Introduction

Operating Systems

Yu-Ting Wu

1

Outline

- What is an operating system
- Computer system organization
- Computer system architecture
- Operating system operations
- Resource management
- Security and protection
- Kernel data structures
- Computer system environments

2

What is an Operating System ?

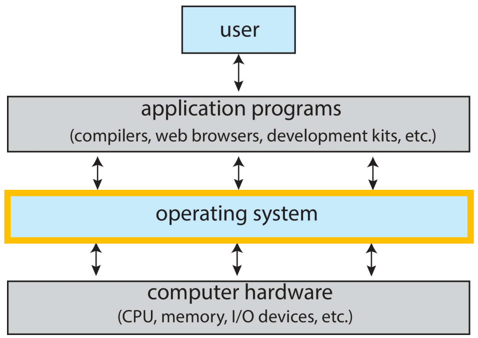
3

What is an Operating System?

- Operating system (OS) is a **software program** that acts as an intermediary between a user and the computer hardware
  - Execute user programs
  - Make the computer system convenient to use
    - Such that users can focus on their problems
  - Use the computer hardware in an efficient manner

4

Computer System Components



An operating system can be considered as a government or environment provider

5

User View (Features)

- Varies by the types of the computer

Personal Computer (PC)	Mainframe, Workstation	Handheld Computer	Embedded Computer
ease of use	reliability efficiency fair sharing	individual usability battery life	run without user intervention

6

System View (Tasks)

- **A resource allocator**
  - CPU time
  - Memory space
  - File storage
  - I/O devices
- **A control program**
  - Control execution of user programs
  - Prevent errors and misuse

7

Definition of an OS

- **No universally accepted definition**
  - Because of the myriads designs and uses of OSes
  - US Dept. of Justice against Microsoft (1998)
    - The stuff shipped by vendors as an OS
    - The one program running all the times on the computer

8

The Goals of an OS

Convenient

for the users

Conflict !

Efficiency

for the computer system

Windows Vista

Microsoft Windows xp

9

OS and Hardware

- Can change due to new computer architectures and hardware devices

(multi-core CPU, GPU)

OS → resource management → **Hardware**

Hardware → new features support → **OS**

(multi-tasking)

Learn OSs by tracing their evolution enables us to predict what they will become !

10

The Development of an OS

- In the past time, operating systems are usually implemented with low-level languages
- Later, **high-level languages** are used for developing an operating system because of the increasing complex functionalities and porting issues
  - But system calls are still implemented by assembly language

➡ As a result, more and more people can involve in the development of operating systems

- Another trend is **modulation**

11

Booting: How an OS is Launched

- **Bootstrap program**
  - Simple program to initialize the system and load the kernel
  - Typically stored in ROM or EPROM
  - Initialize the entire system, including CPU registers, device controllers, memory, … etc.

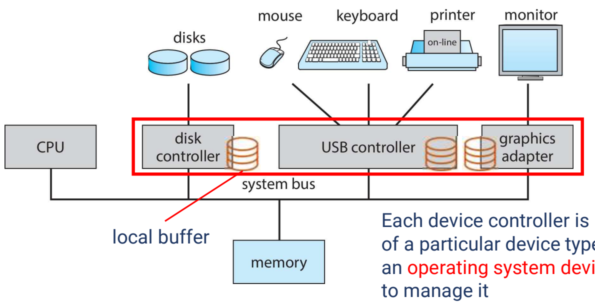
- Steps
  - Booting
  - Load kernel
  - Start system daemon (e.g., login)
  - Kernel (HW/SW) interrupt driven

12

Computer System Organization

Computer System Organization

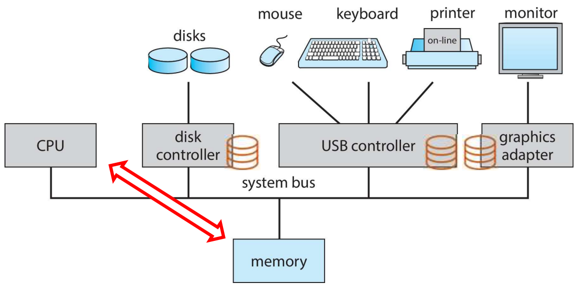
- CPU (or CPUs) and device controllers connect through common **bus**, which provides access to memory



