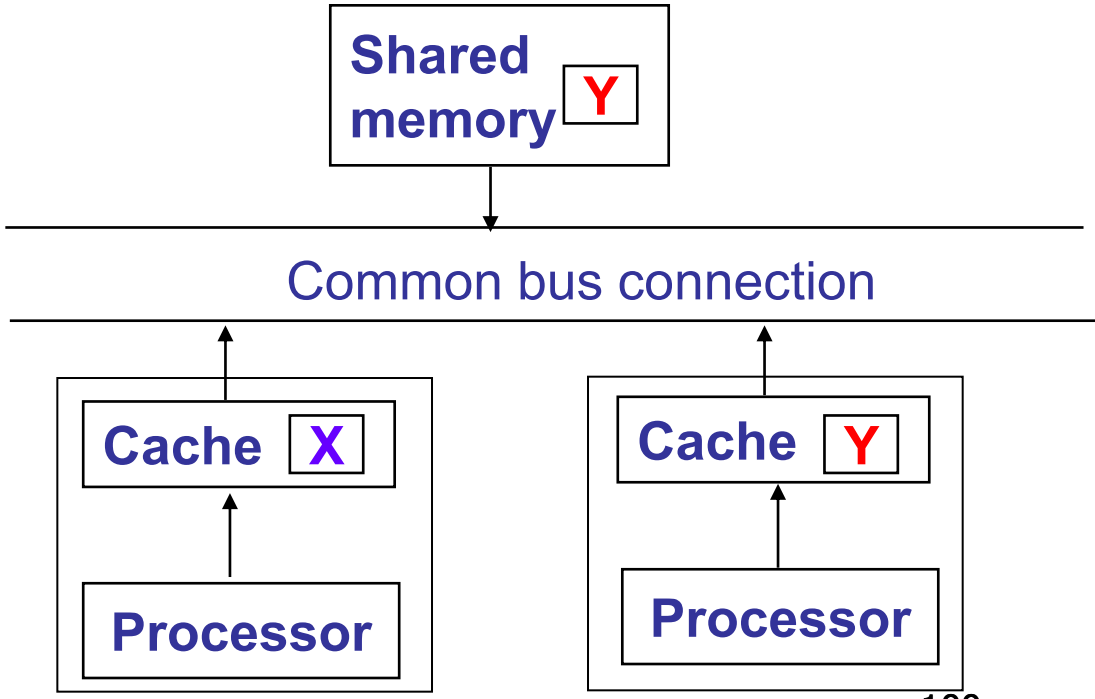
- In a multiprocessor environment, the situation becomes more complicated, the CPU also contains a local cache.
- Problems occur when processors have common elements in their caches.



Cache Coherency 統一;連貫性

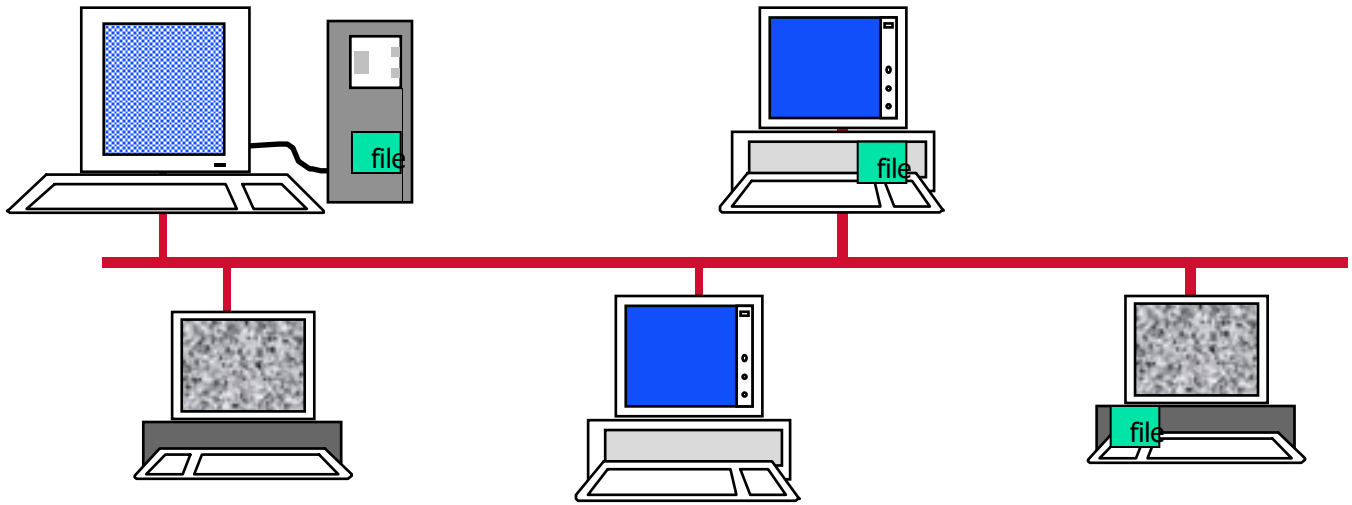
- **Must make sure that an update to the value of A in one cache is immediately reflected in all other caches where A resides.**
  - Is usually a hardware issue

- **Simple solution:**
  - inform the other processors that their cache copies are invalid when shared memory changes. They need to rerouted to shared memory instead

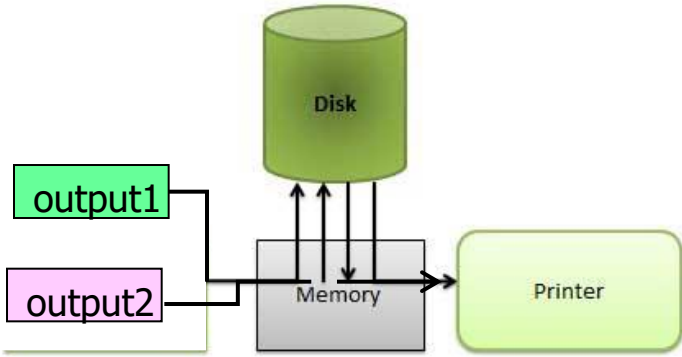


Cache Coherency in Distributed Environment

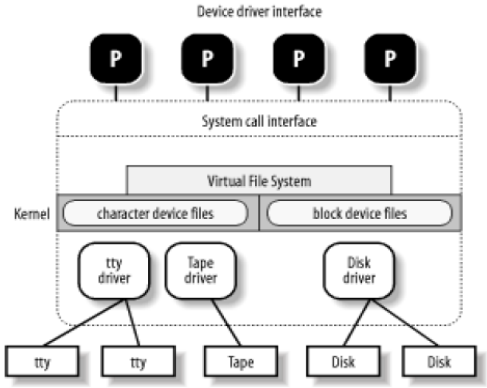
- The situation becomes even more complex, several copies (replicas) of the same file can be kept on different computers.
- Must ensure that, when a replica is updated in one place, then all other replicas are brought up to date as soon as possible.



1.5.6 I/O System Management



- **Purpose:** 奇特, 特性
  - Hide the peculiarities of specific H/W devices from the user.
- **I/O subsystem consists of**
  - A memory-management component that includes **buffering** (storing data temporarily while it is being transferred), **caching** (storing parts of data in faster storage for performance), **spooling** (one way OS can coordinate concurrent output, a spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams)
  - A general **device-driver interface**
  - **Drivers** for specific H/W devices



1.6 Security and Protection

- ***Protection*** **refers to any mechanism for controlling the access of processes or users to the resources defined by a computer system.**
  - Authorization of resources: CPU, memory, disk, files,...
  - Discussed in Chapter 17
- **The protection mechanism must provide means to**
  - specify the controls to be impose
  - enforce the controls
  - distinguish between ***authorized*** and ***unauthorized*** usage
- **Security – to defend a system from external and internal attacks**
  - denial-of-service, worms, viruses, identity theft, theft of service,…
  - Discussed in Chapter 16

安全防護

Security and Protection (Cont.)

