Module Overview

House keeping

- Google classroom: y5tvdbs
- Meetings:
  - Mondays 13:30
  - Thursdays 9:30
  - Fridays 7:30
  - Venue: The Great Hall
- Prescribed text
  - Silberschatz, A., Galvin, P.B., Gagne, G. (2012). Operating System Concepts (9th ed.). New Jersey, USA: Wiley

Time table resolution policy

House keeping

- Labs/Practicals
  - Unsupervised tasks
  - Must do all assigned practical work!
    - You will asked to submit some of your work
    - The choice will be random
- Assessment
  - 40% CA + 60% EOS
- CA
  - 10% practical work
  - 30% Tests

House keeping

- Academic dishonesty
  - A serious offense which carries a maximum penalty of summary dismissal from the University
- Examples
  - Submitting a copied assignment
  - Conferring during a test/exam

What is an Operating System (OS)

- Software that acts as a mediator between user and computer hardware

Purpose

- Provide user with an environment to execute program
  - Convenience
  - Efficient use of computing resources
- Manage computer hardware and provide mechanism for
  1. Correct operation of the computer system
  2. Prevent user programs from interfering with proper operation of the system

Plethora of OSs

- How many OSs are there?
  - Why so many?
- OS design is guided by OS goal
  - Basis for choosing among various algorithms and strategies
- OS is a large system
  - Modular design and development
  - Each module must have a well defined interface and function
  - Interface specifies inputs and outputs
  - Function transform inputs into outputs

Contexts of OSs

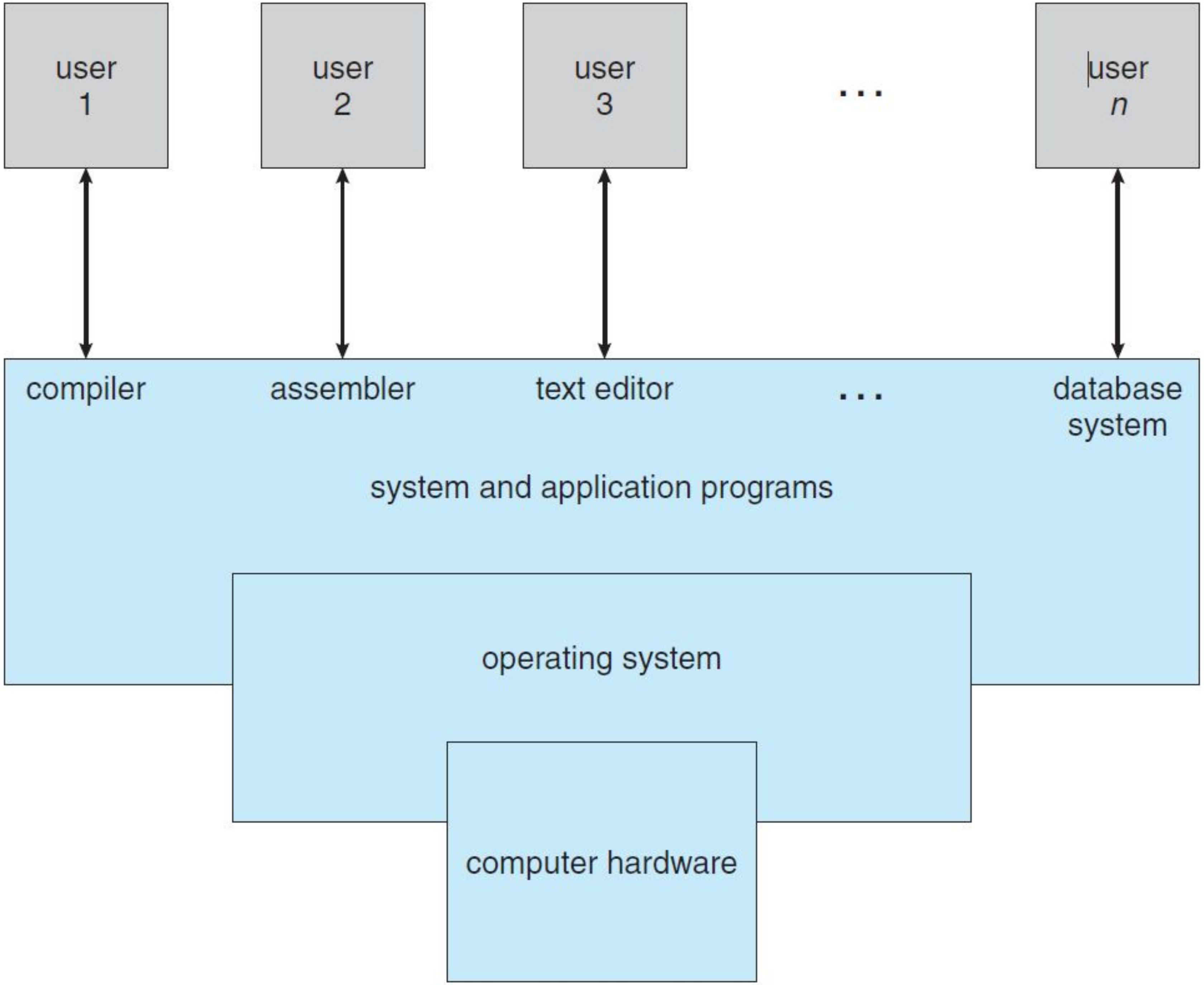
1. Mainframe computers
   - Large and powerful computer system with multiple terminals connected to a central “system box”
   - Terminal = Monitor + keyboard + other I/O devices
   - Simultaneous multiple users
   - Main goal: **optimize hardware utilization**

Contexts of OSs

2. Mobile device
   - Main goal: **convenience of use**
3. Personal computer (PC)
   - Main goal: **convenient interface + performance**

Two views of a computer system

1. Hardware + OS + application programs + users
   - Hardware = CPU + memory + I/O devices
   - App examples: word processors, spreadsheets, compilers, web browsers, etc
   - OS function: **control hardware and coordinate its use among various apps and users**
2. Hardware + software + data


user 1
user 2
user 3
...
user n
compiler
assembler
text editor
...
database system
system and application programs
operating system
computer hardware

Users' view of OS

- PC
  - Single user monopoly
  - Goal is to maximize ease of use
  - Some attention to performance
  - No attention to resource utilization
- Mainframe/Minicomputers
  - Maximize resource utilization: efficiency
  - Fairness to the multiple users
- Workstations = networked PCs + Servers (file, compute, print)

Users’ view of OS

- Workstations = networked PCs + Servers (file, compute, print)
  - Balance between usability of an individual workstation and resource utilization for the shared resources
- Mobile computers
  - Ease of use overload!
- Embedded computers
  - No user intervention

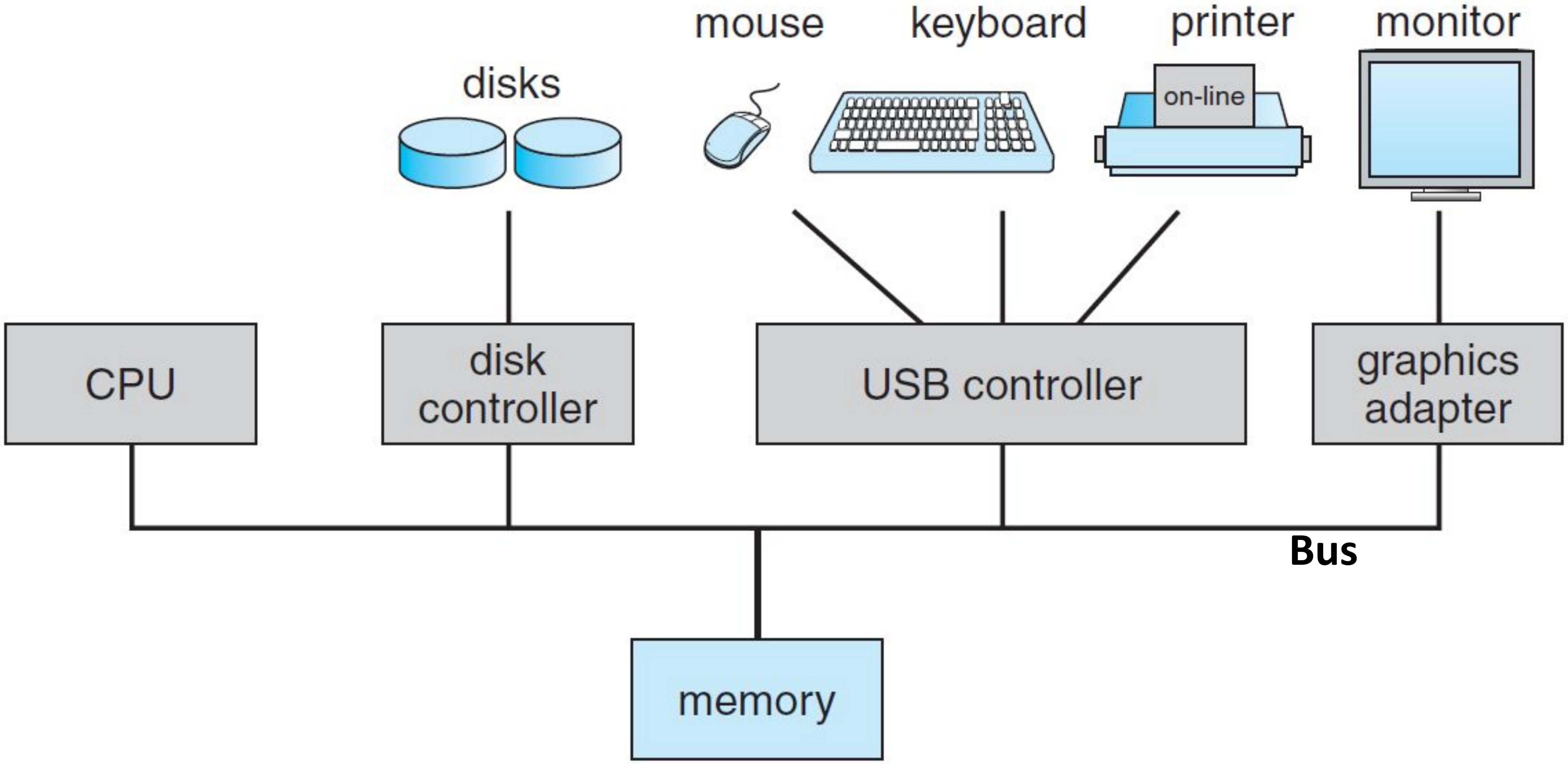
System view of OS

- Resource allocator
  - Fairness
  - Efficiency
  - Crucial in a multiuser environment
- Control program
  - manage execution of system and user programs to prevent errors and improper use of the computer
  - control I/O devices

What is an OS?