Each device controller is in charge of a particular device type, and has an operating system device driver to manage it

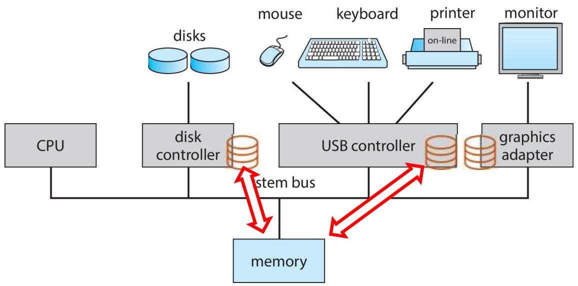
Computer System Operations

- CPU moves data from/to main memory to/from local buffer for executing programs



Computer System Operations (cont.)

- IO: from the device to local buffer of controller
  - Use **interrupt** to inform CPU that it has finished its task



Computer System Operations (cont.)

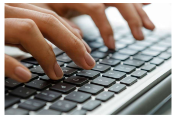
- A simplest design for accessing devices
  - Use instructions to test when a device is ready
  - Busy/wait

```
#define OUT_CHAR 0x1000  // device data register
#define OUT_STATUS 0x1001 // device status register

current_char = mystring;
while (*current_char != '₩0') {
   poke(OUT_CHAR,*current_char);
   while (peek(OUT_STATUS) != 0); // busy waiting
   current_char++;
}
```


Interrupt

- Busy/wait is inefficient
  - CPU cannot do other task while testing device
- Interrupt provides a way to change the **flow of control** in the CPU
- **Hardware interrupt (signal)**
  - Service requests from one of the devices
    - Ex: keyboard, mouse click, etc.

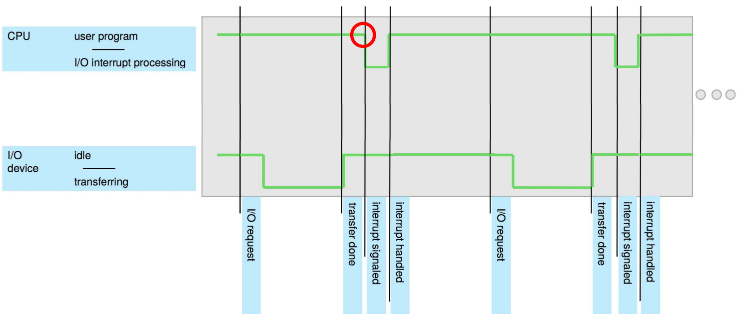


- **Software interrupt (trap)**
  - Invalid memory access
  - Software error
    - Ex: division by zero
  - System calls
    - Request for system services

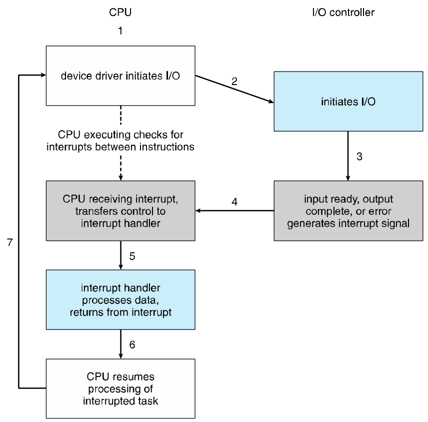
➔ system call (open)

Interrupt Timeline

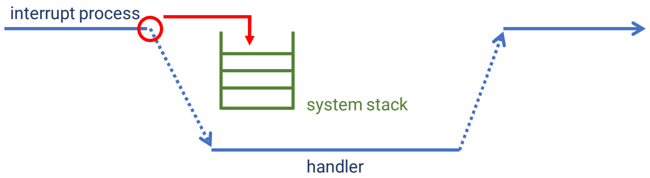
- Transfer control to the interrupt service routine
- Must save the address of the interrupted instruction