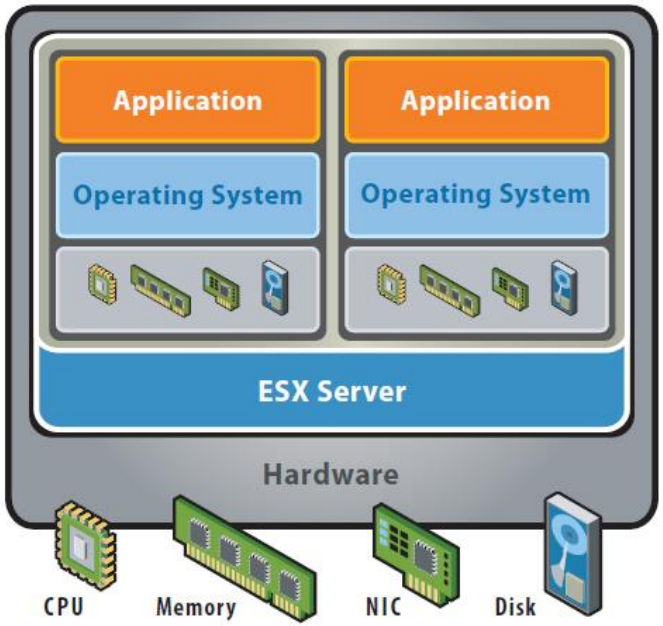
- **Protection and security require the system to be able to distinguish among all its users, to determine who can do what**
  - User identities (user IDs, security IDs) include name and associated number, one per user
  - User ID then associated with all processes of that user to determine access control
  - Group identifier (group ID) allows set of users to be defined and controls managed, then also associated with each process, thread
  - Privilege escalation allows user to change to effective ID with more rights 逐步擴大

1.7 Virtualization

- **Virtualization** is a technology that allows us to abstract the hardware of a single computer (CPU, memory, disk drives, network interface cards…) into several different execution environments,

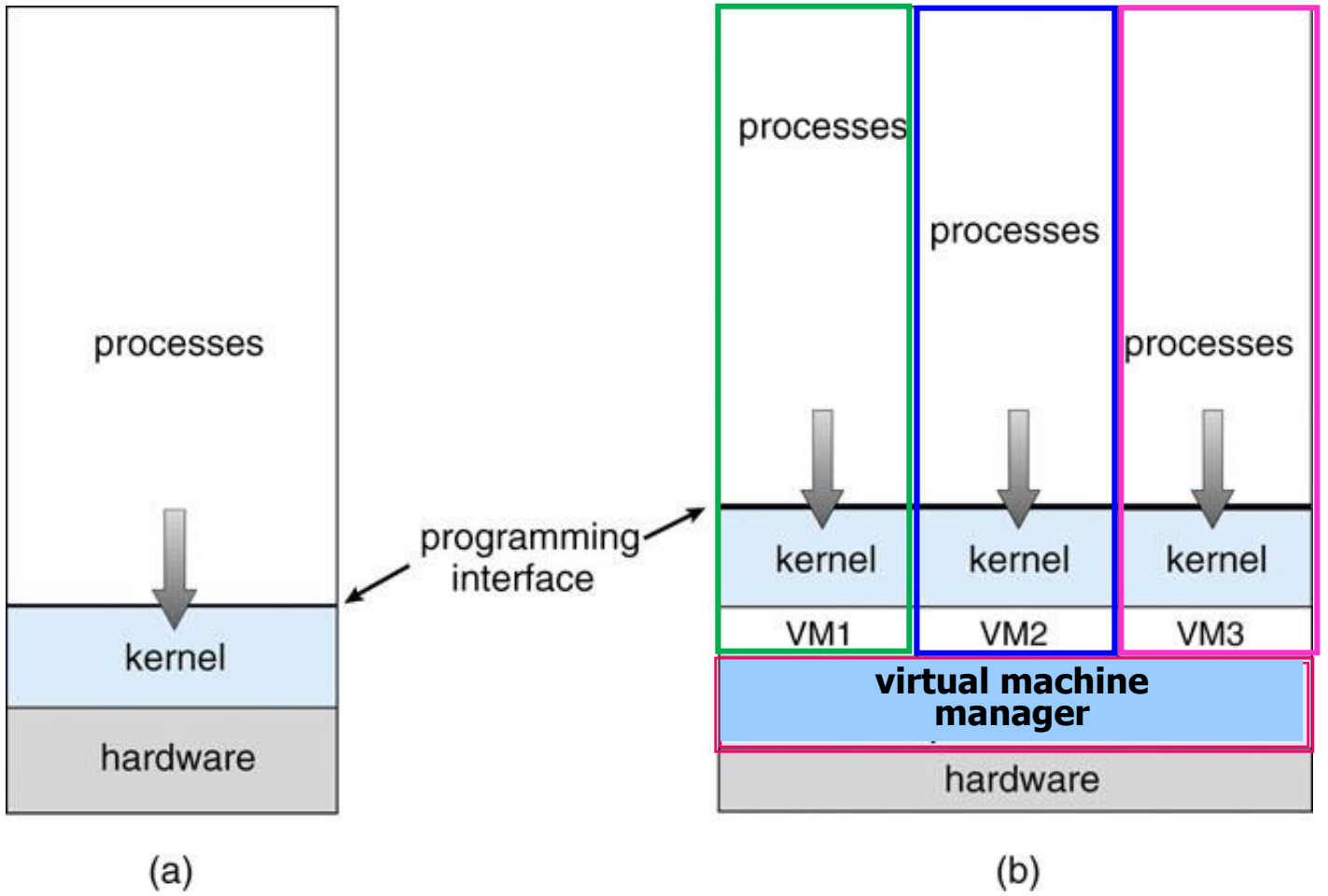
  thereby creating the illusion that each separate environment is running on its own private computer.

- A single physical machine can run multiple OSes concurrently, each in its own virtual machine.



VMware ESX Server virtualizes server storage and networking, allowing multiple applications to run in virtual machines on the same physical server.

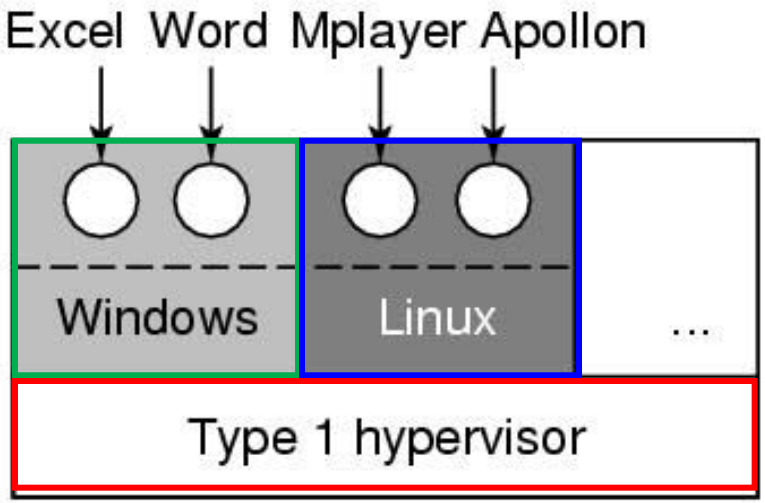
A computer running with/without VMs



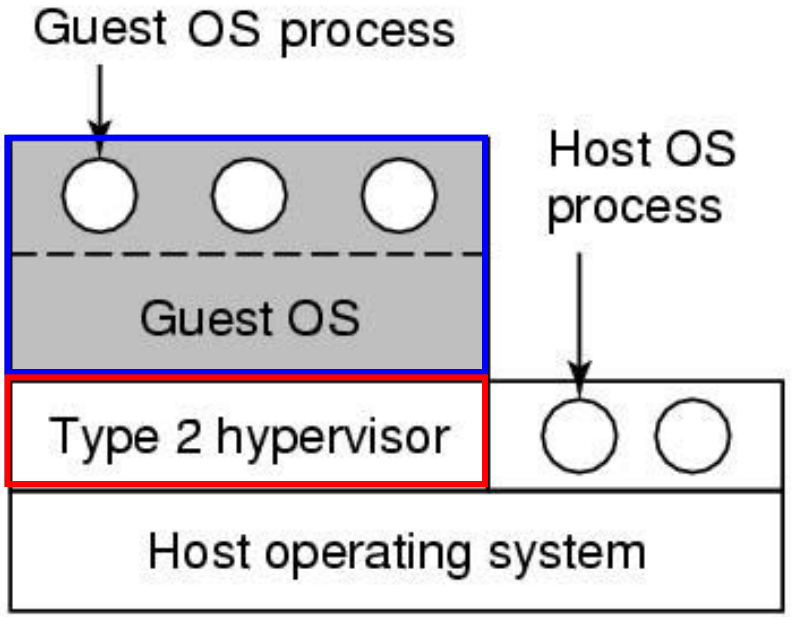
A single OS

Three virtual machines

Types of Hypervisors (Virtual Machine Manager, VMM)