- Everything that ships when one orders an OS?
- Kernel?
- Kernel + system programs?
    - Definition of what constitute system programs is problematic
        - Resulted in a 1998 US Department of Justice lawsuit against Microsoft – too much functionality in OS left little room for competition with application software vendors.
- Current trends in mobile Oss are going against the outcome of 1998 lawsuit
    - Kernel + middleware (for graphics, multimedia, graphics, etc)

Computer System Organization



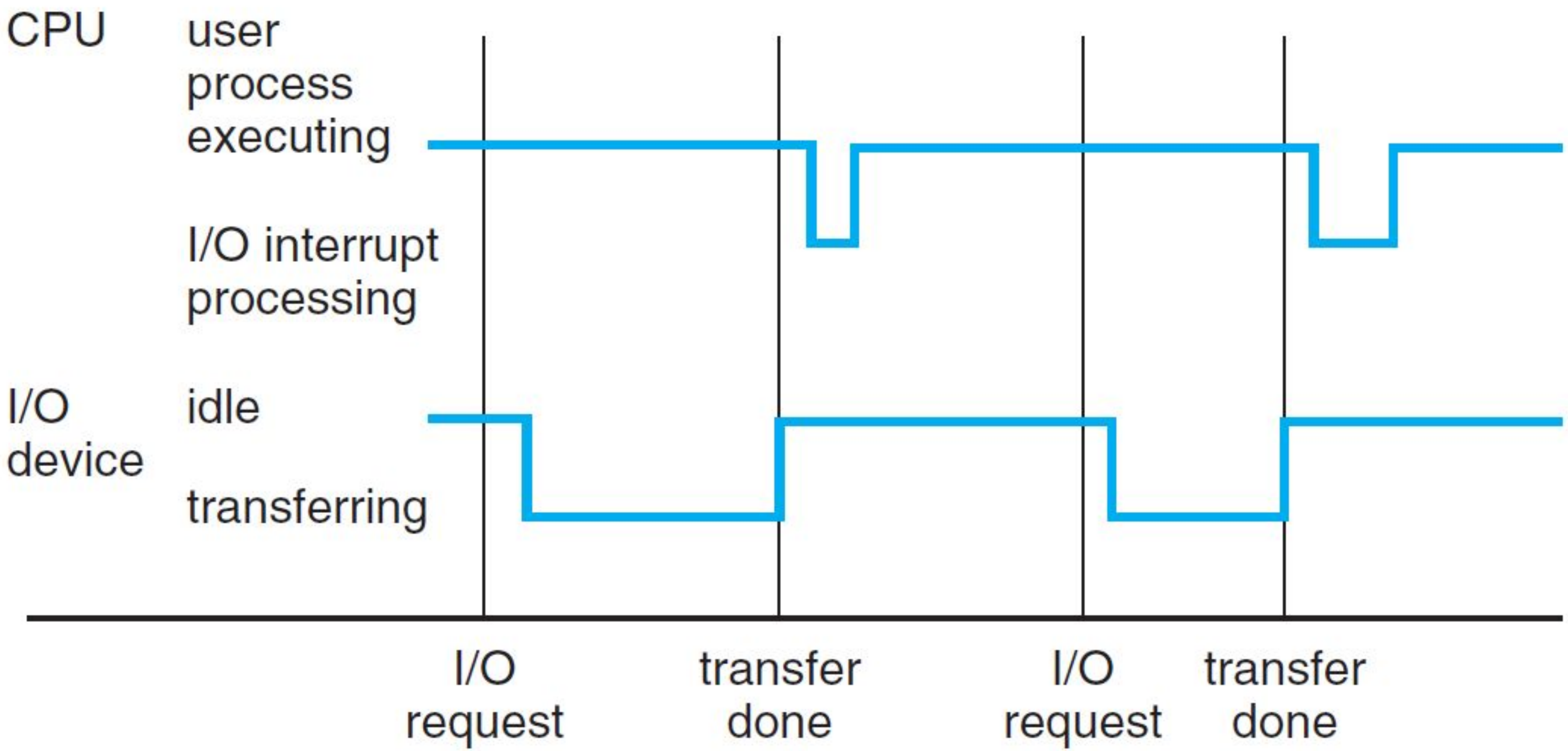
Starting the System

- Bootstrap program
  - Located in either ROM or EEPROM (firmware)
  - Initializes system: CPU register/memory contents, controllers
  - Locates and loads OS kernel
  - Kernel then starts other system daemons
- Kernel + daemons run in infinite loops waiting for some events to occur - interrupts

Interrupts

- Two kinds
  - hardware interrupts
  - software interrupts – system calls (monitor calls)
- Service routine – respond to interrupt
- Interrupt vector – facilitate fast and efficient call to service routines
- The system stack – return address + system state

Interrupts


CPU
user process executing
I/O interrupt processing
I/O device
idle
transferring
I/O request
transfer done
I/O request
transfer done

Storage Structure

- Main memory
  - Also called random-access memory (RAM)
  - Uses dynamic random-access memory (DRAM) semiconductor technology
  - CPU can only access this memory
- Other memory
  - Read-only memory (ROM)
  - Electrically erasable programmable read-only memory (EEPROM)

Organization of memory

- Array of addressable bytes
- Load – load contents (byte/word) at an address into CPU registers
- Store – Store contents of a CPU register in main memory
- CPU automatically transfers instructions from main memory for execution
- Fetch, decode, execute cycle – Instruction-execution cycle on a system with a von Neumann architecture

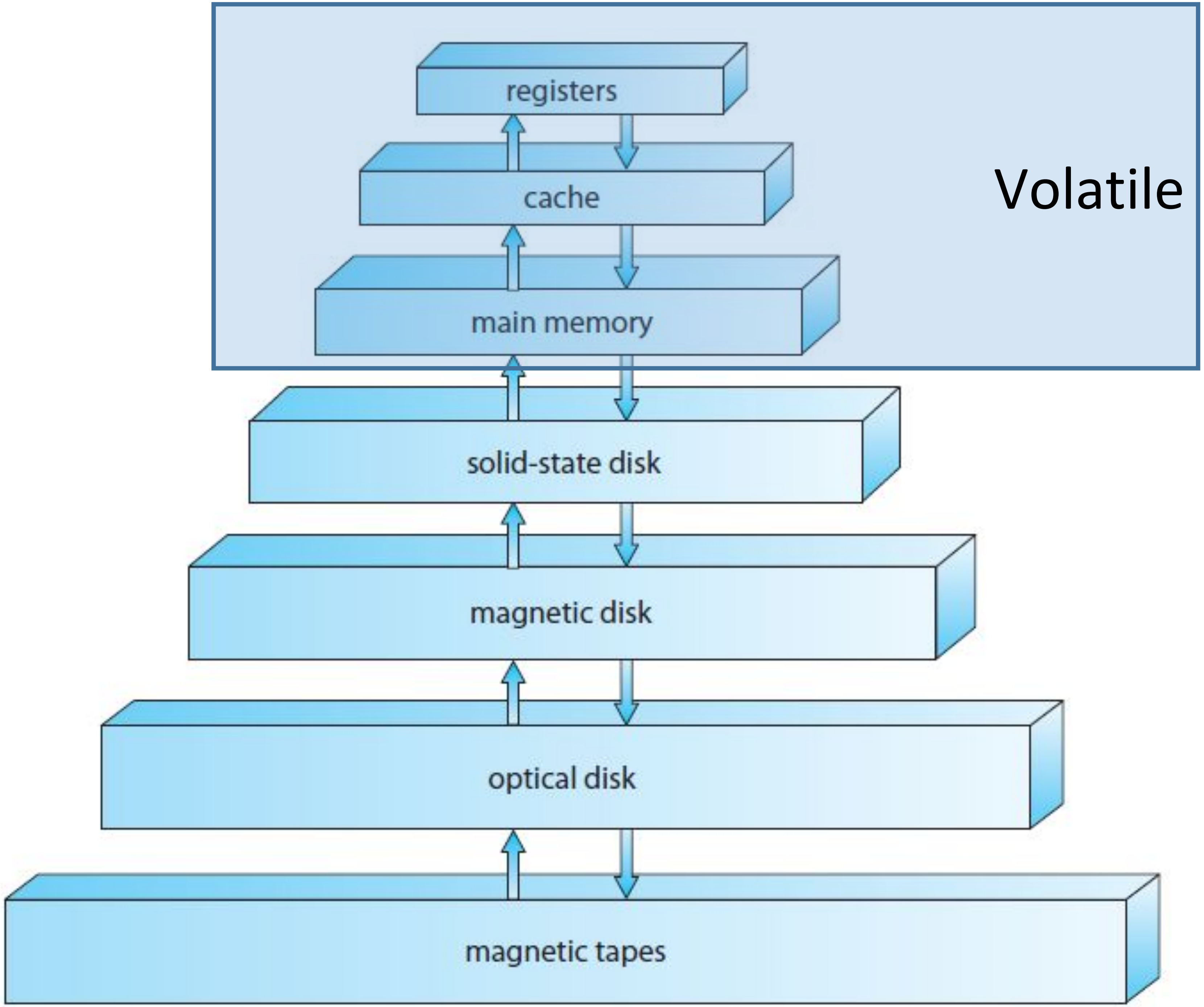
Limitations of main memory

- Limited in size
  - Due to cost
- Volatile

Secondary storage

- Magnetic disks
- Solid-state disks
  - NVRAM – RAM with battery for persistence
  - NVRAM backed by magnetic disk
  - Flash memory
- Optical disks
- Magnetic tapes
- Paper tapes - obsolete

Storage hier

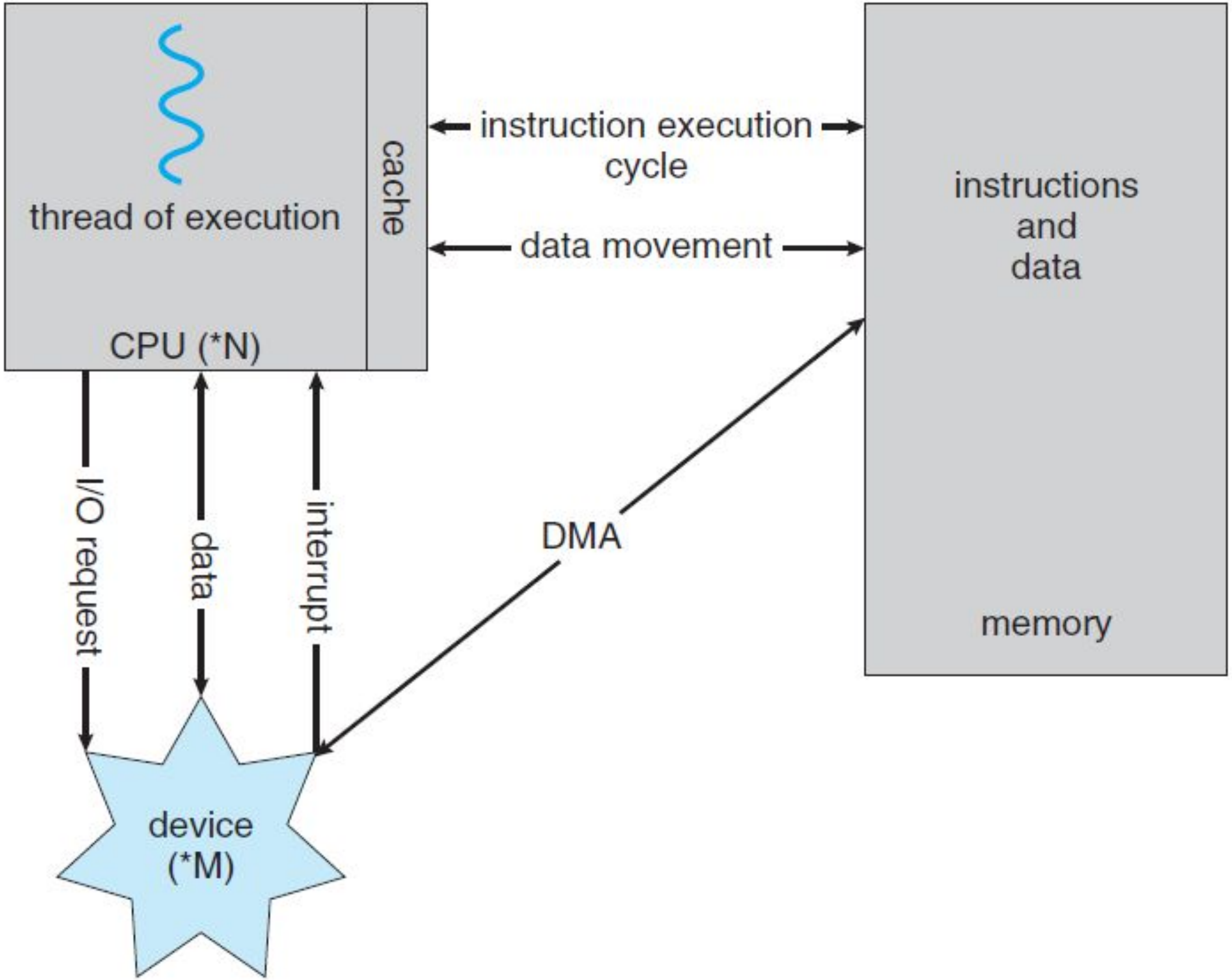


Increasing cost and speed

I/O structure

- Each I/O device is connected to the bus via a controller
- Several devices can share a controller
- Controller has a local buffer and registers
- The OS has a device driver for each controller
    - provides the rest of OS with a uniform interface to the device

