Chap. 1) Introduction

경희대학교 컴퓨터공학과

방 재 훈

jhb@oslab.khu.ac.kr

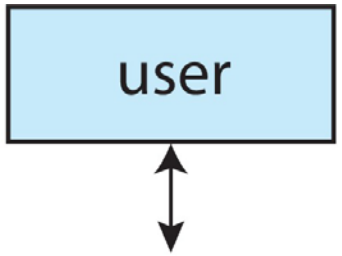
What is an Operating System?

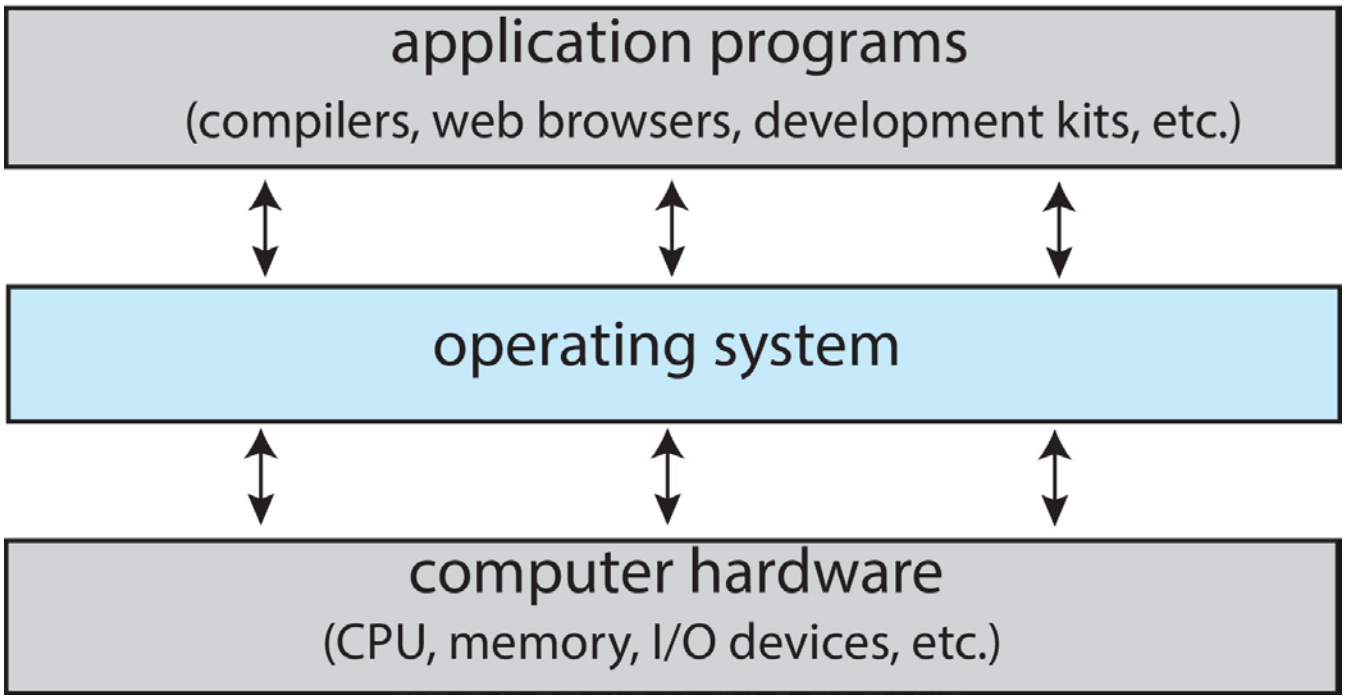
- A program that acts as an intermediary between a user of a computer and the computer hardware

- Operating system goals:
  - ✓ Execute user programs and make solving user problems easier
  - ✓ Make the computer system convenient to use

- Use the computer hardware in an efficient manner

Abstract View of Computer System Components





Operating System Definitions

- Resource allocator – manages and allocates resources
- Control program – controls the execution of user programs and operations of I/O devices
- Kernel – the one program running at all times (all else being application programs)

What is an OS?