Interrupt-driven I/O Cycle



Interrupt Handling

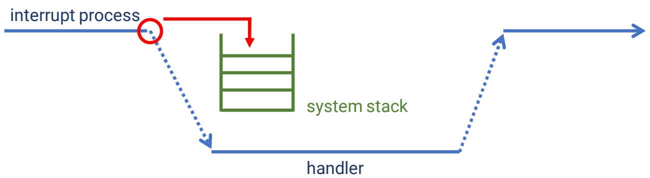


- Saving of the address of the interrupted instruction
  - Fixed location
  - Fixed location per interrupt type
  - Stacks
- Interrupt disabling /enabling issues

Interrupt Handling

- Determine which interrupt service routine should be called
  - **Generic handler**
    - A specific program for handling all types of interrupts
    - Call the corresponding interrupt service routine after checking
    - Inefficient but can handle infinite types of interrupts
  - **Interrupt vector**
    - Use a vector to store all interrupt service routines
    - Hardware jumps to the corresponding interrupt service routine based on an **interrupt number** (ID)
    - Efficient but can only handle a fixed number of interrupt types
  - Current OSs usually hybrid of the two strategies

Interrupt Handling Procedure (Summary)

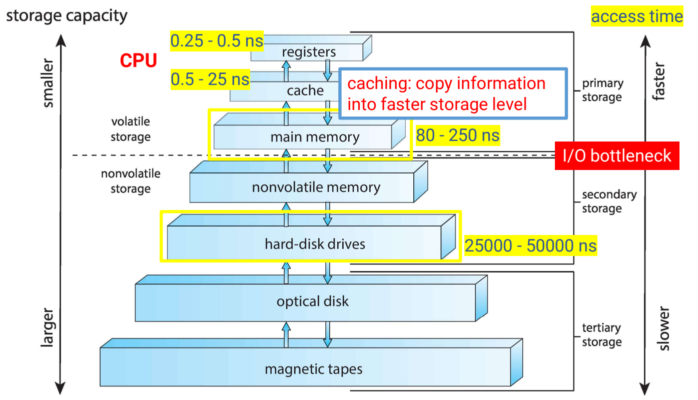


- Steps:
  - Save interrupt information
  - OS determine the interrupt type
  - Call the corresponding handlers
  - return to the interrupted job by restoring the information of original process

Storage Structure

- **Main memory**
  - The only large storage media that the CPU can access directly
  - Typically volatile
- **Secondary storage** (ex: HDD, USB sticks, CD, DVD, …)
  - Extension of main memory that provides large storage capacity
  - Typically nonvolatile
- Organized in hierarchy based on
  - **Speed**
  - **Cost**
  - **Volatility**

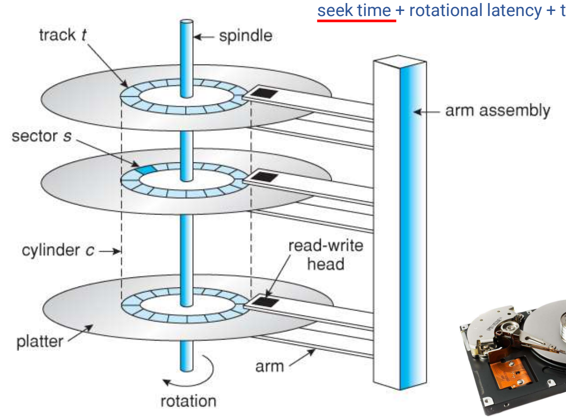
Storage Structure (cont.)