I/O structure



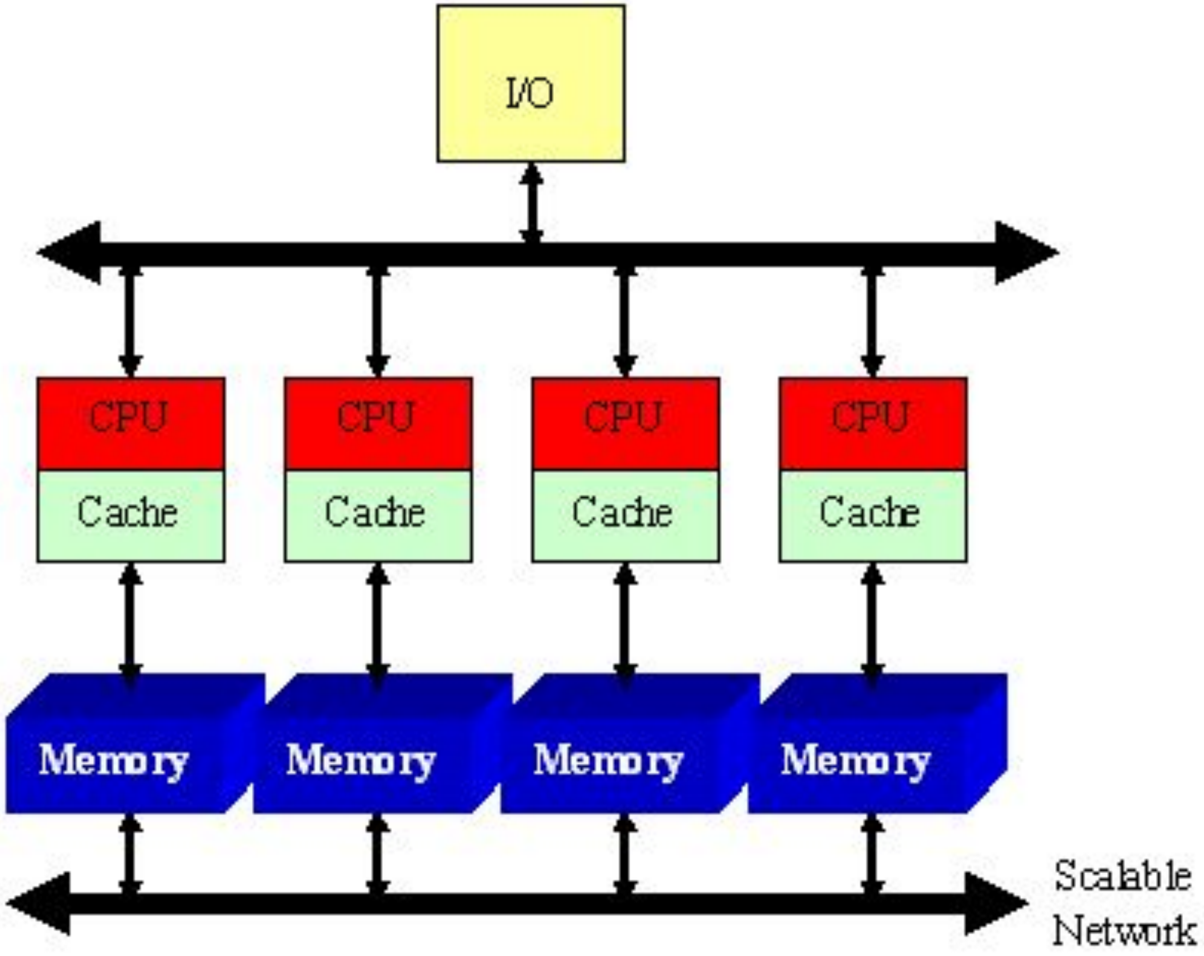
Computer system architecture

- Single-processor systems
  - One general-purpose CPU
  - Might include other device specific CPUs with limited instruction sets
- Multiprocessor systems
  - also called parallel or multicore systems
  - Have multiple CPUs and shared bus (sometimes even clock), memory and I/O devices
  - Advantages:
    - Increased throughput (N processor N times speedup?)
    - Economy of scale (multiple CPUs per box vs. multiple boxes)
    - Increases reliability (Graceful degradation, fault tolerance)

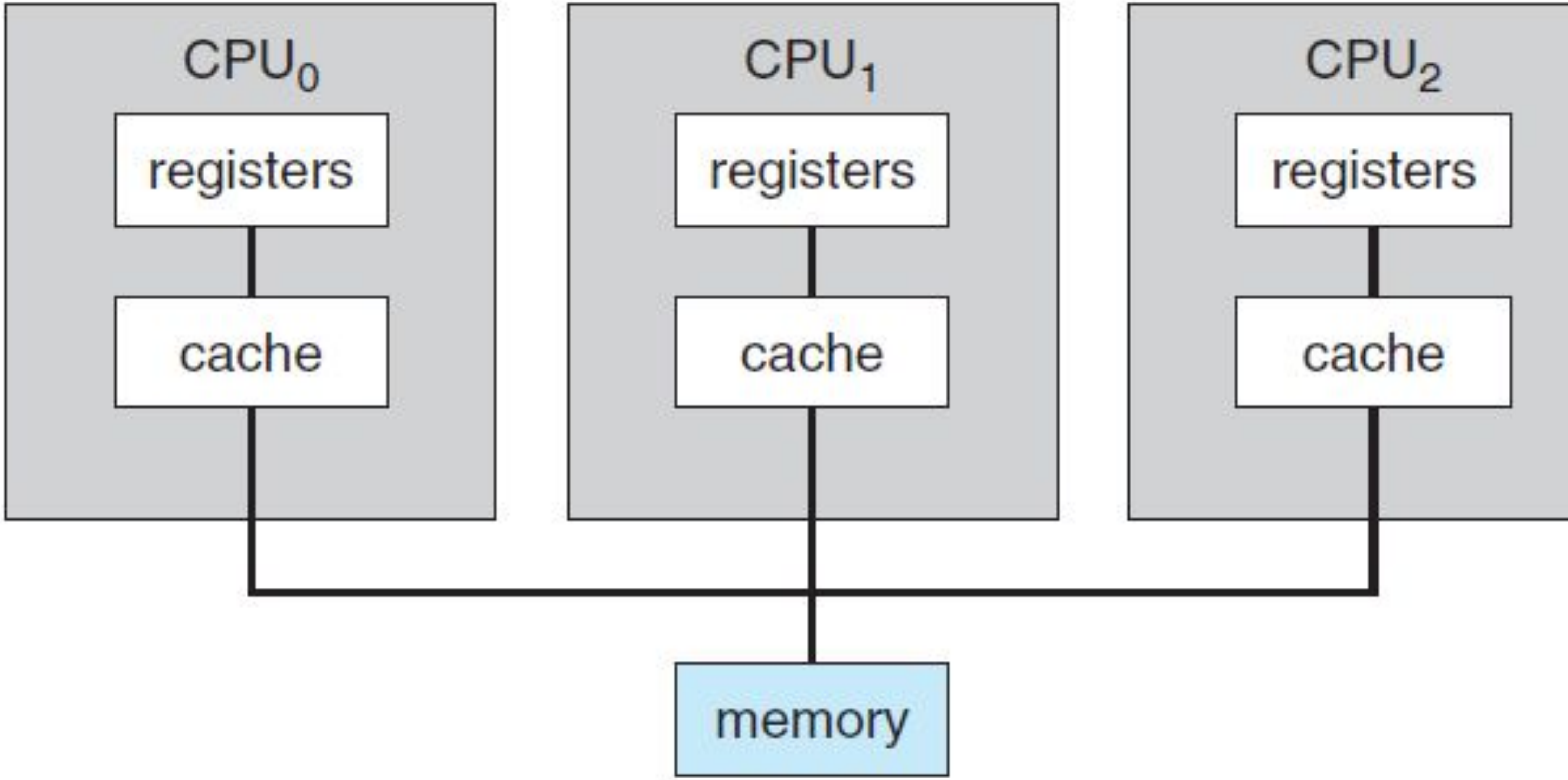
Multiprocessor systems

- Asymmetric multiprocessing
  - One boss + multiple workers
- Symmetric multiprocessing (SMP)
- The difference can be at hardware or software level
- If each CPU has an integrate memory controller, then there is an increase in the address space
  - Results in non-uniform memory access (NUMA)
- Multiple processors on same chip – multicore system