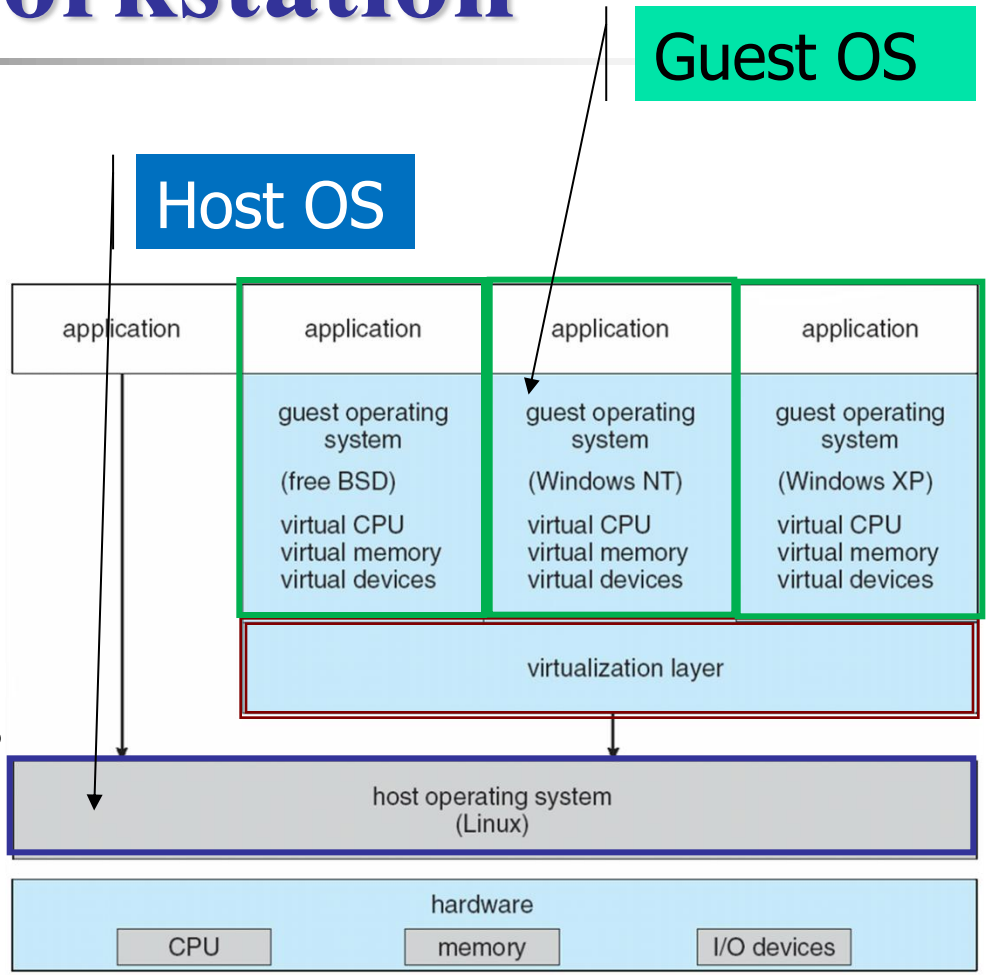
Examples: VMware ESX
Citrix XenServer



Examples: VMware Workstation
VirtualBox

VMware Workstation

- VMM runs guest OSes, manages their resource use, and protects each guest from the others
- Runs as an application on a host OS such as Windows and Linux, and allows this host system to currently run several different guest OSes as independent virtual machines



Virtualization (Cont.)

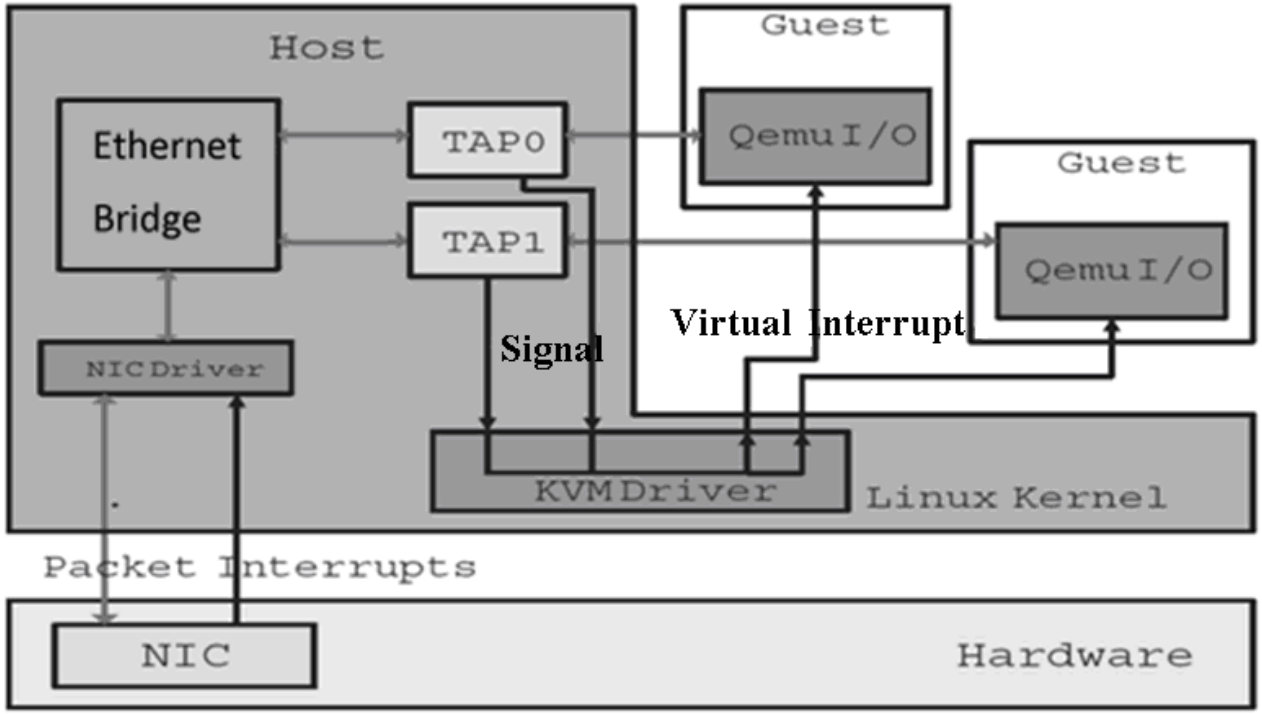
- **Emulation , which involves simulating computer hardware in software, is typically used when source CPU type is different from target type**
  - e.g., when Apple switched from IBM Power CPU to Intel x86 CPU for its desktop and laptop computers, an emulator called “Rosetta” allows applications compiled for IBM CPU to run on Intel CPU
  - Generally slowest method

Ex: ***Android emulator***

The Android emulator mimics all of the H/W & S/W features of a typical mobile device, except that it cannot place actual phone calls.

The emulator lets you prototype, develop and test Android applications without using a physical device.

Linux Kernel-based Virtual Machine (KVM)

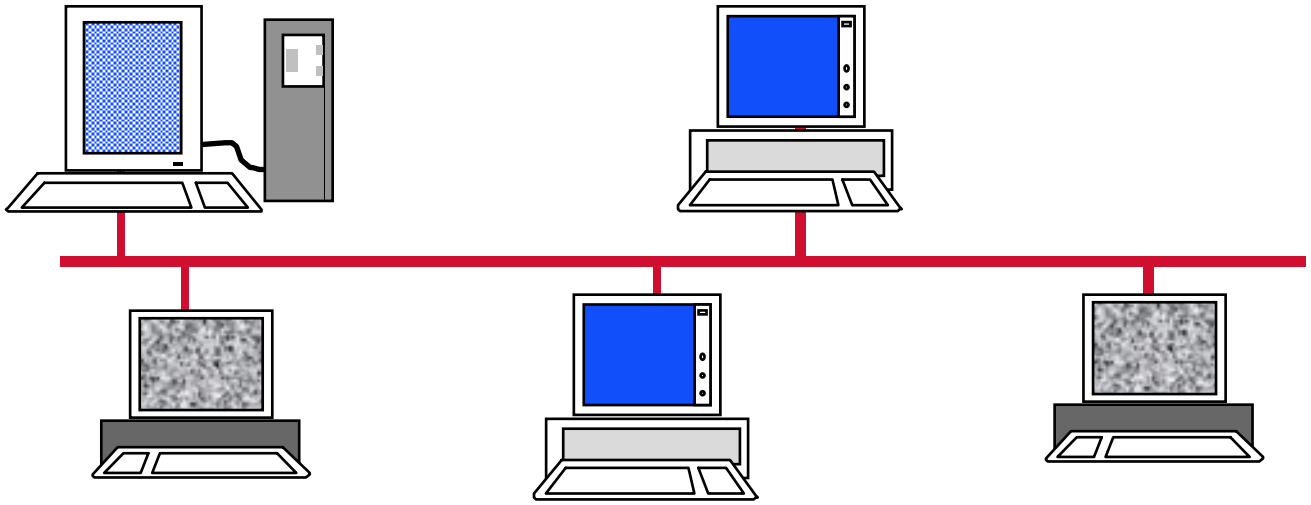


Virtualization (Cont.)