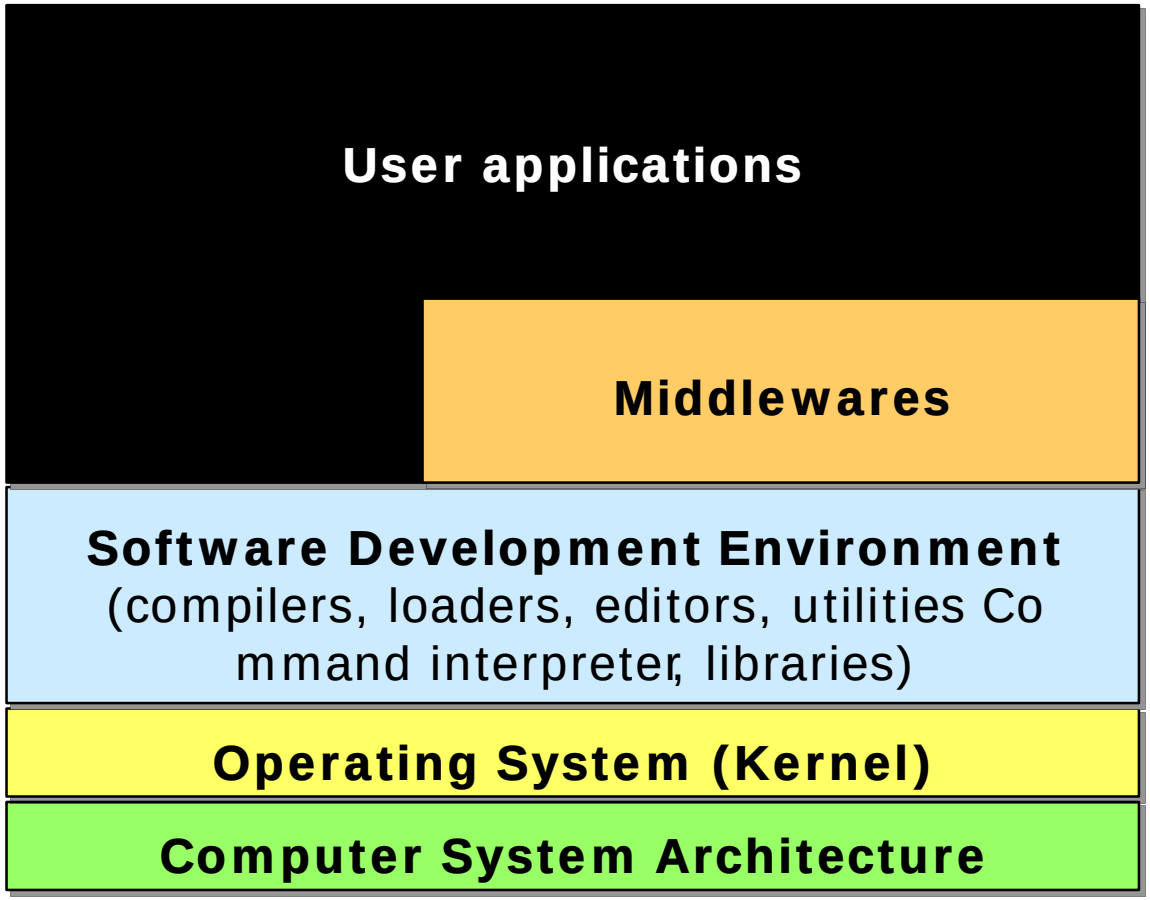
- OS is a resource manager
  - ✓ Abstraction
  - ✓ Sharing
    - Time multiplexing
    - Space multiplexing
  - ✓ Protection
  - ✓ Fairness
  - ✓ Performance

Resources

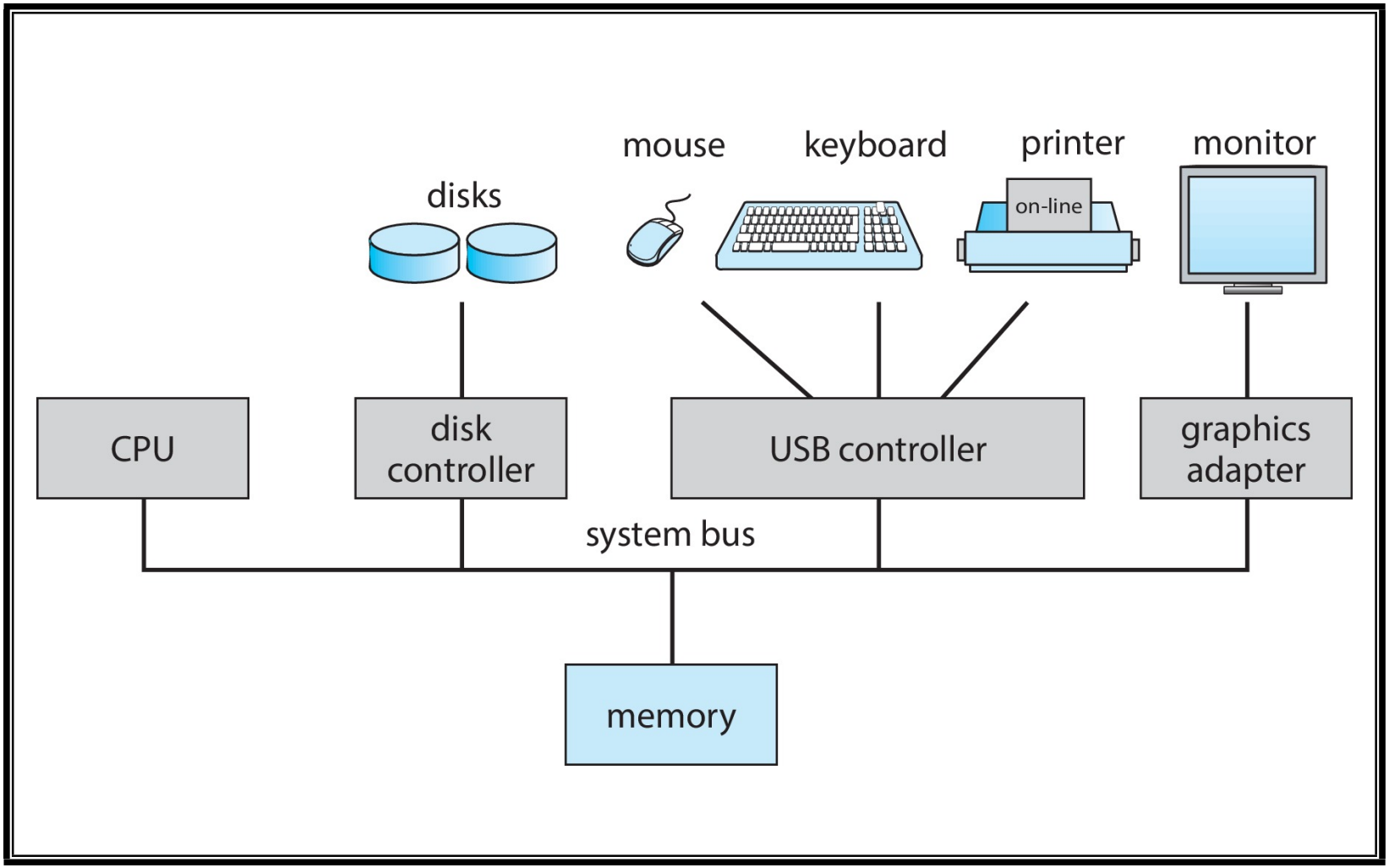
- CPU
- Memory
- I/O devices
- ...