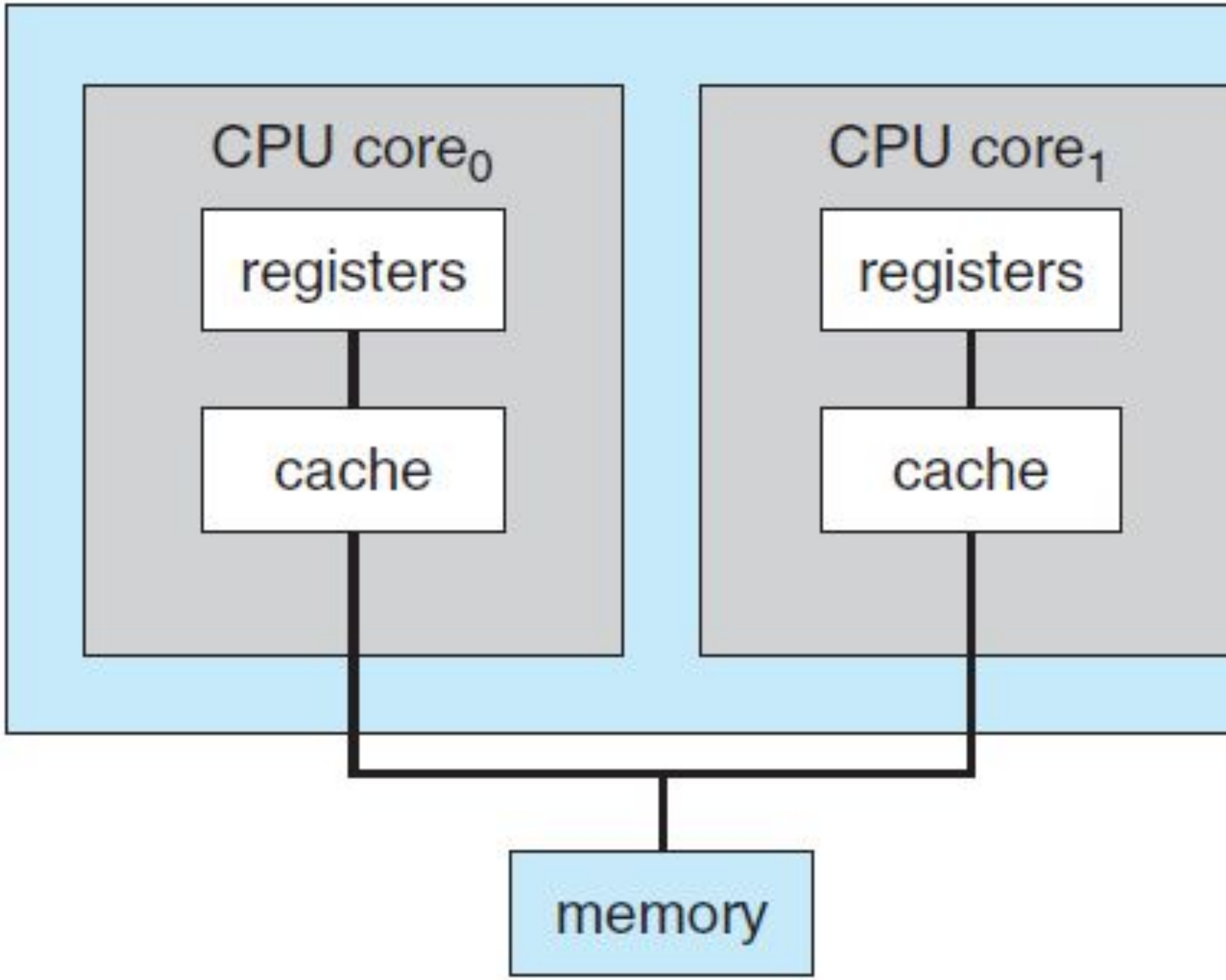
NUMA Architecture



SMP architecture



Dual-core system



Blade servers

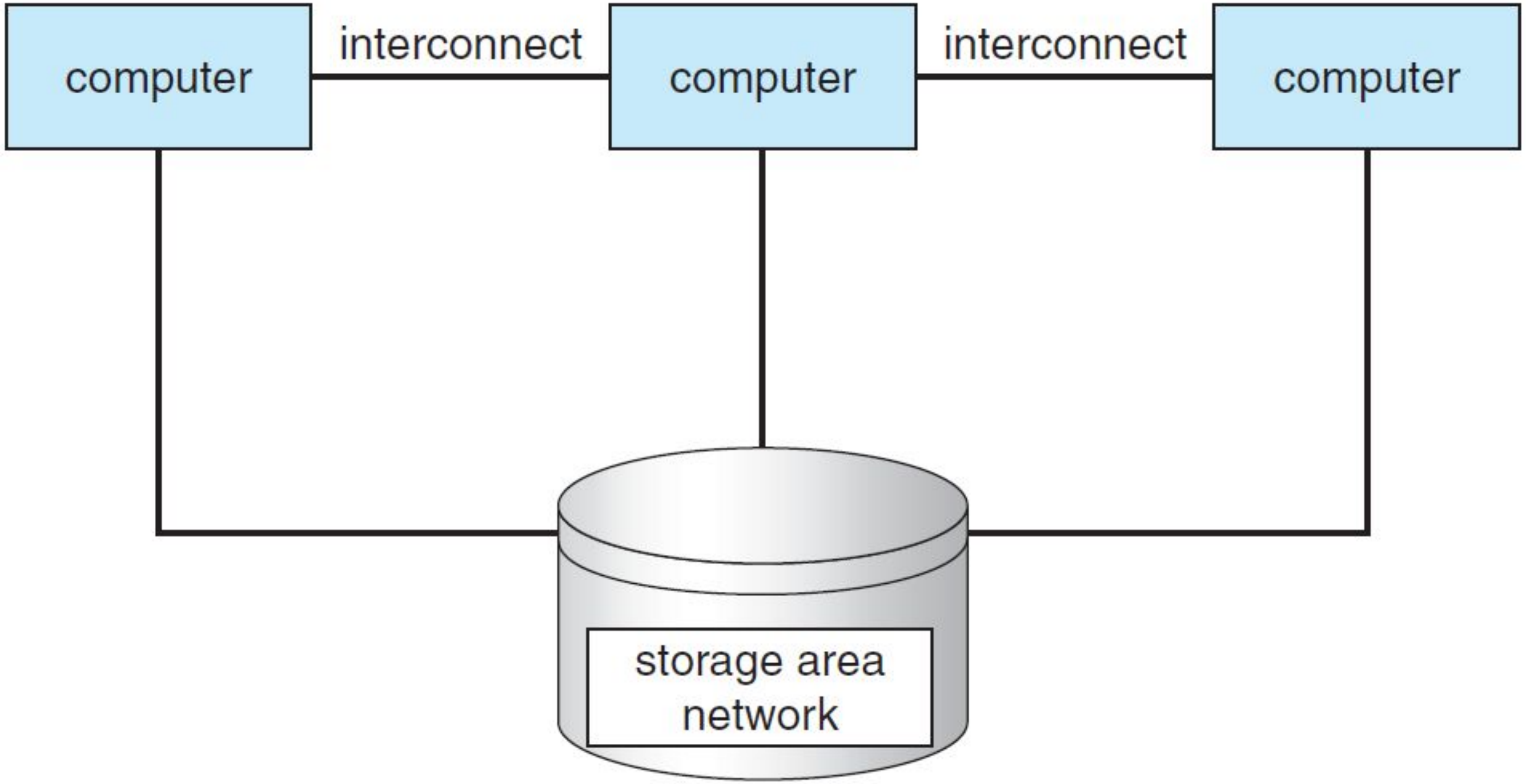
- Several boards (processing, I/O, networking, etc) in one system box
- Some processing boards might be multiprocessor

Clustered Systems

- Loosely coupled processor compared with multiprocessor/multicore
- Nodes connected by a LAN (sometimes WAN)
- Shared storage
- Provide high availability
- Asymmetric clustering
  - Hot-standby mode
- Symmetric clustering
- High performance environment
  - Parallelization of software

Clustered Systems

- Distributed lock manager (DLM)



Operating system structure

- **Multiprogramming** – multiple jobs (programs + data) in main memory taking turns to execute in the CPU and use I/O
    - Switches only occur if some job is waiting for some event
    - no user interaction with the system
- **Job pools** kept on disk awaiting transfer to main memory
- **Time sharing** (**multitasking**) – multiple jobs in memory
    - Switch among jobs occurs very frequently – gives the impression that each job has exclusive ownership of the CPU and I/O devices
    - Assumes an interactive system
    - Short **response time**
    - **Multiuser** environment

Time sharing

- **Process** – a program loaded in memory and executing
- **Job scheduling** – how to select from multiple jobs in the job pool on the disk for loading into [limited] memory
- **CPU scheduling** – how to select a process for execution from ready processes in memory
- **Swapping** – transferring processes between main memory and disk
  - Improves response time
- **Virtual memory** – allows execution of a process that requires memory than is physically available
  - **Logical** vs. **physical memory** separation

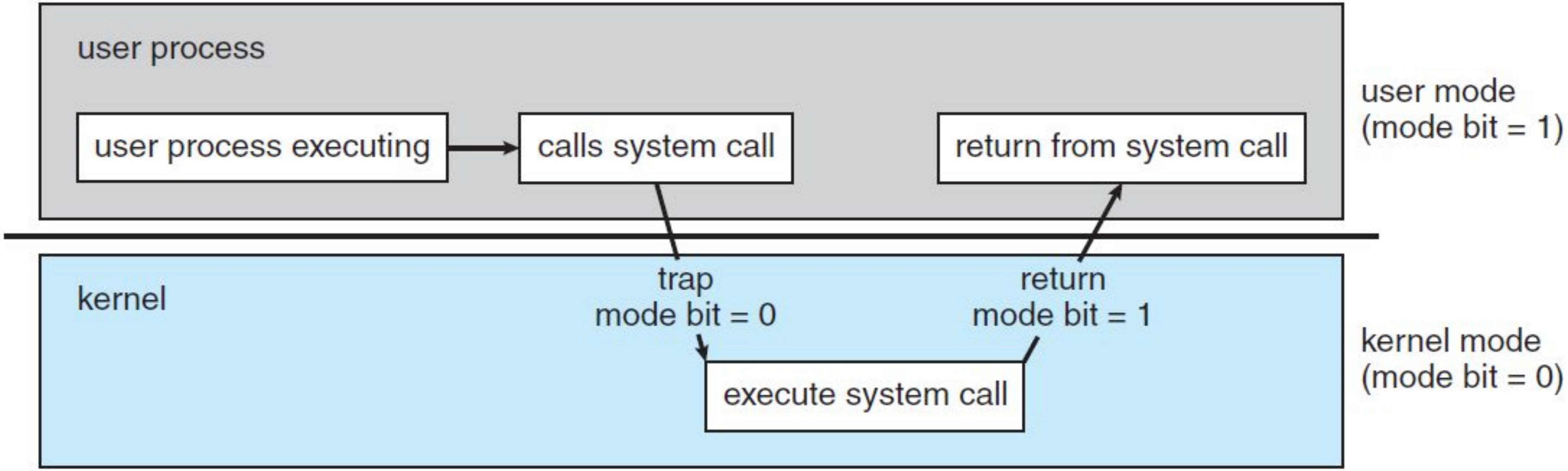
Time sharing

- File system
  - resides on multiple disks
- Disk management
- Protecting resources from inappropriate use
- Job synchronization and communication
  - For orderly execution
- Deadlocks
  - every process waiting for the other

Operating system operations

- **Trap** – software generated interrupt cased by error or specific user request that must be handled by the OS
- How to limit errors/malicious intents in user programs – **protection**
- **Dual-mode and multimode** operation
  - Has to be supported at hardware level
  - For dual mode, CPU must have a mode bit
    - 0 – kernel mode (also called supervisor mode or system mode or privileged mode)
    - 1 – user mode

Dual-mode


user process
user process executing
calls system call
return from system call
user mode
(mode bit = 1)
kernel
trap
mode bit = 0
return
mode bit = 1
execute system call
kernel mode
(mode bit = 0)

Timer

- Used to prevent user programs from using the CPU exclusively
- OS sets timer when handing control to a user process
- OS periodically decrements the counter – based on system clock
- OS terminates the process when timer gets to zero

Processes

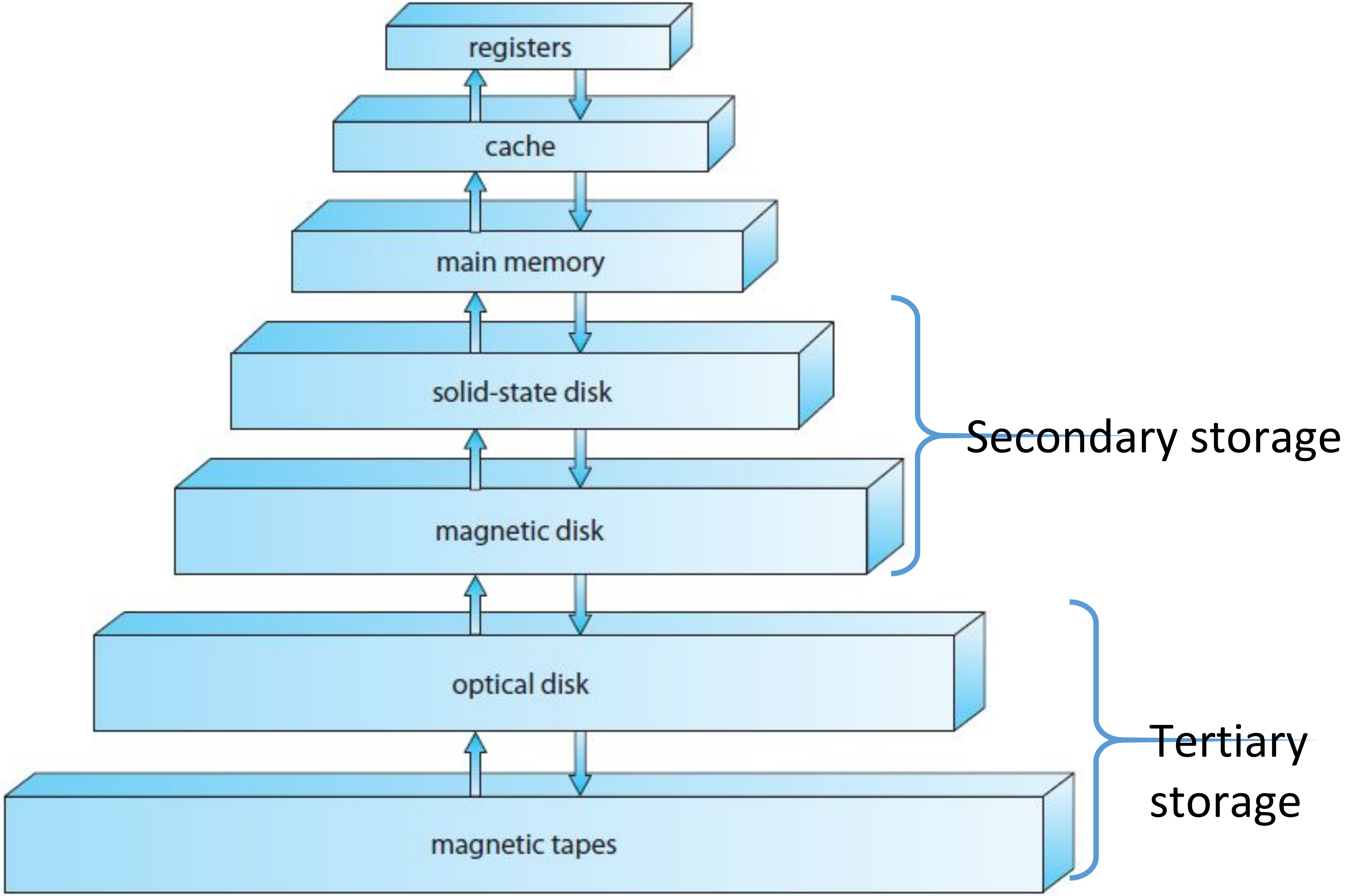
- A **process** is a program in execution
    - A program sitting on the hard disk is not a process!
- A process must share resources with other processes in the system
    - CPU, memory, files, I/O devices
- A **single-threaded process** has one **program counter (pc)**
- A **multithreaded process** has as many program counters as there are parallel threads
- A single program may have several processes each with its own pc
- Instructions in a process are executed sequentially