25

Magnetic Disks

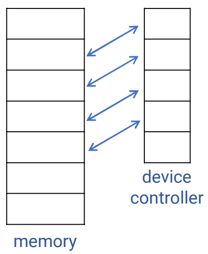
seek time + rotational latency + transfer time



26

Strategies for Handling I/O

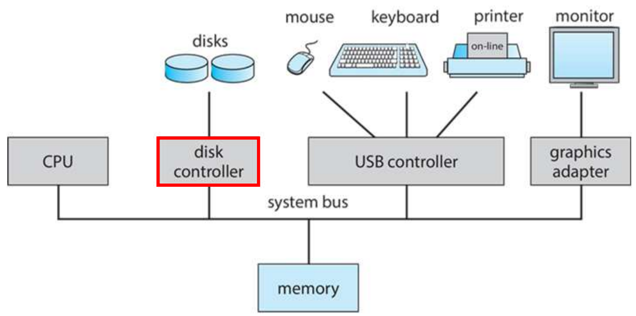
- **Interrupted-based I/O**
  - Slow devices
  - informed when jobs have been done
  - Ex: disk
- **Programmed I/O (pulling)**
  - Keep asking if the jobs have been done
  - Ex: network interface card
- **Memory-mapped I/O**
  - Frequently used or very fast devices
  - Special I/O instructions are used to move data between memory & device controller registers
  - Ex: GPU



27

Data Transferring

- Device controllers are responsible of transferring data between the peripheral devices and their local buffer storages



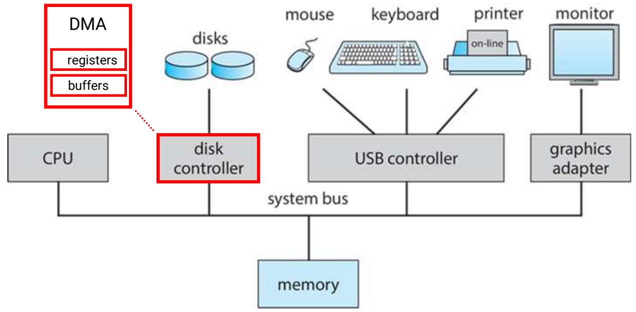
28

I/O Operation Procedure

- CPU setups specific controller registers within the controller
- Read / Write
  - Read: devices ➔ controller buffers ➔ memory
  - Write: memory ➔ controller buffers ➔ devices
- Notify the completion of the operation by triggering an interrupt

DMA: Direct Memory Access

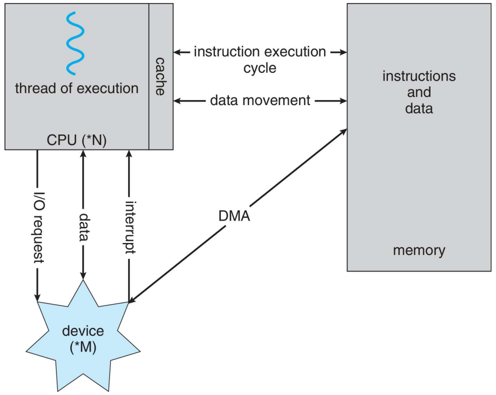
- Transfer blocks of data without bothering CPU



DMA: Direct Memory Access (cont.)

- **Goal**
  - Device controller can transfer blocks of data from buffer storage to main memory without CPU intervention
  - Only one interrupt is generated per block (rather than per byte), thus avoiding CPU handling excessive interrupts
- **Procedure with DMA**
  - Execute the device driver to setup the registers of the DMA controller
  - DMA moves blocks of data between the memory and its own buffers
  - Transfer from its buffers to its devices
  - Interrupt the CPU when the job is done

Storage Structure Summary



Computer System Architecture

33

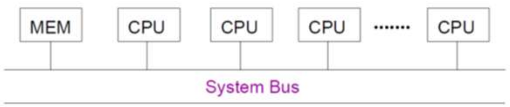
Single-Processor Systems

- One main CPU per system
  - Control other low-end processors, e.g., disk controller microprocessors
  - Ex: earlier desktop or mobile devices

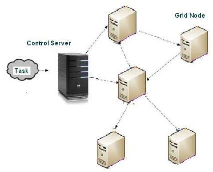
34

Multi-Processor Systems

- **Tightly coupled**
  - More than one processor in close communication sharing bus, memory, and peripheral devices



- **Loosely coupled**
  - Otherwise (such as distributed systems)
  - Each machine has its own memory



35

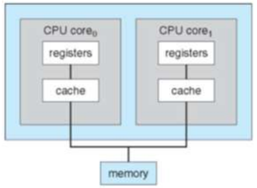
Multi-Processor Systems (cont.)