- **Use cases involve laptops and desktops running multiple OSes for exploration or running applications written for OSes other than the native host**
  - e.g., Apple laptop running MacOS host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - Developing, testing, and debugging applications without having multiple systems
  - Executing and managing compute environments within data centers
- **VMM can run natively, in which case they are also the host**
  - VMMs like VMware ESX and Citrix XenServer
- **Detailed in Chapter 18**

1.8 Distributed Systems

- A distributed system is a collection of physically separate, possibly heterogeneous, computer systems networked together to provide users with access to the various resources that the system maintains.



Distributed Systems (Cont.)

- **Access to a shared resource increases computational speed, functionality, data availability, and reliability.**
- **Network protocol, network adapter, device driver**
- **A network operating system provides features between systems across network**
  - Communication scheme allows processes on different computers to exchange messages
- **A distributed operating system provides a less autonomous environment**

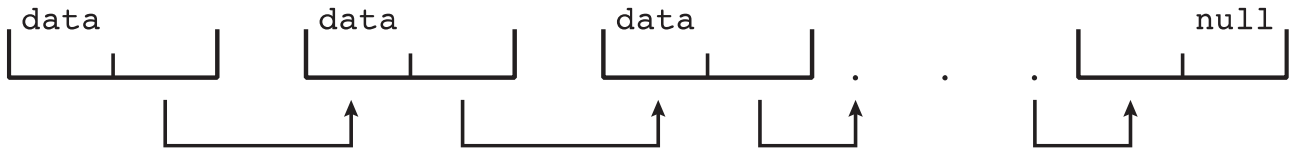
自主的

  - Different computers communicate closely enough to provide the Illusion that only a single OS controls the network
- **CH19**

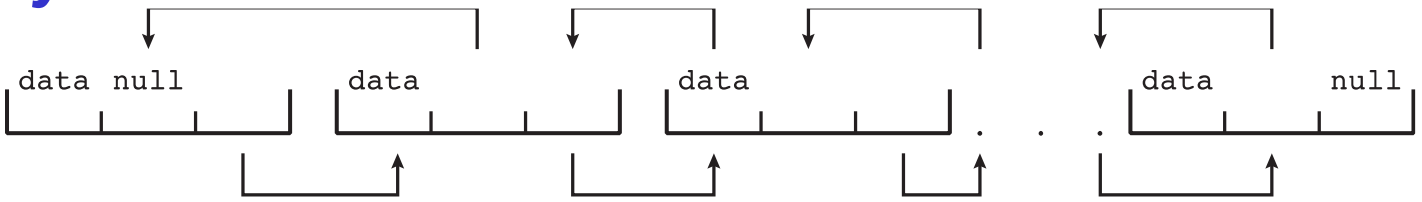
process, data/file migration

1.9 Kernel Data Structures

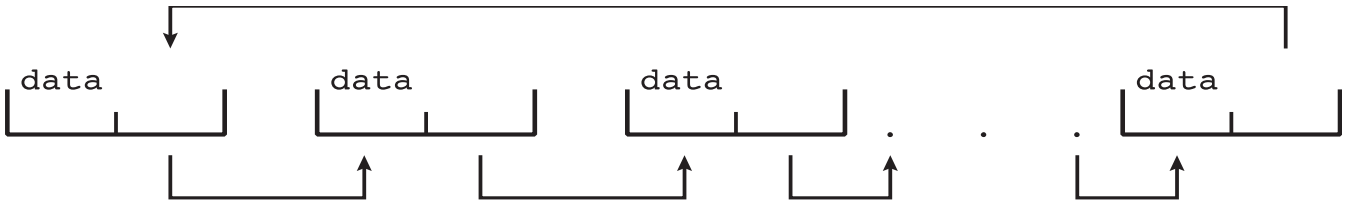
- **Fundamental data structures used in OS**
  - Central to OS implementation
- **Arrays, lists, stacks, and queues**
- ***Singly linked list***

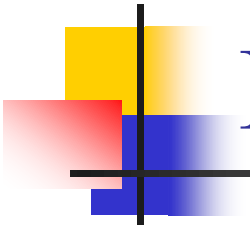


- ***Doubly linked list***



- ***Circular linked list***



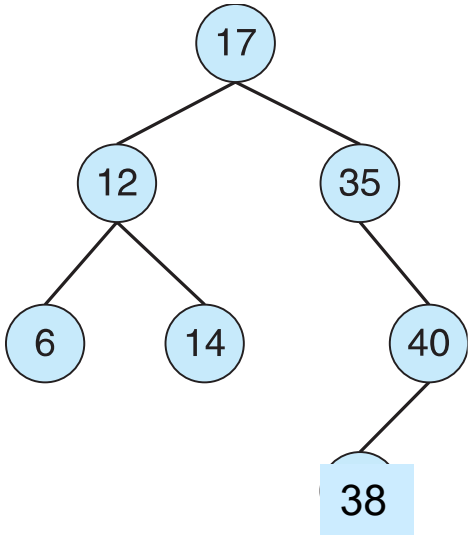


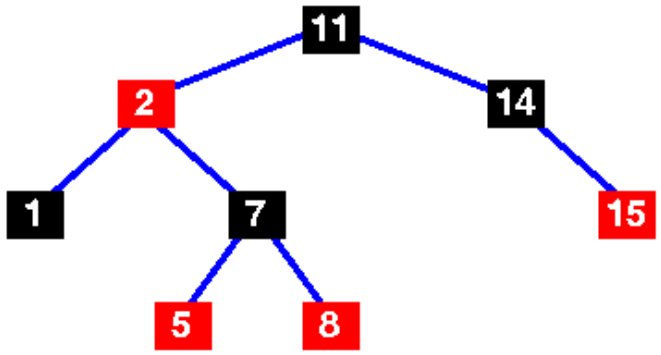
Kernel Data Structures (Cont.)

- **Stack**
  - A sequentially ordered data structure that uses the last in, first out)(LIFO) principle for adding/removing items.
  - Push / pop operations
- **Queue**
  - A sequentially ordered data structure that uses the first in, first out (FIFO) principle
- **Tree**
  - A data structure that can be used to represent data hierarchically.

Kernel Data Structures (Cont.)

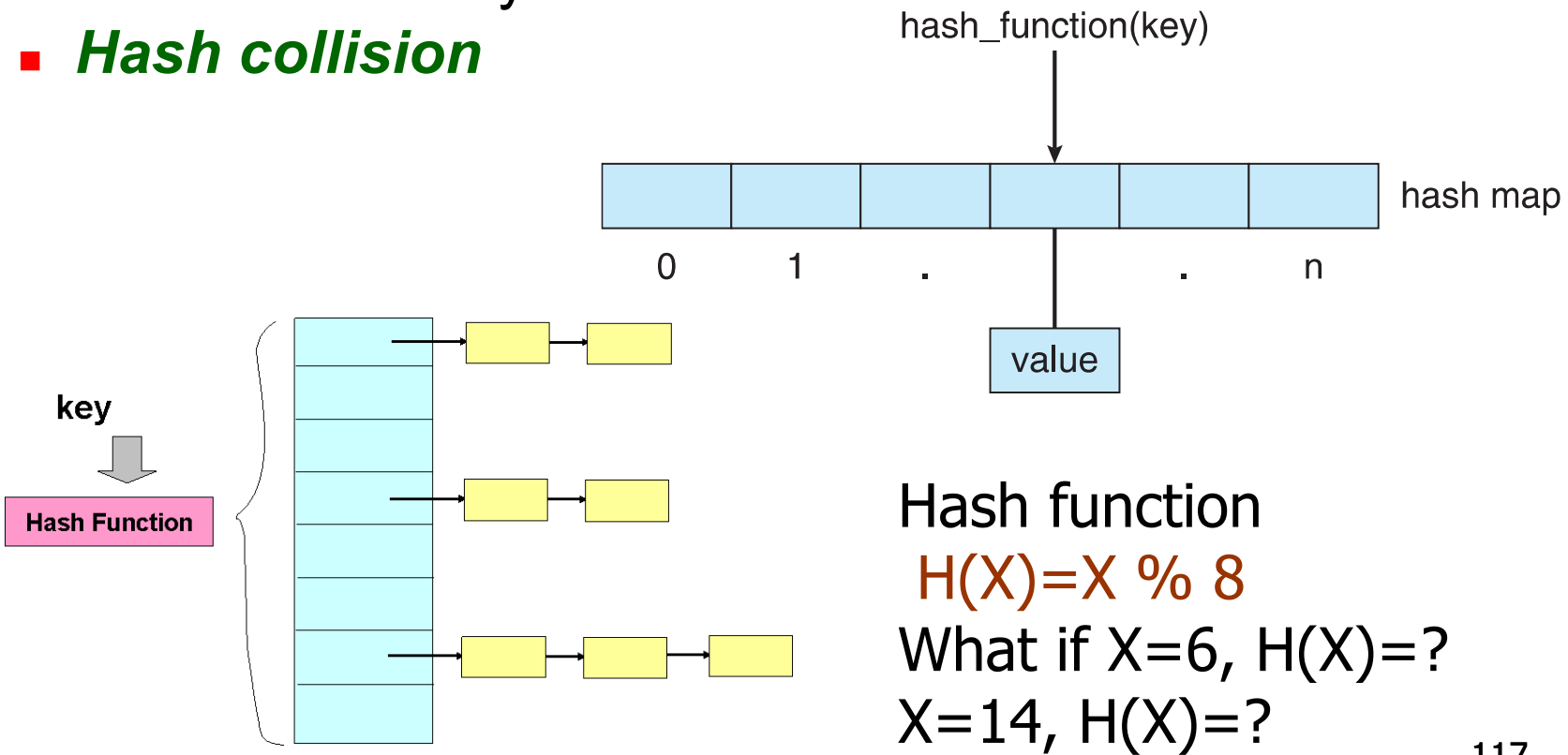
- **Binary search tree**
  left_child <= right_child
  - Search worst-case performance is *O(n)*
- **Balanced binary search tree**
  - At most *lg n* levels
  - worst-case performance is *O(lg n)*
  - Linux uses *red-black tree*
    in CPU scheduling