Process management

- scheduling processes and threads on CPU
- Creating and deleting threads (both system and user)
- Suspending and resuming processes
- Providing mechanism for process synchronization
- Providing mechanism for process communication

Memory management

- Keeping track of memory in use and its users
- Moving process and data in and out of memory
- Allocating and deallocating memory as needed


registers
cache
main memory
solid-state disk
magnetic disk
optical disk
magnetic tapes
Secondary storage
Tertiary
storage

Storage management

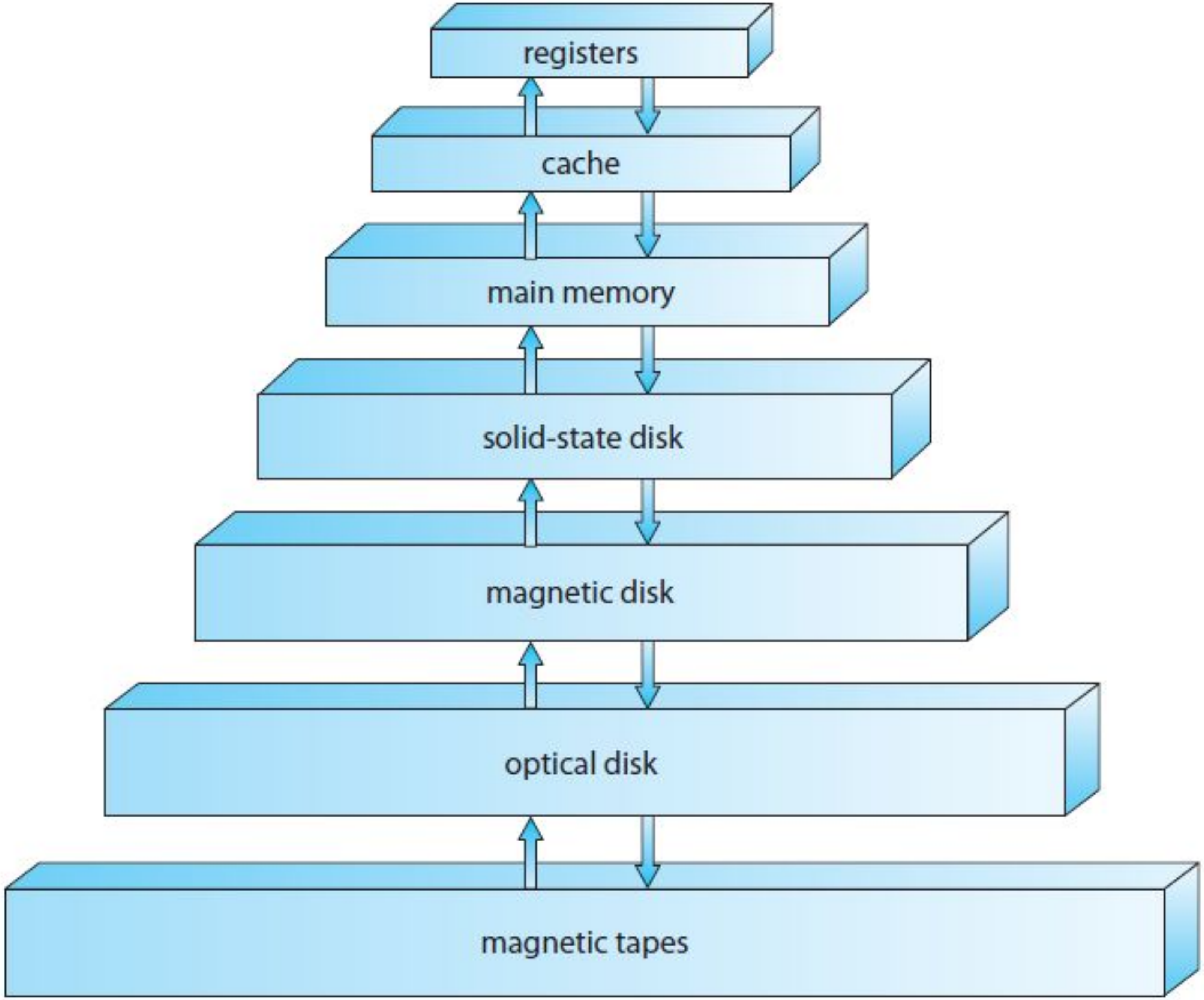
- Providing uniform logical view of information storage
  - File abstraction
- Mapping files into physical storage media
- File system management operations
  - Creating and deleting files
  - Creating and deleting directories
  - Primitive for file and directory manipulation
  - Mapping files onto secondary storage
  - Backing up files

Storage management

- Mass storage management
  - Free space management
  - Disk allocation
  - Disk scheduling
- Tertiary storage management
  - mounting and unmounting
  - allocating and free devices for exclusive use by processes
  - migration of data from secondary to tertiary storage
- Caching
  - Used to mitigate bottleneck in the storage hierarchy

Caching

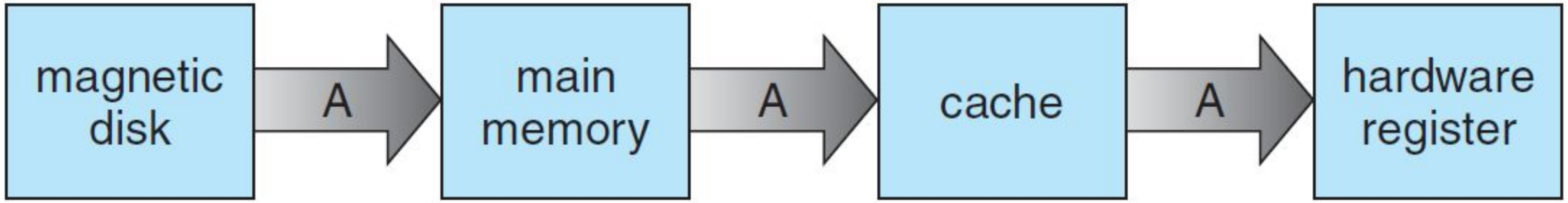
- Cache manag
  - Size
  - Replaceme



Caching

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

Storage management



- Migration of integer A from disk to a register
  - Propagating changes
- What if multiple processes access A?
- What if the multiple processes are running on multiple CPUs, each with it’s own cache?
  - Cache coherence problem
- What if in a distributed system?

I/O devices

- OS must hide peculiarities of I/O devices
- OS must provide
  - memory management – buffering, caching, and spooling
  - general device-driver interface
  - drivers for specific hardware

Protection and security

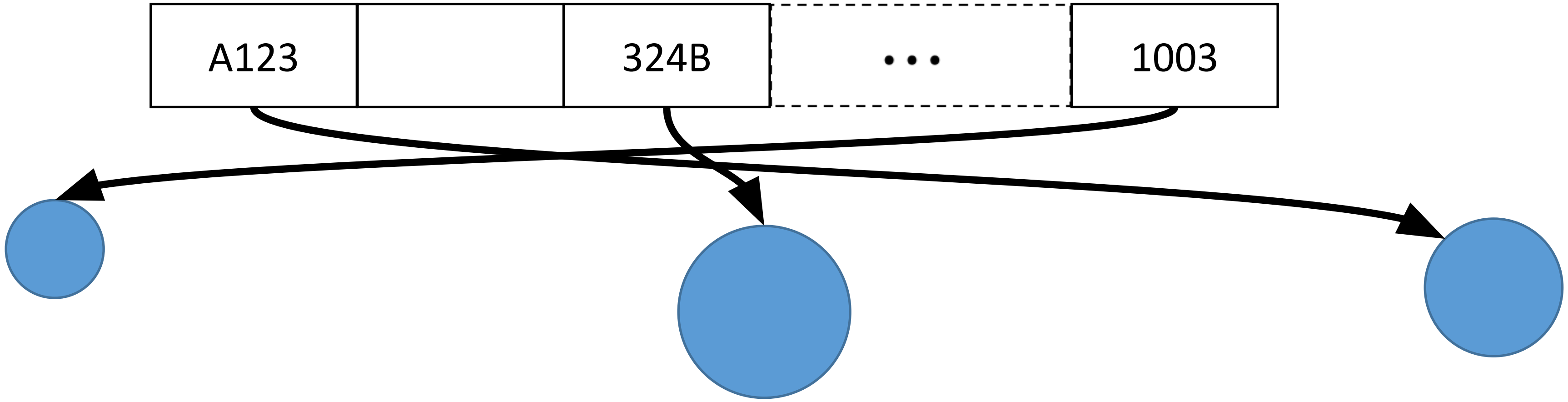
- Protection
    - control access to resources to prevent errors
- Security
    - Defend the system from internal and external attacks
        - viruses, worms
        - denial of service attacks
        - identity theft
        - theft of service – unauthorized access
- User and group IDs

Kernel Data Structures

- Lists
- Stacks
- Queues
- Trees
- Hash Tables/Maps
- Bitmaps

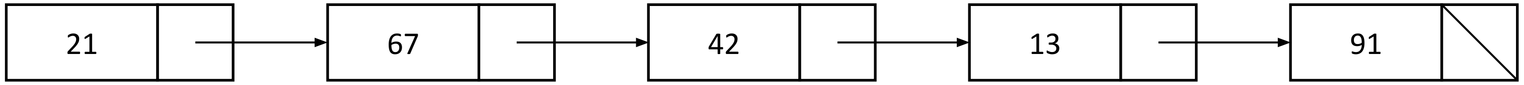
Arrays

- Array – simplest way to organize data
  - Main memory = array of bytes
    - indexing: base-memory-address $+ i$
  - What if want to store more than a byte?
    - Indexing: base-memory-address $+(i \times$ size-in-bytes$)$
  - Only works if each stored item has fixed size
  - What if size of stored values vary?
    - Each item is a fixed-size reference to actual stored data

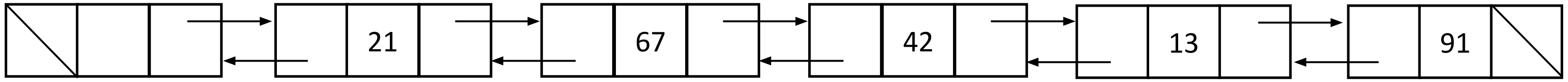


Lists

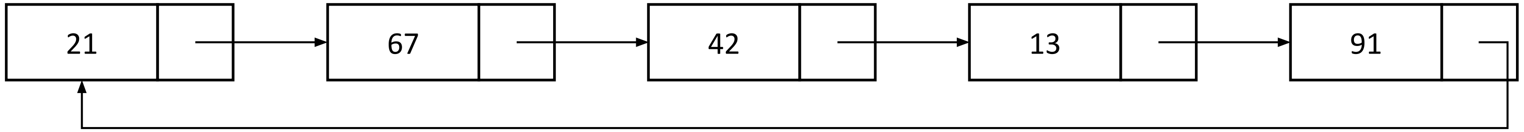
- No direst access (random access) as in arrays
- Elements must be accessed in some order – sequence of elements
- Can be implemented as a linked list