- **Symmetric model**
  - Each processor in the system runs an identical copy of the OS
- **Asymmetric model**
  - master–slave
  - Commonly seen in extremely large systems
- Task allocation strategies
  - Dynamic allocation
  - Pre-allocation

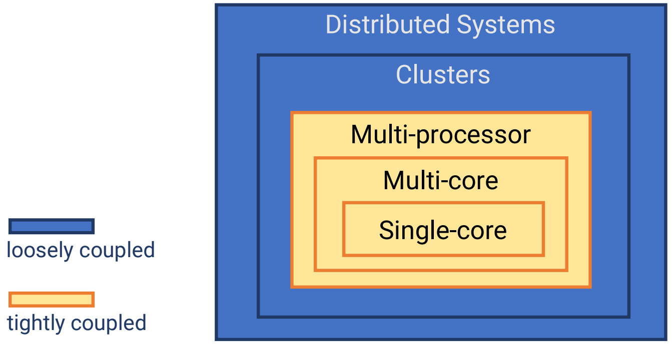
36

Multi-Processor Systems (cont.)

- **Advantages of multi-processor systems**
  - **Speedup**: better throughput
  - **Lower cost**: building one small fast chip is very expensive
  - **More reliable**: Graceful degradation and fail soft
- The recent trend: from a fast single processor to lots of processors
  - **Multiple cores** over a single chip
  - Hyperthreading (logical core)



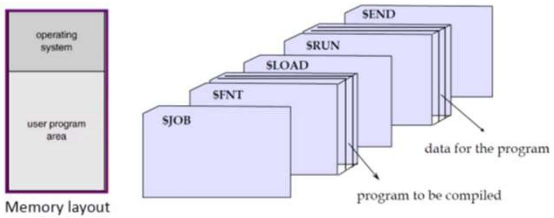
Computer System Architecture Summary



Operating System Structure

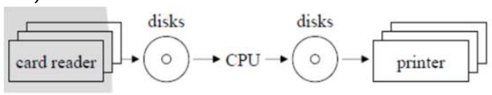
Simple Batch System

- Workflow
  - Users submit data (program, data, control card)
  - Operator sort jobs with similar requirement
  - OS simply transfer control from one job to the next one
    - Resident monitor: automatically transfer control from one job the next



Simple Batch System (cont.)

- **Problem of batch systems**
  - One job at a time ➔ multi-programming
  - No interaction between users and jobs ➔ time sharing
  - CPU is often idle
- **Spooling** (Simultaneous Peripheral Operation On-Line)
  - Replace sequential-access devices with random-access devices (disks)



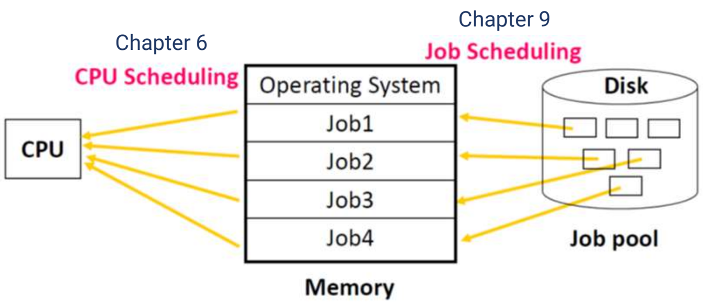
  - Ex: printer
    - Data for printing is first written into a directory as files, then printed by the printer

Multi-Programming

- Single user cannot always keep CPU and I/O devices busy
  - Even with spooling, disk I/O is still too slow compared to CPU and memory
- Put multiple programs in memory
- OS organizes jobs so that the CPU always has one to execute
  - When job has to wait (e.g., for I/O), OS switches to another job
  - Increase CPU utilization
  - Issue: job and CPU scheduling

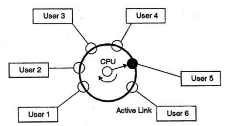
Multi-Programming (cont.)