Kernel Data Structures (Cont.)

- **One use of a hash function is to implement a hash map**
  - Search worst-case performance is *O(1)*
  - Used extensively in OS
  - ***Hash collision***



Hash function
H(X)=X % 8
What if X=6, H(X)=?
X=14, H(X)=?

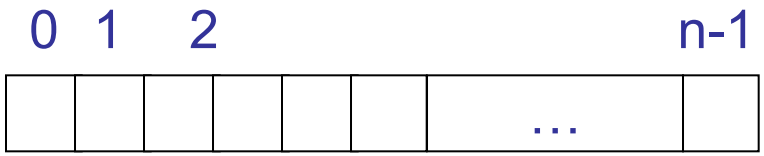
Kernel Data Structures (Cont.)

- **Bitmap**
  - A string of *n* binary digits representing the status of *n* items
  - e.g. n resources

0 0 1 0 1 1 1 0 1

resources 0, 1, 3, 7 are available

  - e.g. bitmap (*n* disk blocks)



$$bit[i] = \begin{cases} 0 \Rightarrow block[i] \text{ free} \\ 1 \Rightarrow block[i] \text{ occupied} \end{cases}$$

1.10 Computing Environments

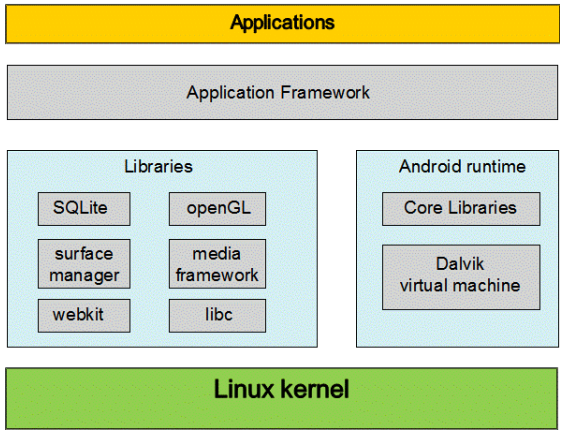
- 1.10.1 Traditional Computing 
- 1.10.2 Mobile Computing 
- 1.10.3 Client-Server Computing 
- 1.10.4 Peer-to-Peer Computing 
- 1.10.5 Cloud Computing 
- 1.10.6 Real-Time Embedded Systems 

1.10.1 Traditional Computing

- Stand-alone general purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- Portals provide web access to internal systems
- Network computers (thin clients) are like Web terminals
- Mobile computers interconnect via wireless networks
- Networking becoming ubiquitous – even home systems use firewalls to protect home computers from Internet attacks

1.10.2 Mobile Computing

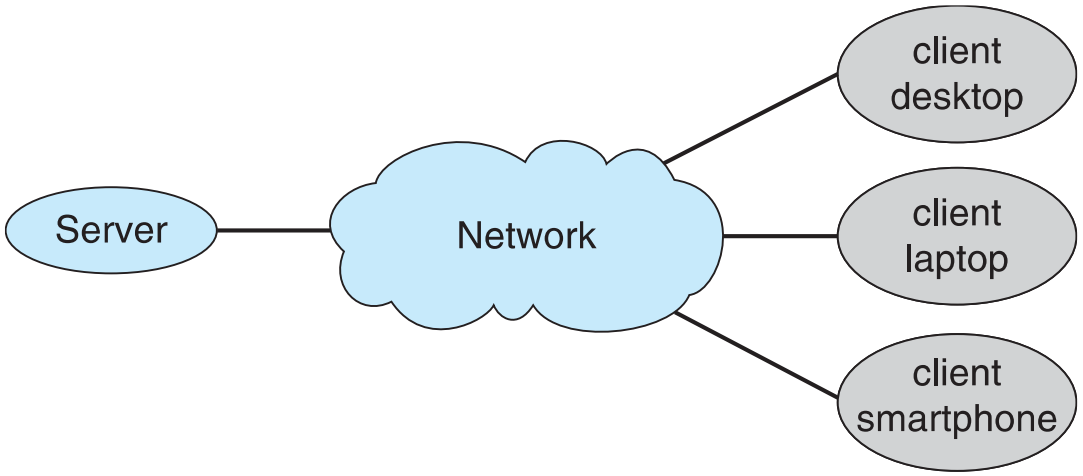
- **Mobile computing**
  - refers to computing on handheld smartphones, tablet computers, etc
- **What is the functional difference between them and a "traditional" laptop?**
- **Extra feature – more OS features (GPS, gyroscope)**
- **Allows new types of apps like *augmented reality***
- **Use IEEE 802.11 wireless, or cellular data networks for connectivity**
- **Leaders are**
  - Apple iOS
  - Google Android



1.10.3 Client-Server Computing

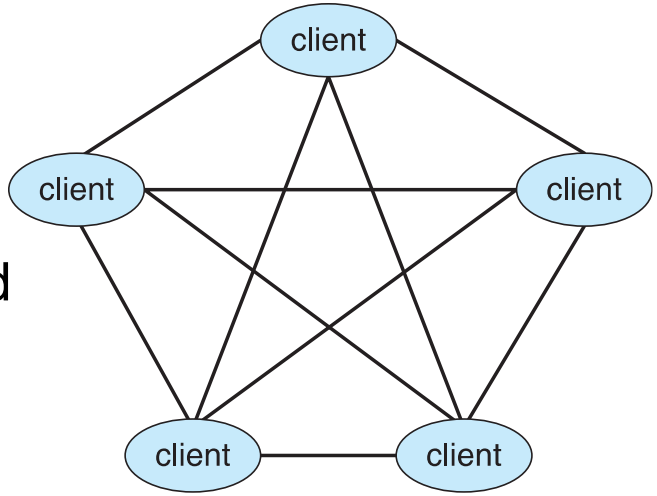
Many systems now have **servers**, responding to requests generated by **clients**

- **Compute-server** system provides an interface to client to request services (i.e., database)
- **File-server** system provides a file-system interface for clients to store and retrieve files



1.10.4 Peer-to-Peer Computing

- **Another structure for a distributed system**
- **P2P does not distinguish clients and servers**
  - Instead, all nodes are considered peers
  - Each may act as client, server or both
  - Node must join **P2P network**
    - Registers its service with a centralized lookup service on network, or
    - Broadcast a request for service and respond to requests for service via ***discovery protocol*** that allows peers to discover services provided by other peers in the network
  - e.g., *Napster* and *Gnutella,* **Voice over IP** *(***VoIP***)* such as **Skype**