Singly linked list



Doubly linked list



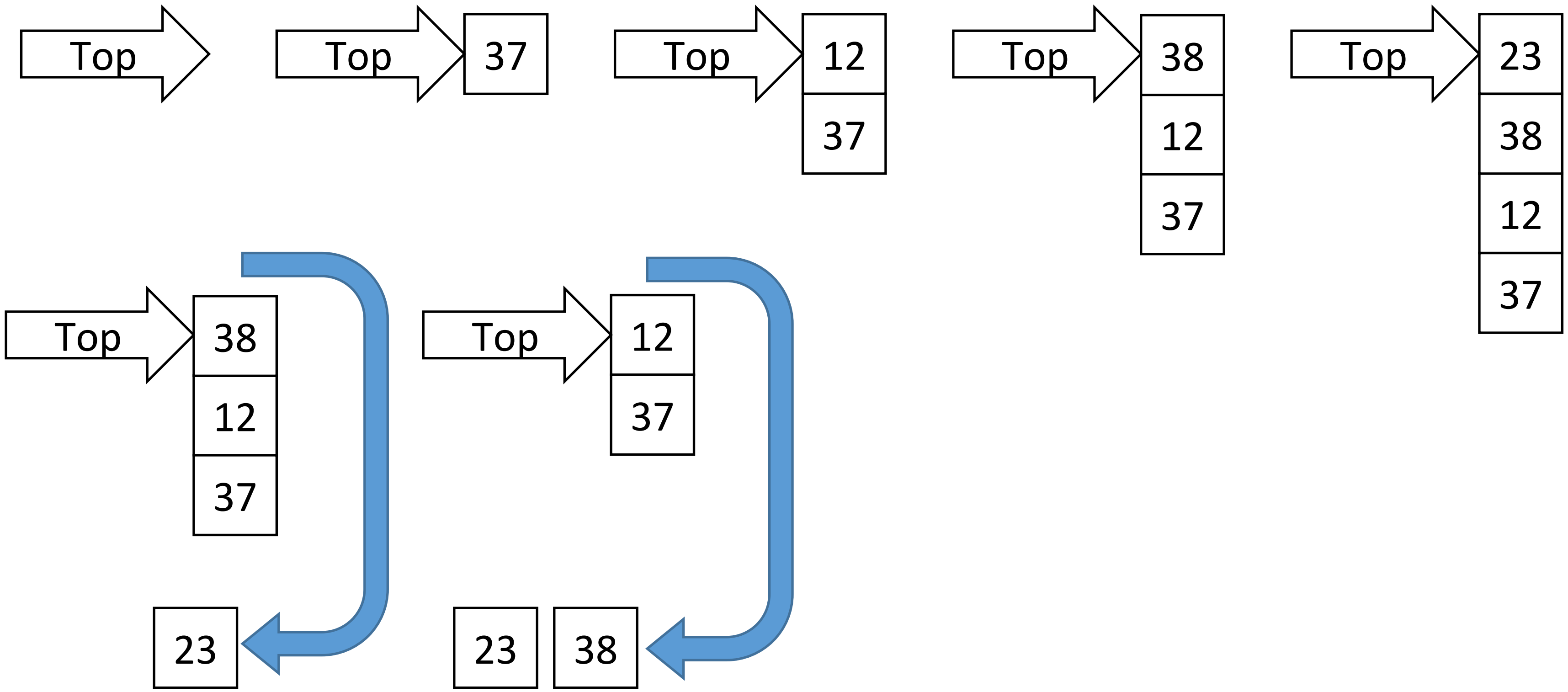
Circular linked list

Stack

- Restricted list
- Access only allowed at one end – top
- Push – adding an element
- Pop – removing an element
- Last-in-first-out access (LIFO)

Stack

- Starting with an empty stack
- Push 37
- push 12
- push 38
- push 23
- pop
- pop

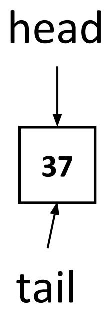


Queue

- Another restricted list
- Access only allowed at the two ends
  - Head – removals
  - Tail – Insertions
- First-in-first-out access (FIFO)

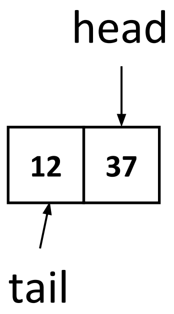
Queues

- Starting with an empty queue
- enqueue 37



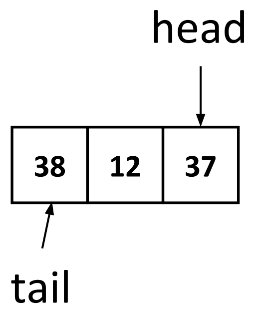
Queues

- Starting with an empty queue
- enqueue 37
- enqueue 12



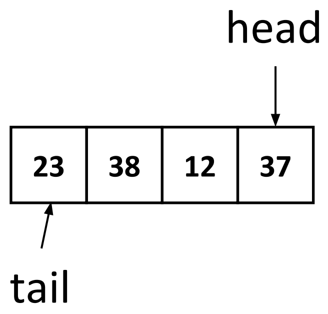
Queues

- Starting with an empty queue
- enqueue 37
- enqueue 12
- enqueue 38



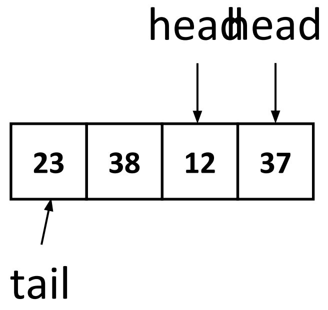
Queues

- Starting with an empty queue
- enqueue 37
- enqueue 12
- enqueue 38
- enqueue 23



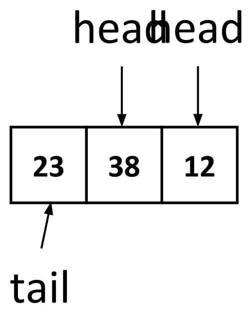
Queues

- Starting with an empty queue
- enqueue 37
- enqueue 12
- enqueue 38
- enqueue 23
- dequeue



Queues

- Starting with an empty queue
- enqueue 37
- enqueue 12
- enqueue 38
- enqueue 23
- dequeue
- dequeue

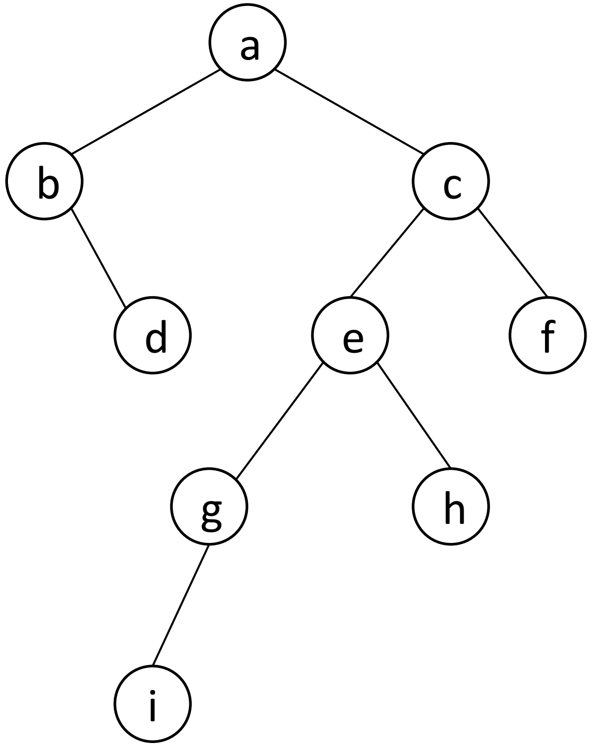


Trees

- Hierarchical arrangement of nodes
    - Parent-child relationship
    - Root
    - Leaf
    - General trees – a node may have zero or more children
    - Binary tree – a node may have up to two children

Notation

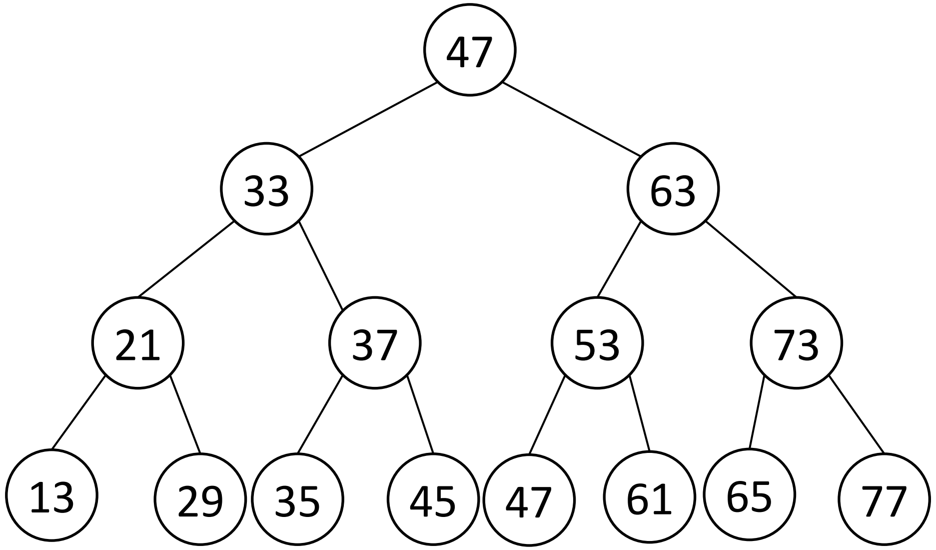
- ***Left child***: the root of the left subtree
  - Examples
    - Left child of node a is b
    - Left child of node e is g
- ***Right child***: the root of the right subtree
  - Examples
    - Right child of node a is c
    - Right child of node e is h
    - Node g does not have a right child
- When left/right subtree does not exist, we say that the subtree is ***empty***



Binary search tree (BST)

- Given any node in the tree
  - left child has a smaller key
  - right child as an equal or greater key
- Searching is $O(n)$ in the worst case
  - $O(\log n)$ in the average case
- Balanced BST
  - Searching is $O(\log n)$ in the worst case