- Job scheduling and CPU scheduling



Time-Sharing (Multi-Tasking)

- CPU switches jobs frequently so that users can interact with each job while it is running
  - A logical extension of multi-programming
  - **Interactivity** !
    - Response time should be less than 1 sec.



- Brings lots of new issues
  - Online file system (chapter 11 and chapter 12)
  - Virtual memory (chapter 10)
    - Allow execution of processes not completely in memory
  - Job synchronization (chapter 7 and chapter 8)
  - Protection and security

Batch, Multi-Programming, Time-Sharing

	Batch	Multi-Programming	Time-sharing (Multi-Tasking)
System Model	single user single job	multiple programs	multiple Users multiple programs
Purpose	simple	resource utilization	interactive response time
OS Features	N.A	CPU scheduling I/O system	file system virtual memory synchronization

Resource Management

Process Management

- Process (running program) is an **active** entity using physical and logical resources
  - Memory, I/O buffers, data, …etc.
  - Data structure representing current activities:
    - Program counter
    - Stack
    - Data section
    - CPU registers

context

- Activities
  - Process creation, suspension, resumption, and deletion
  - Process synchronization
  - Process communication
  - Deadlock handling

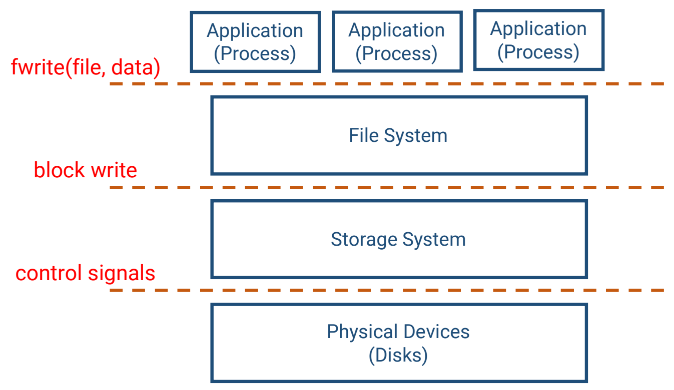
Memory Management

- To execute a program, all (or part) of the instructions and its needed data must be in memory
- Memory management
  - Determines what is in memory and when
  - Optimizes CPU utilization and response time
  - Sometimes need hardware support
- Activities
  - Keep track of which parts of memory are currently being used and by whom
  - Memory scheduling (move process and data in/out memory)
  - Memory allocation and deallocation

File System Management

- OS provides a uniform logical view of information storage
  - **File: a logical storage unit**
    - Need to be cross-platform
    - Treated as a sequence of bits, bytes, lines, records
  - The format of a file is determined by the application
- File system services provided by OS
  - File creation and deletion
  - Directory creation and deletion
  - Directory hierarchy maintenance
  - File backup

File System Management (cont.)



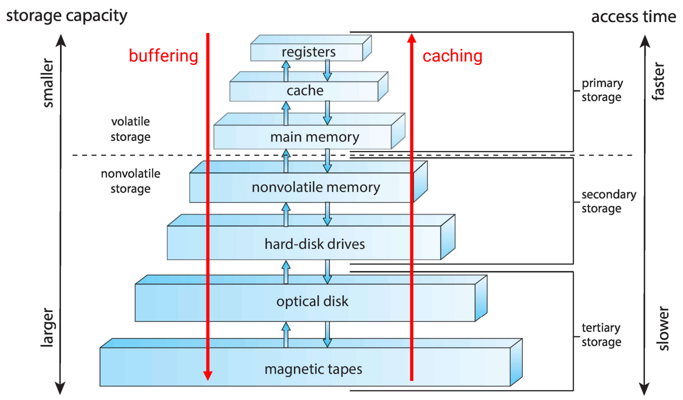
Mass and Tertiary Storage Management

- Usage
  - Store data that does not fit in main memory or data that must be kept for a long time (e.g., disks)
  - Backups of disk data, seldom-used data (e.g., tape drives and tapes, CD/DVD drives and platters)
- Activities
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling (e.g., FCFS)