1.10.5 Cloud Computing

雲端運算是一種基於網際網路的運算方式，使用者透過網際網路來隨選租用遠端資訊科技專業廠商所提供的資源、平台與應用系統。

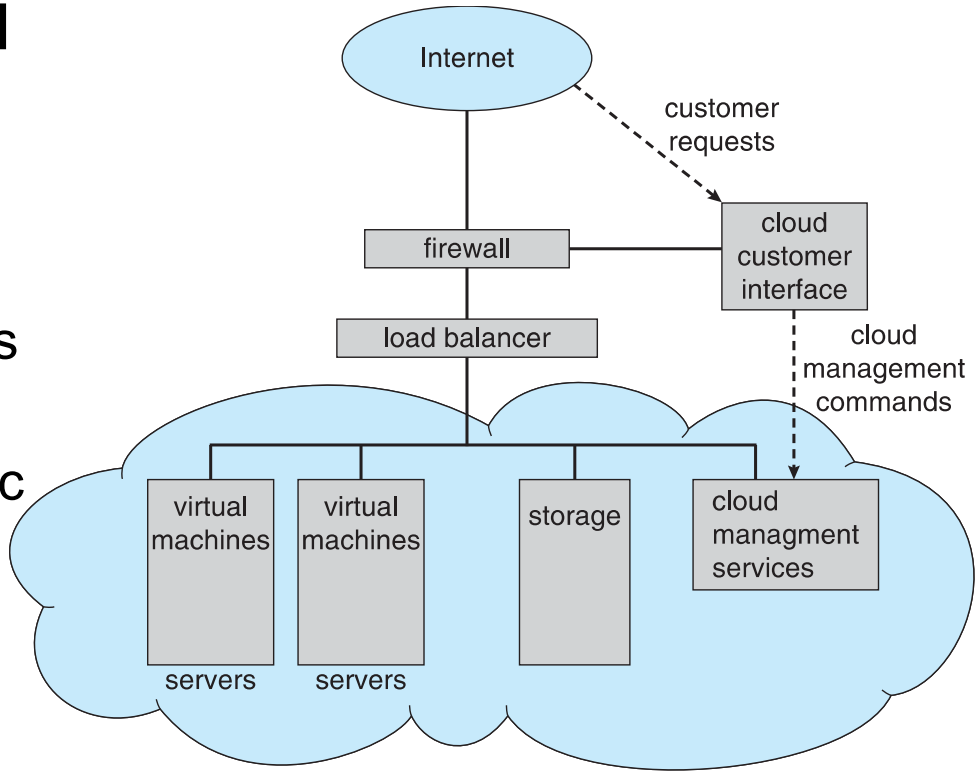
- **Cloud computing is a type of computing that delivers computing, storage, even applications as a service across a network**
- **In some ways, it's a logical extension of virtualization as based on virtualization**
  - e.g., Amazon **EC2** has thousands of servers, millions of VMs, petabytes of storage available across the Internet. Users pay based on their usage

Cloud Computing (Cont.)

- **Many types**
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - **Software as a Service** (**SaaS**) – one or more applications available via the Internet
    - e.g., word processor, spreadsheets
  - **Platform as a Service** (**PaaS**) – software stack ready for application use via the Internet
    - e.g., a database server
  - **Infrastructure as a Service** (**IaaS**) – servers or storage available over Internet
    - e.g., storage available for backup use

Cloud Computing (Cont.)

- **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



e.g., Vmware vCloud Director

1.10.6 Real-Time Embedded Systems

嵌入式系統

- **An *embedded system* is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform specific tasks.**
- **Embedded systems vary considerably.**
  - Some have OSes (general-purpose computer running standard OS with special-purpose applications, limited features)
  - Some perform tasks without an OS
- **Embedded systems amost always run embedded real-time OS.**
  - A **real-time system** is used when rigid time requirements have been placed on the operation of a processor or the flow of data

相當大的

A "short list" of embedded systems

- Anti-lock brakes
- Auto-focus cameras
- Automatic teller machines
- Automatic toll systems
- Automatic transmission
- Avionic systems
- Battery chargers
- Camcorders
- Cell phones
- Cell-phone base stations
- Cordless phones
- Cruise control
- Curbside check-in systems
- Digital cameras
- Disk drives
- Electronic card readers
- Electronic instruments
- Electronic toys/games
- Factory control
- Fax machines
- Fingerprint identifiers
- Home security systems
- Life-support systems
- Medical testing systems
- Modems
- MPEG decoders
- Network cards
- Network switches/routers
- On-board navigation
- Pagers
- Photocopiers
- Point-of-sale systems
- Portable video games
- Printers
- Satellite phones
- Scanners
- Smart ovens/dishwashers
- Speech recognizers
- Stereo systems
- Teleconferencing systems
- Televisions
- Temperature controllers
- Theft tracking systems
- TV set-top boxes
- VCR's, DVD players
- Video game consoles
- Video phones
- Washers and dryers

And the list goes on and on

https://www.youtube.com/watch?v=Qpc1M-BntaM

Embedded Systems

Robots

https://www.youtube.com/watch?v=Jky9I1ihAkg

Self-driving bicycles

https://www.youtube.com/watch?v=LSZPNwZex9s

Self-driving cars

https://www.youtube.com/watch?v=xMH8dk9b3yA

Embedded Systems (Cont.)

Drone

iRobot

即時

Real-Time Systems

- **Often used as a control device in a dedicated application**
  - e.g., controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems, automobile-engine, home-appliance controllers, weapon systems …
- **Have well-defined, fixed time constraints.**
- **Processing *must* be done within the defined constraint, or the system will fail.**
- **A real-time system functions correctly only if it returns the correct result within its time constraints.**
- **CH5**

1.11 Free and Open-Source OS

- **Open-source OS vs. closed-source**
  - **Operating systems made available in source-code format rather than just binary closed-source**
- **Richard Stallman, Free Software Foundation (FSF) encouraging the free exchange of software source code and the free use of that software**
  - Encourage sharing and improvement
- **GNU Public License (GPL) is a common license under which free software is released.**
  - GPL requires that the source code be distributed with any binaries and that any changes made to the source code be released under the same GPL license
- **e.g., GNU/Linux, BSD UNIX (including core of Mac OS X), and many more**

免費軟體 (Freeware)

- 「免費軟體（freeware）」是一種不須付費就可取得的軟體，但是通常有其他的限制，使用者並沒有使用、複製、研究、修改和分發的自由。該軟體的原始碼不一定會公開，開放的原始碼會限制重製及再發行的自由。

- http://zh.wikipedia.org/zh-tw/%E8%87%AA%E7%94%B1%E8%BB%9F%E9%AB%94

自由軟體 (Free Software)

- **Richard Stallman**
  - http://en.wikipedia.org/wiki/Richard_Stallman
  - 1953年出生於美國，1971年受聘於麻省理工學院人工智慧實驗室（AI Laboratory）
- **1984年發起GNU計劃：創造一套完全自由，兼容於Unix 的作業系統GNU（GNU's Not Unix!）。**
- 1985年成立**自由軟體基金會（Free Software Foundation, FSF)**
- **bash, ispell, gcc, grep**

自由軟體 (Free Software)

- **根據Richard Stallman和自由軟體基金會（FSF）的定義，自由軟體賦予使用者四種自由:**
  - 自由之零：不論目的為何，有使用該軟體的自由。
  - 自由之一：有研究該軟體如何運作的自由，並且得以修改該軟體來符合使用者自身的需求。取得該軟體之源碼為達成此目的之前提。
  - 自由之二：有重新散佈該軟體的自由，所以每個人都可以藉由散佈自由軟體來敦親睦鄰。
  - 自由之三：有改善再利用該軟體的自由，並且可以發表修訂後的版本供公眾使用，如此一來，整個社群都可以受惠。如前項，取得該軟體之原始碼為達成此目的之前提。

如果一軟體的使用者具有上述四種權利，則該軟體得以被稱之為「自由軟體」。

GNU通用公共許可證 GPL（General Public License）

- 使用者免費取得了自由軟體的原始碼，那麼如果使用者修改了它的原始碼，基於公平互惠的原則，使用者也必須公開其修改的成果。而這就是GPL的精神 -- 自由、分享、互惠。

- http://zh.wikipedia.org/zh-tw/%E8%87%AA%E7%94%B1%E8%BB%9F%E9%AB%94

Homework (Review)

- **Written exercises**
  - EX: 1.5, 1.6, 1.8, 1.9 (textbooks p.53)
  - 1.12 How do clustered systems differ from multiprocessor systems? What is required for two machines belonging to a cluster to cooperate to provide a highly available service?
  - 1.14 What is the purpose of interrupts? How does an interrupt differ from a trap? Can traps be generated intentionally by a user program? If so, for what purpose?