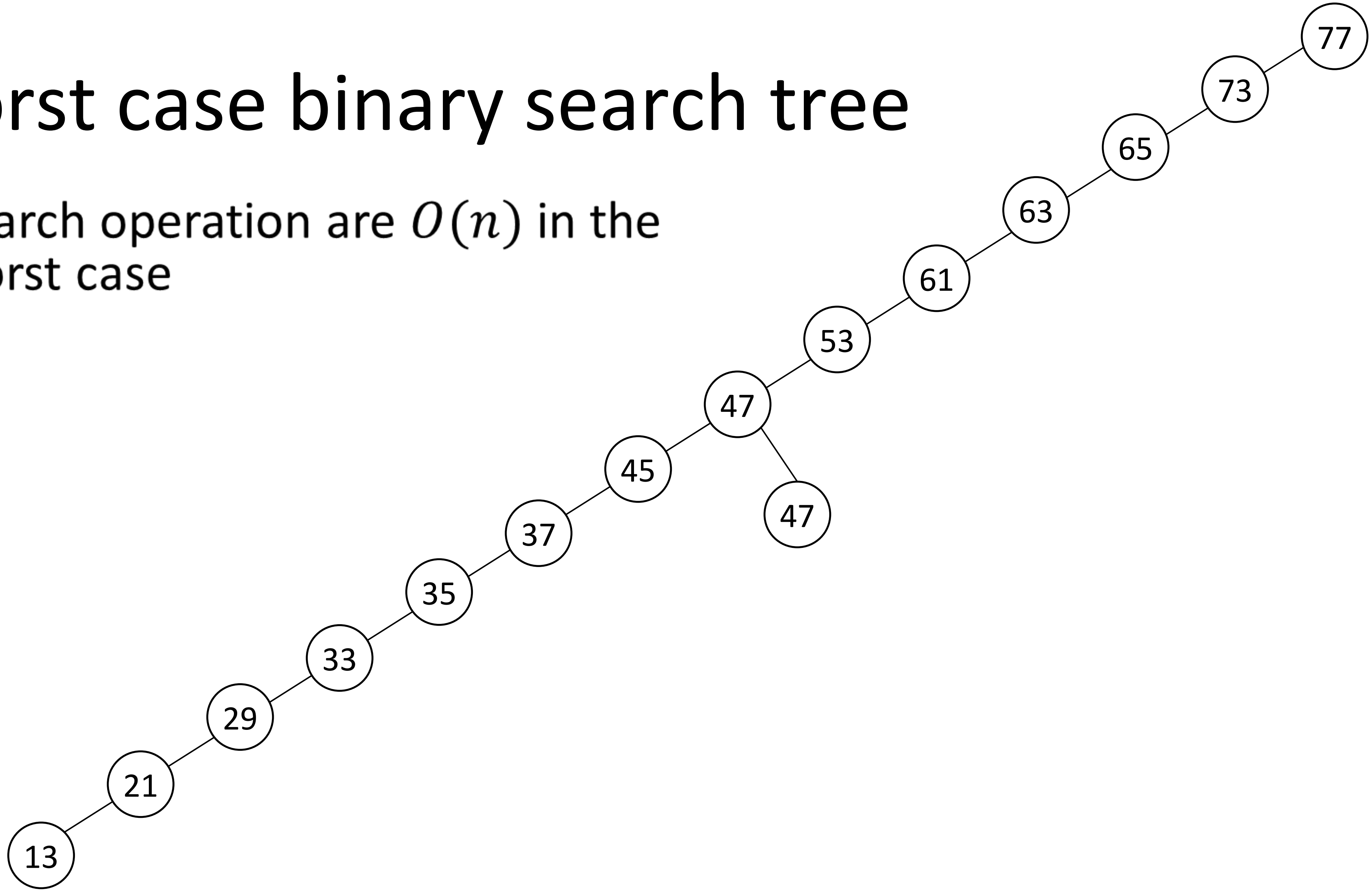
Balanced binary search tree



- All search operation are $O(\log n)$ in the worst case

Worst case binary search tree

- Search operation are $O(n)$ in the worst case



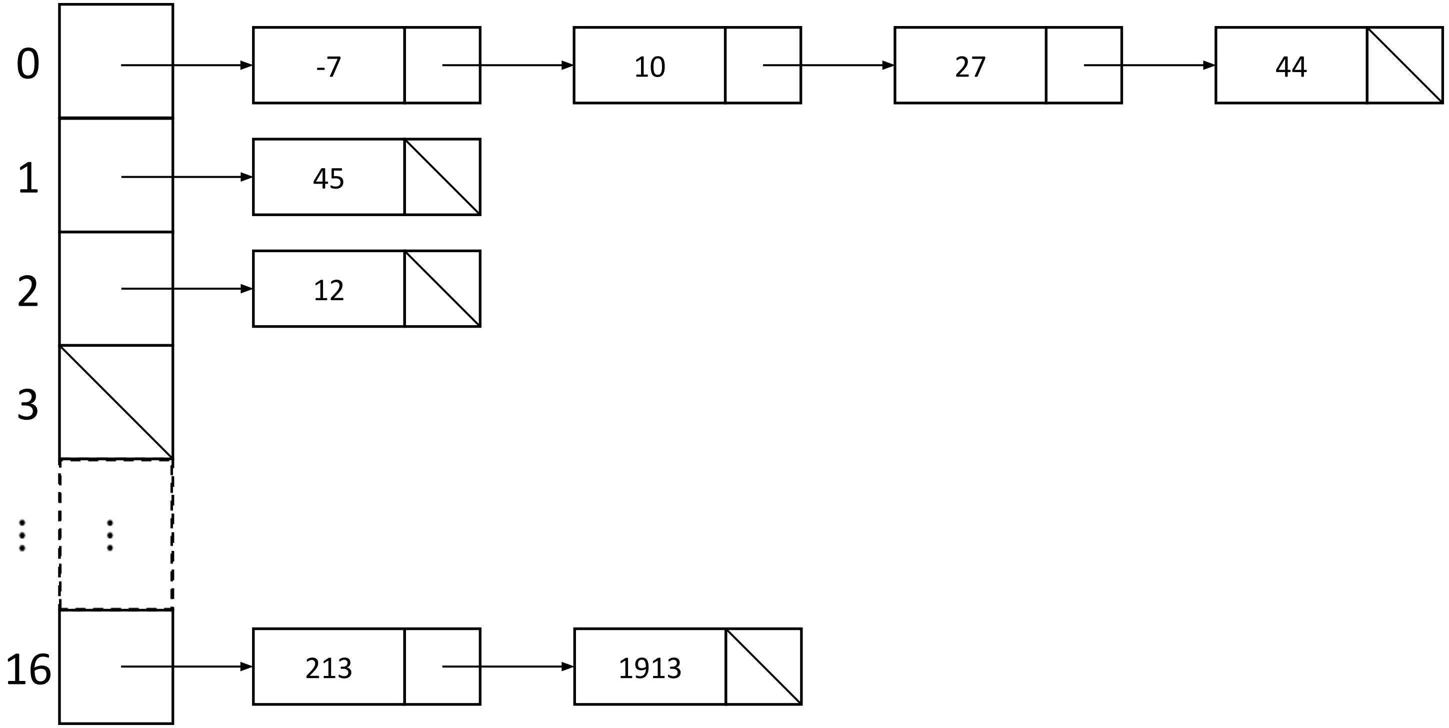
Hash tables

- Hash function – maps any object to an non-negative integer
  - The hash can be used to index a table – hash table
- Example: $hash(k) = k + 7\ mod\ 17$

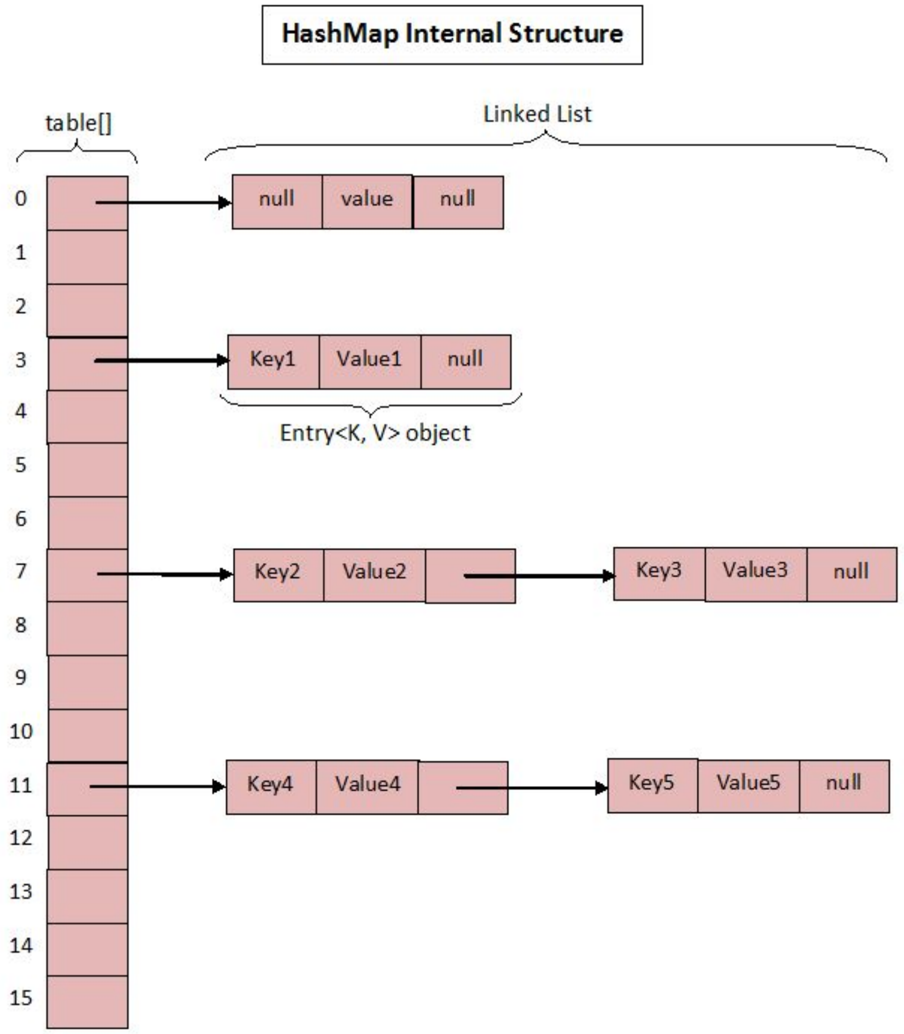
27	45	12	…	213
0	1	2	…	16

- Collisions: multiple objects with the same hash
  - e.g. -7, 10, 27, 44, … hash to 0
  - One solution is to store the object in a linked list at each array index

Hash tables – $hash(k) = k + 7 \; mod \; 17$



Hash maps


HashMap Internal Structure
table[]
Linked List
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
null
value
null
Key1
Value1
null
Entry<K, V> object
Key2
Value2
Key3
Value3
null
Key4
Value4
Key5
Value5
null

Bitmaps

- An array of bits
  - Can be used to represent state of a resource
    - Each index is associated with a resource
    - 0 means free
    - 1 means busy
    - e.g 00101001
      - resources 0, 1, 3, 5 and 6 are free
      - resources 2, 4 and 7 are busy
- More efficient than using the Boolean type – which is typically implemented using a byte (8 bits)

Computing environments - traditional

- Stand-alone general purpose machines
- Not a pure environment due to interconnections of systems – e.g. through the internet
  - Portals
  - Network computers (thin clients)
  - Mobile computers interconnected via wireless networks

Computing environments - Mobile

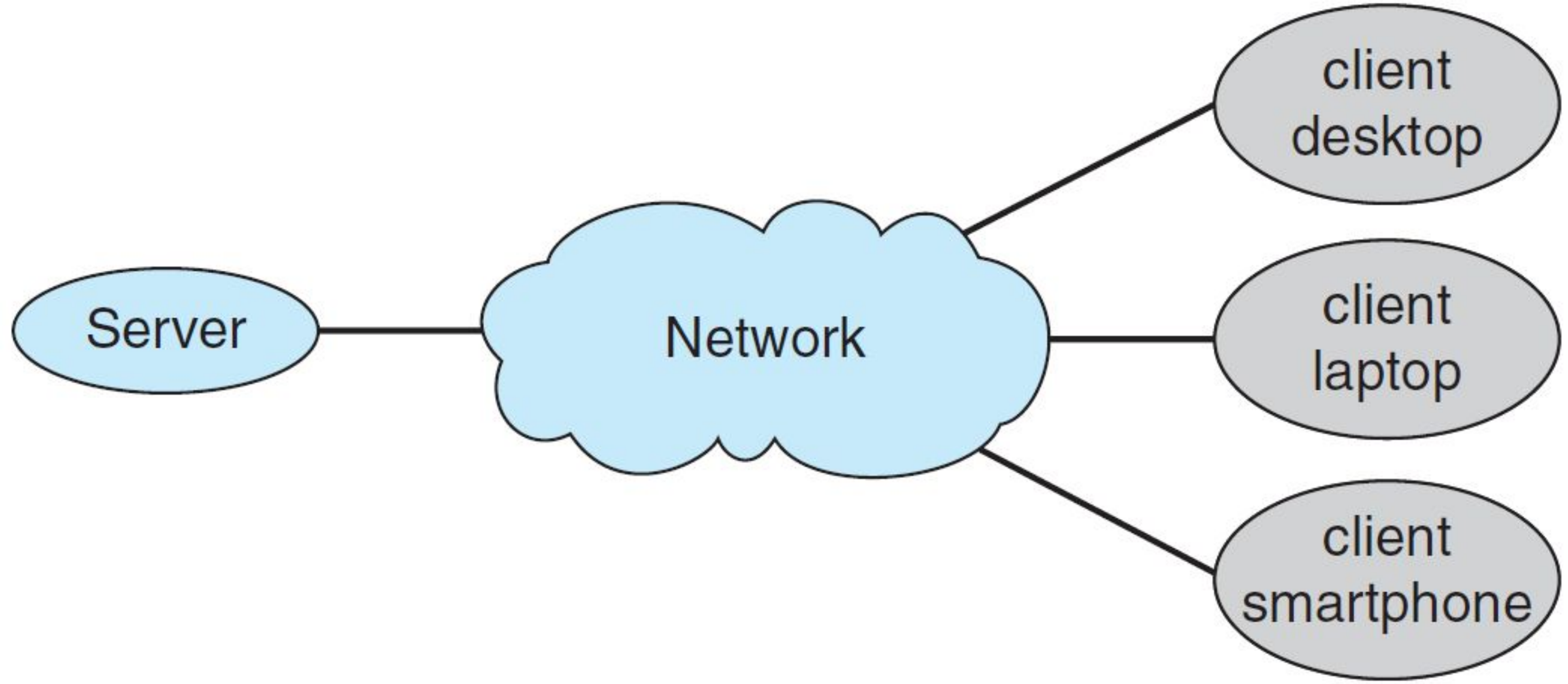
- Handheld devices – smartphones, tablets, etc
- Generally weaker than traditional systems
  - slower CPU, fewer cores, less memory
- Include specialized hardware not available in traditional systems
  - GPS sensors, accelerometers, gyroscope, etc
  - Enable applications and interaction styles not possible on traditional systems
- Use IEEE 802.11 wireless or cellular data networks for connectivity
- Popular OSs: Android by Google and iOS by Apple