I/O System Management

- Hide the peculiarities of specific hardware devices from users
  - Provide a uniform interface for new devices (driver) so they can “plug and play”
- Components
  - **Buffering**, **caching**, and **spooling** system
  - A general device-driver interface

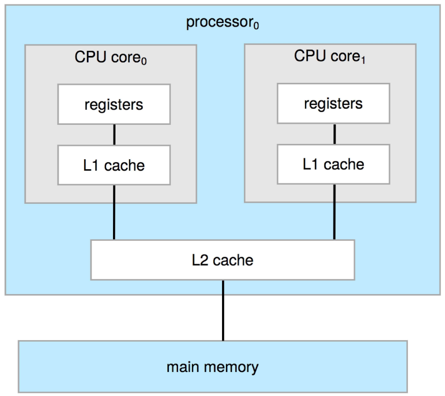
Recap. Storage Structure



Recap. Storage Structure

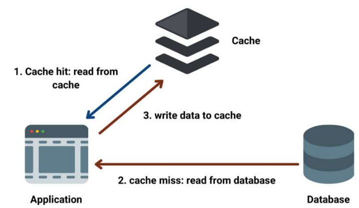
Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	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

Recap. Storage Structure



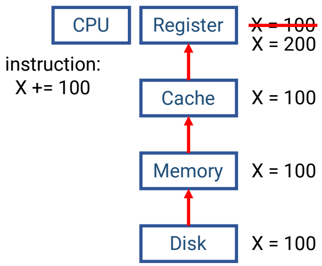
Caching

- Information is **copied** to a faster storage system on a **temporary** basis
- Assumption: data will be used again soon (locality)
- Cache management
  - Cache size
  - Replacement policy



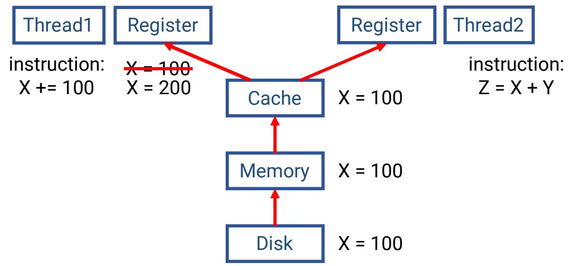
Caching (cont.)

- Issues: data coherence and consistency among several storage levels
- Uni-tasking: no problem



Caching (cont.)

- Issues: data coherence and consistency among several storage levels
- Uni-tasking: no problem
- Multi-tasking: need handle



Protection and Security

Protection and Security

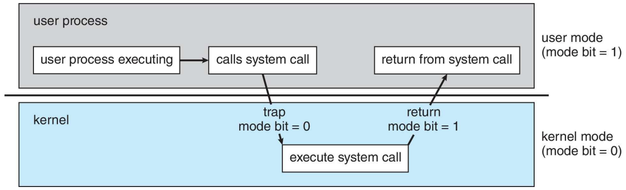
- **Goal**
  - Prevent error and misuse
  - Resources are only allowed to be accessed by authorized processes
- **Protection**
  - Any mechanism for controlling the access of processes or users to the resources defined by the computer system
- **Security**
  - Defense of a system from external and internal attacks
  - Ex: viruses, denial of service, identity theft

Protection

- **Dual-mode operations**
  - **User mode**
    - Executions except those after a trap or an interrupt occurs
  - **Monitor mode (system mode, privileged mode)**
    - Can execute all instructions including privileged ones (machine instructions that may cause harm)
  - Implemented by a **mode bit** and **system calls**

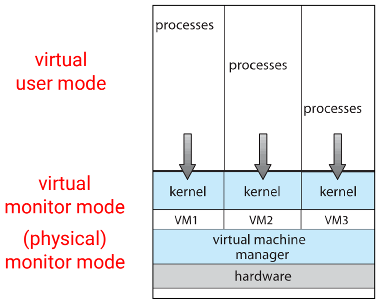
Protection (cont.)

- **System calls**
  - Trap to OS for executing privileged instructions
  - Protect hardware resources such as I/O devices, memory, and CPU



Protection (cont.)