Homework (Review)

- **Written exercises**
  - For the following types of operating systems, please describe the essential properties and discuss the reasons for the rise of them: 特性
    - multiprogramming system
    - multitasking system
    - multiprocessor system
    - clustered system
    - real time system
    - embedded system

- **Virtualization software & Linux installation**
  - VMware Workstation **Player 17**
  - VirtualBox **7.2**
  - Ubuntu Linux **20.04 / 22.04 / 24.04**

Homework

(20%) 帳號: 學號
(20%) 電腦名稱: 學號
(10%) 網路要通
(20%) 重點螢幕截圖
(10%) 列出參考文獻資料來源

(80%)

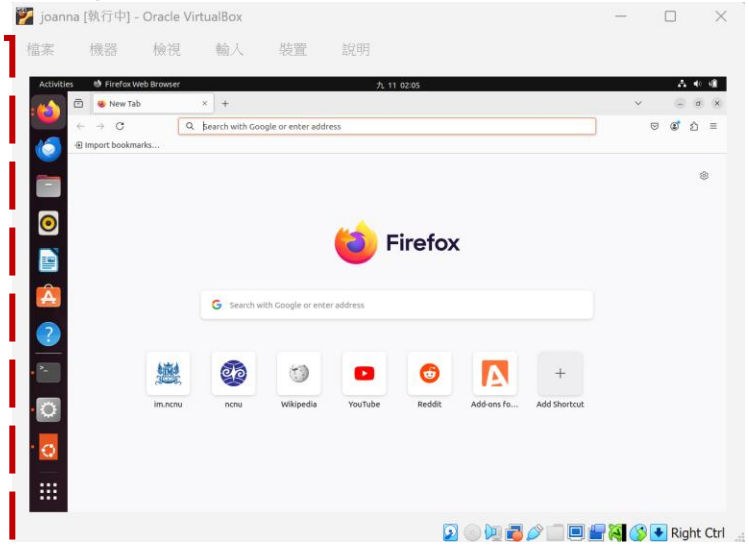
- **Virtualization software & Linux installation**
  - VirtualBox **7.2**
  - Ubuntu Linux **20.04 / 22.04 / 24.04**
  - Screenshot
  - Reference (Youtube or Web documents)

(10%)

- **Write/Compile/Run a program**
  - Screenshot, demo

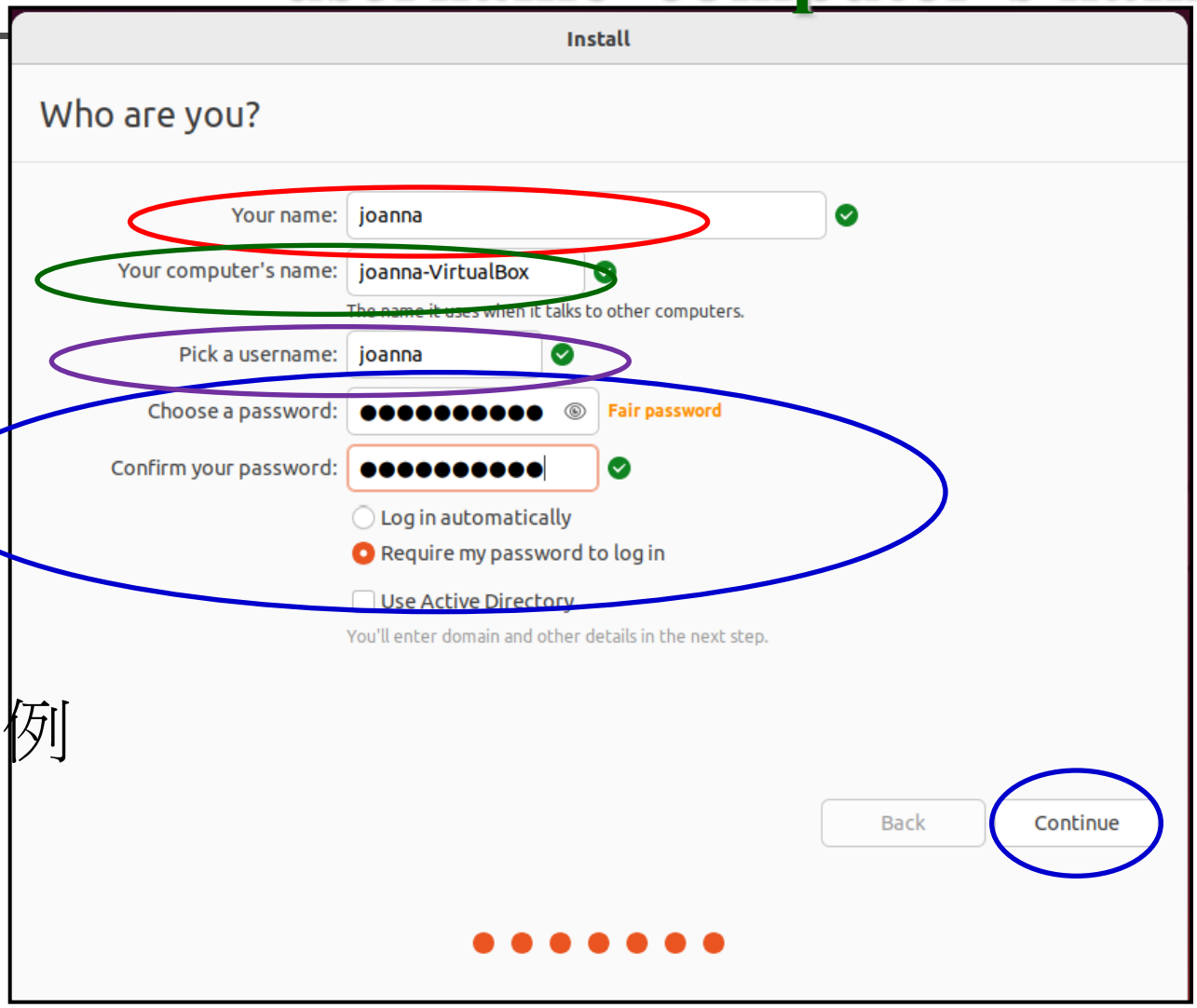
(10%)