Computing environments - Distributed

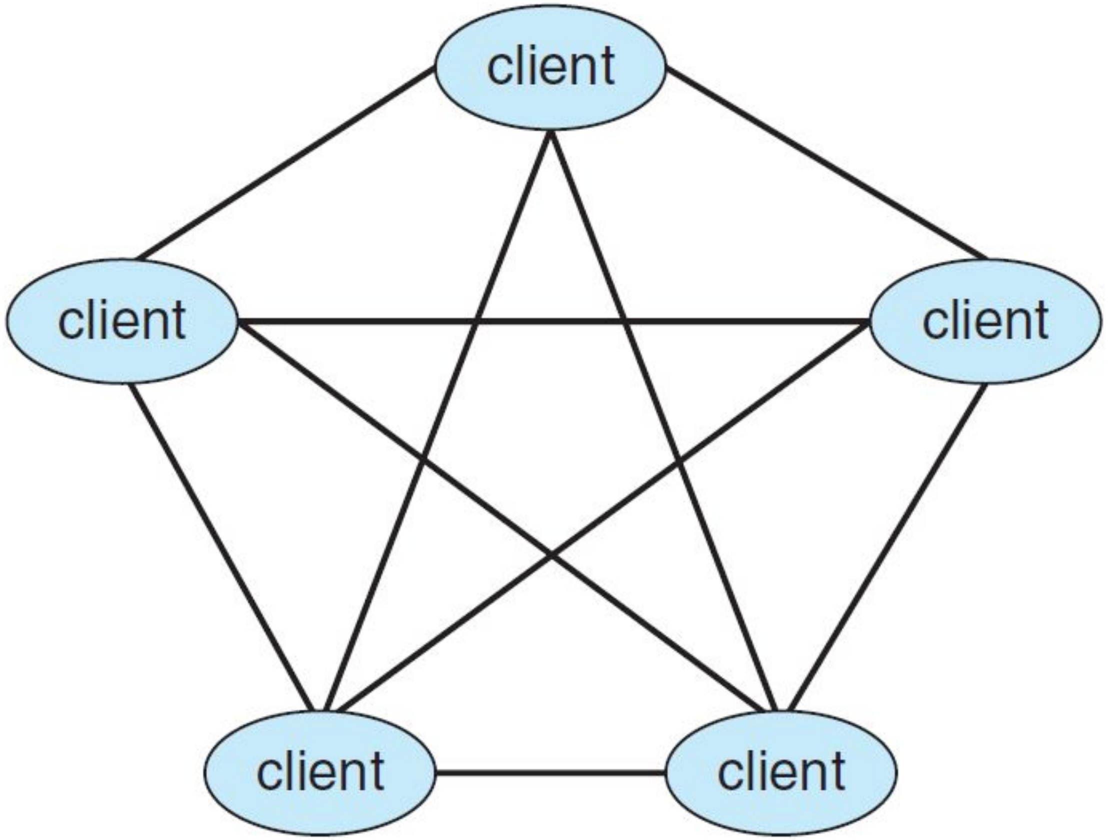
- Collection of separate, possibly heterogeneous, networked systems
- TCP/IP – the most common protocol
- Network scales
  - Personal area network (PAN) – infrared, Bluetooth
  - Local area network (LAN) – wires, fibers, IEEE 802.11
  - Wide area network (WAN) – wires, fibers, satellites, cellular, microwaves, etc
  - Metropolitan are network (MAN)
- Network OS
  - Facilitate communication
  - Illusion of a single system

Computing environments – client-server

- Clients make service requests
- Servers satisfy the request
- Compute
- Files serv



Computing environments – peer-to-peer



Computing environments – peer-to-peer

- No distinction between clients and servers
  - both roles at each node
- Nodes must join the P2P network
- May have centralized lookup service on the network
- Or a node may broadcast service request – discovery protocol
- Examples
  - File sharing: Napster, Gnutella
  - Communication: Skype

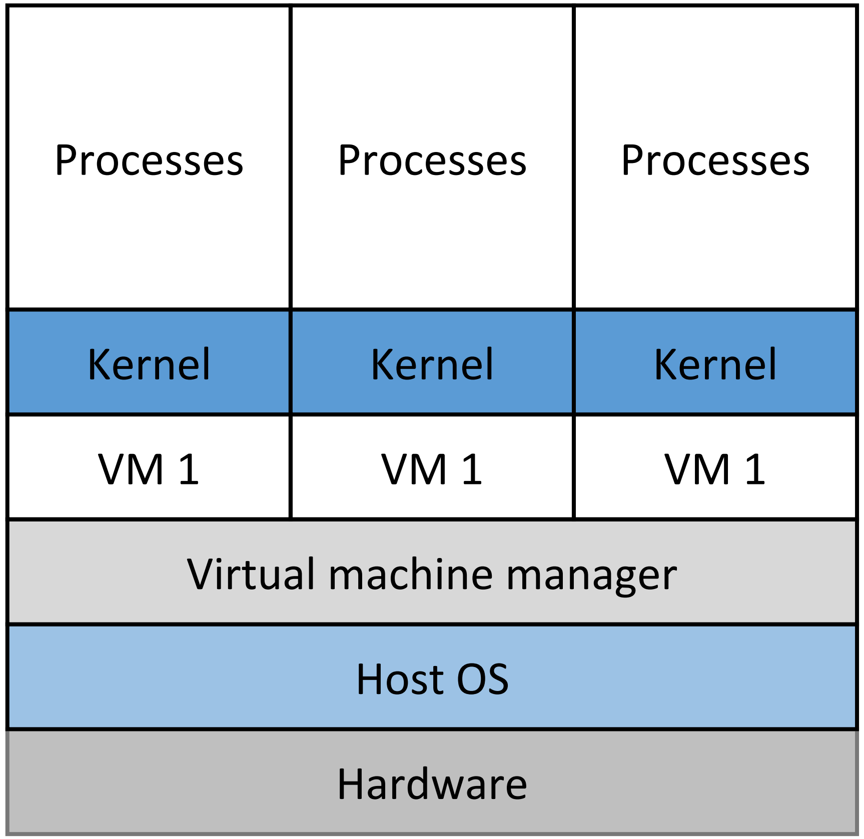
Computing environments – virtualization

- Allows an OS to run applications within another OS
- Emulation – source CPU different from target CPU
  - Slowest
  - Used in interpreted programming languages – e.g. Java and its JVM
- Virtualization – the other OS (guest OS) is natively compiled for the CPU
  - Virtual machine manager (VMM)
    - Can run on a host OS
    - Or can be the host

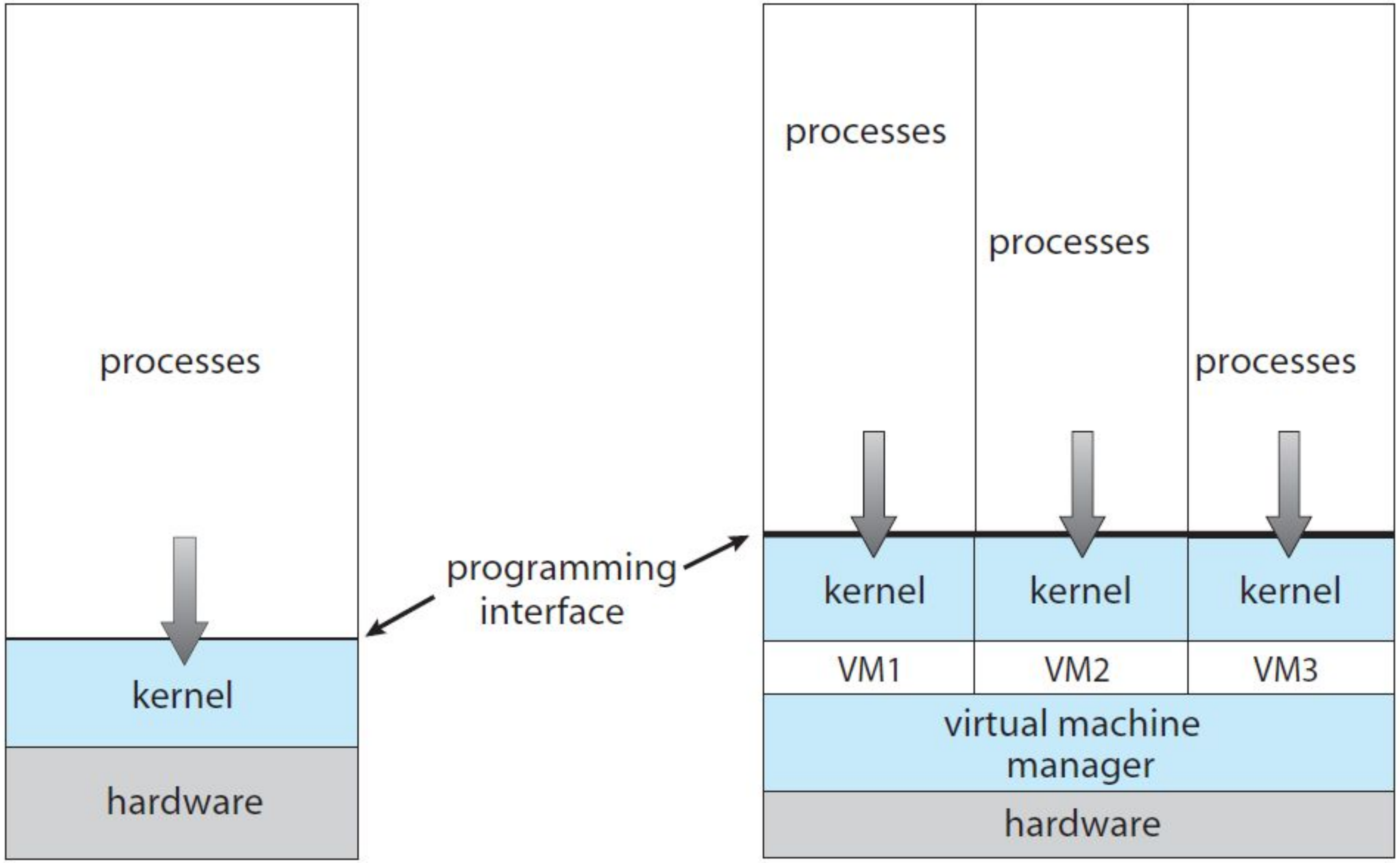
Computing environments – virtualization

- Use cases
  - Exploration
  - Compatibility
  - Development apps for multiple OS
  - Testing cross-platform apps without having multiple systems
  - Executing and managing compute environments in data centers

Computing environments – virtualization



Computing environments – virtualization



Computing environments – cloud computing

- Deliver computing, storage and apps across a network
  - based on virtualization
- Types
  - public cloud – accessible via internet, at a fee
  - private cloud – private company network
  - hybrid cloud
  - Software as a service (SaaS)
  - Platform as a service (PaaS)
  - Infrastructure as a service (IaaS)
- Cloud compute environments: traditional OSs + VMMs + cloud management tools

Computing environments – Real-time embedded systems

- Most prevalent form of computing
- Real-time – stringent time constraints
  - Well-defined and fixed
- Some do not have OSs, some have

Open-source OSs

- Distributed with source code
- GNU General Public Licence (GPL)
- Examples:
  - GNU/Linux
  - BSD UNIX (including the core of Mac OS X)