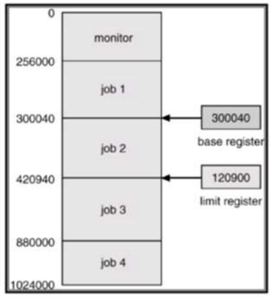
- **Virtual machine** (more modes)
  - Provide an interface that is identical to the underlying bare hardware

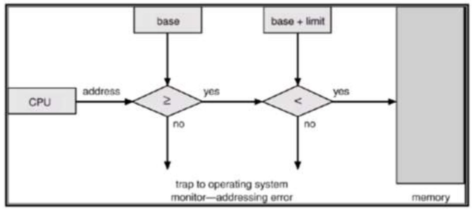


Protection (cont.)

- **I/O protection**
  - I/O devices are scarce resources, user programs must issue I/O through OS
  - Ex: fopen (open), gets (read), puts (write)
- **Memory protection**
  - Prevent a user program from modifying the code or data structures of either the OS or other users
  - Ex: instructions to modify the memory space
- **CPU protection**
  - Prevent user programs from sucking up CPU power
  - Implement by timers and time-sharing
    - Need context switch

Protection (cont.)





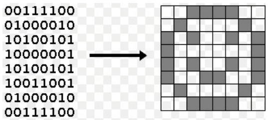
65

Kernel Data Structures

66

Kernel Data Structures

- Frequently used data structures
  - Array
  - List (singly, doubly, circular)
  - Stack
  - Queue
  - Tree
  - Hash
  - Bitmap: string of *n* binary digits representing the status of *n* items



67

Computer System Environments

68

Overview

- Today's computing environments have changed a lot
  - Traditional
  - Mobile
  - Client server
  - Peer-to-peer
  - Cloud computing
  - Real-time embedded
- Trends: network and mobility
- The relatively new mobile and cloud computing bring new issues and concerns on OS design

69

Traditional

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

70

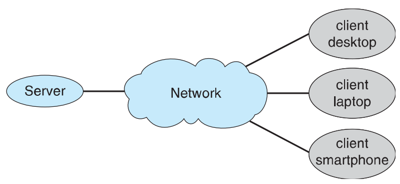
Mobile

- Handheld smartphones, tablets, etc.
- Has functional difference with a "traditional" laptop
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like augmented reality
- Leaders are Apple iOS and Google Android

71

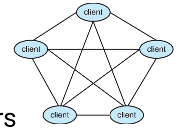
Client Server

- Dumb terminals supplanted by smart PCs
- Many systems are now 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 interface for clients to store and retrieve files



72

Peer-to-Peer (P2P)



- Another model of distributed system
- P2P does not distinguish clients and servers
  - The role depends on who is requesting or providing a service
- Node must join P2P network
  - Registers its service with central lookup service on network
  - Broadcast request for service and respond to requests for service via discovery protocol
- Examples include Napster and Gnutella, Voice over IP (VoIP) such as Skype

Cloud Computing

- Deliver computing, storage, and even applications as a service across a network
- 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 (i.e., online photo editor)
  - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server)
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use)

Cloud Computing (cont.)

- Why cloud computing becomes so popular?
  - Fast improvement of web technology
    - Portals, network computers …
  - Network connectivity
  - New categories of devices
  - Embedded computing
    - Car engines, ETC, robots, home automation, AR glasses

Real-Time Embedded Systems

- Most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, real-time OS
  - Use expanding
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Hard real-time system
    - Processing **must** be done within constraint
    - Correct operation only if constraints met
  - Soft real-time system
    - Missing a timing is serious but does not necessary result in failure (ex: multimedia)
- Real-time means on time!

Free and Open-Source OSes

- Definition: OS with available source
  (Otherwise: closed-source OS. E.g., MS Windows, iOS)
- Arguably issues on bugs, security, support
- Examples: GNU/Linux, BSD UNIX …

Objectives Review

- Describe the general organization of a computer system and the role of interrupts
- Describe the components in a modern, multiprocessor computer system
- Illustrate the transition from user mode to kernel mode
- Discuss how operating systems are used in various computing environments