- **demo to TA, demo the usage**

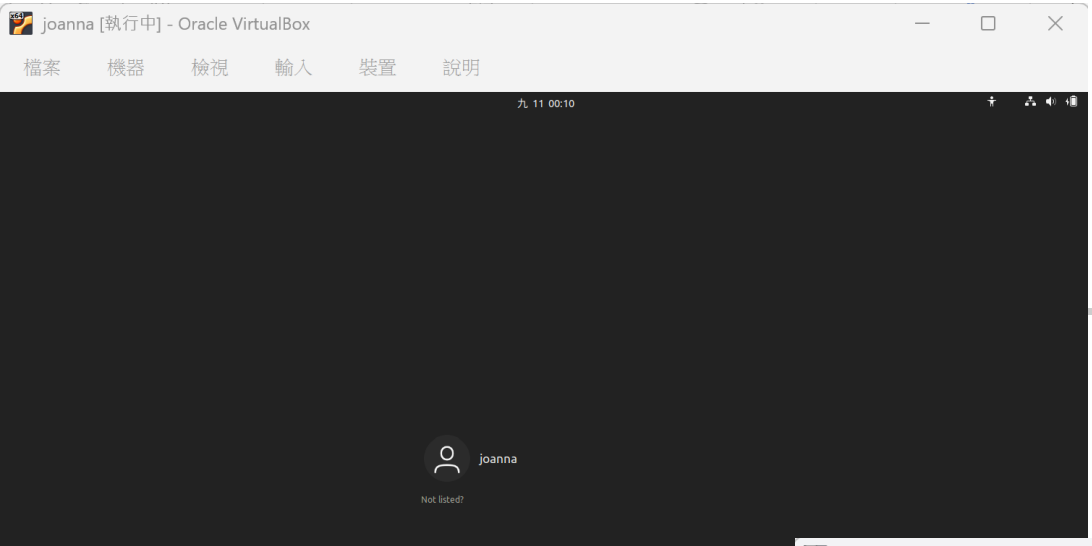


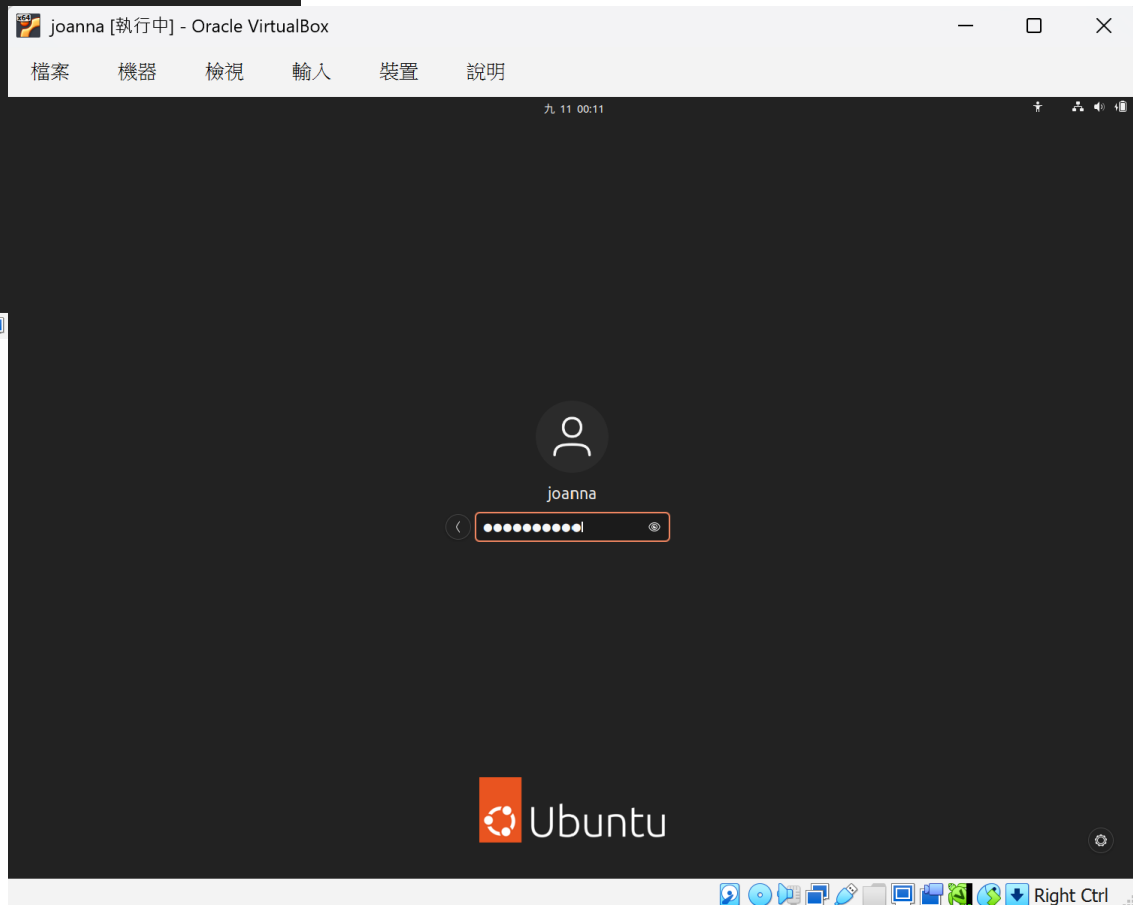
請以學號為帳號和機器名稱

username computer's name



以學號為joanna為例


joanna [執行中] - Oracle VirtualBox
檔案
機器
檢視
輸入
裝置
說明
九 11 00:10
joanna
Not listed?
Ubuntu

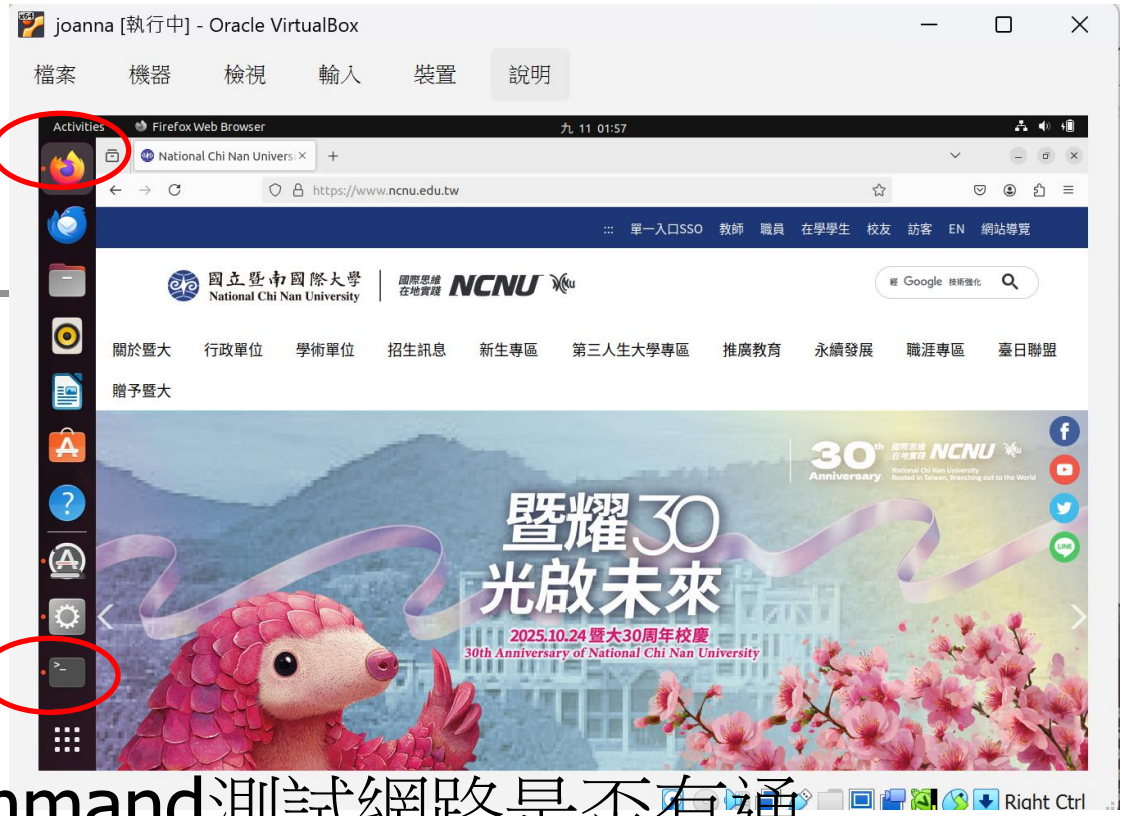

joanna [執行中] - Oracle VirtualBox
檔案
機器
檢視
輸入
裝置
說明
九 11 00:11
joanna
Ubuntu
Right Ctrl

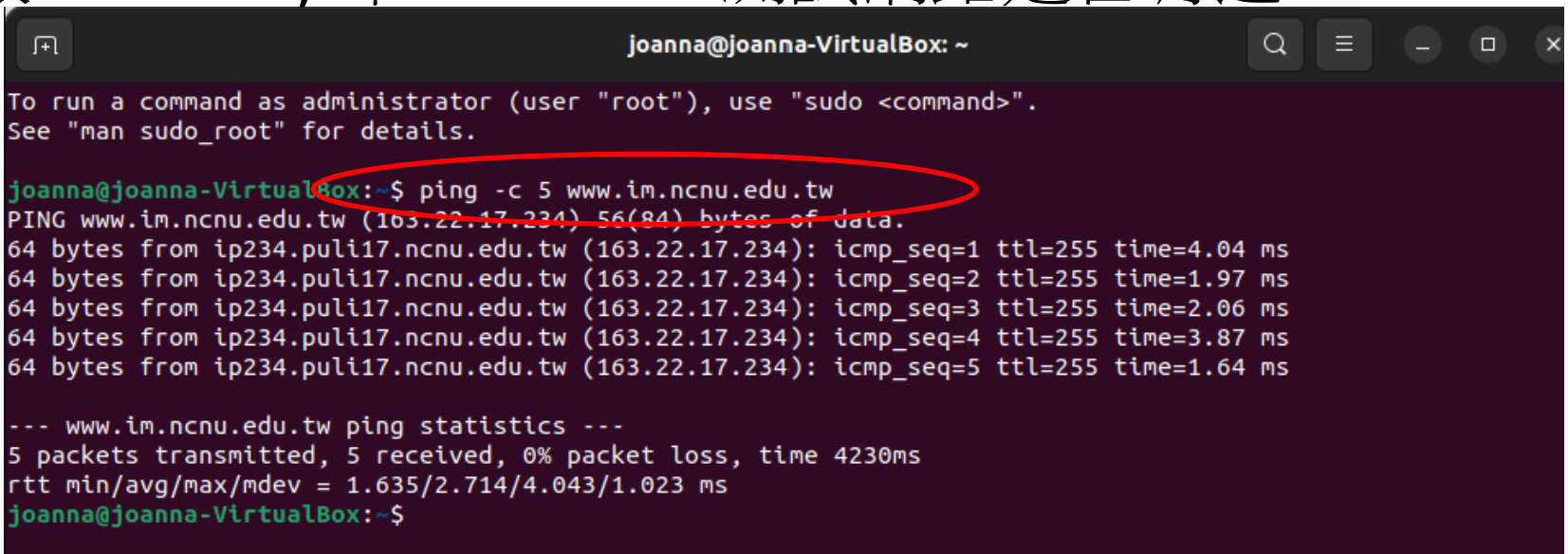
開啟browser

測試網路是否有通(5%)

(5%)

開啟Terminal, 下command測試網路是否有通





Write/Compile/Run a program
寫個小程式來執行 (10%)

- **Reference:**
  - vi, vim 程式編輯器
    https://linux.vbird.org/linux_basic/centos7/0310vi.php