- OS provides the program execution environment

System Software Layers



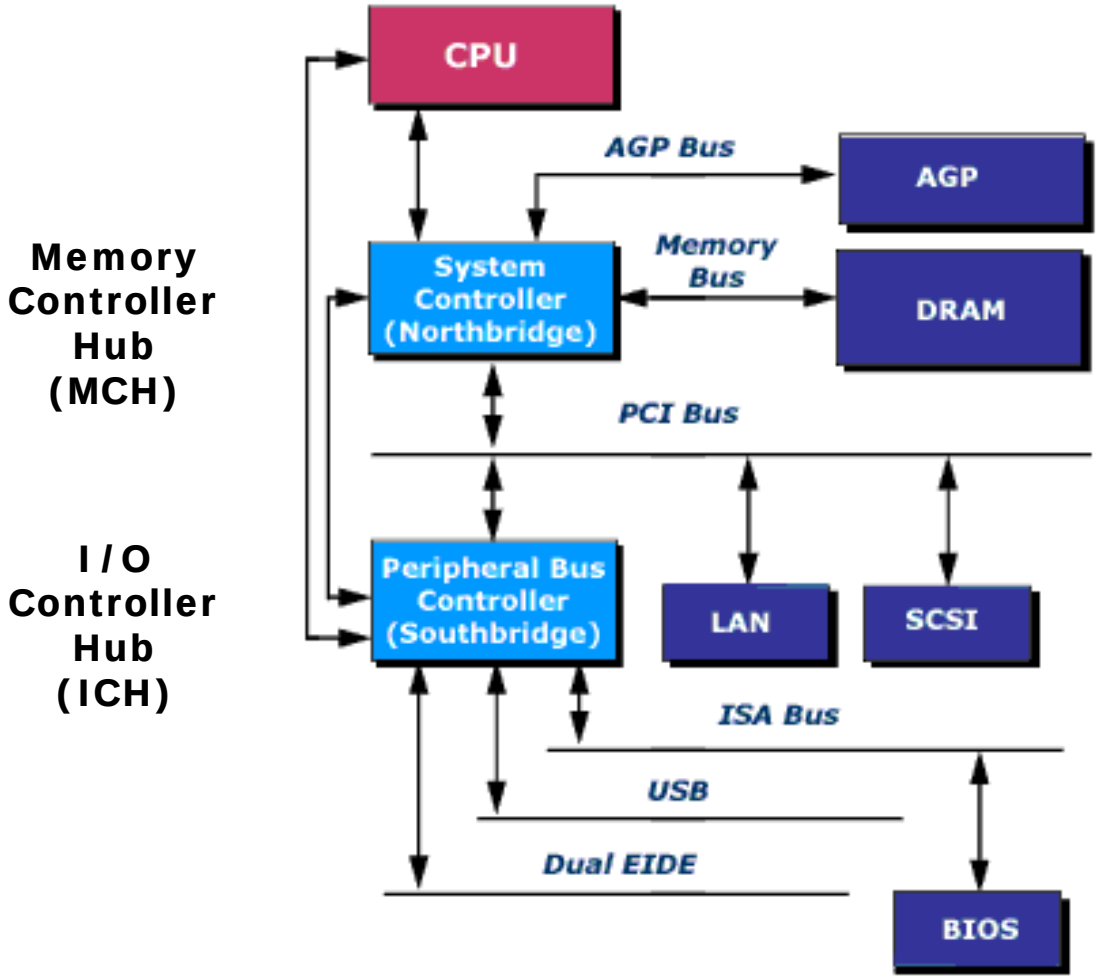
Computer-System Architecture



Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
  - ✓ Cf) DMA (Direct Memory Access)
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an *interrupt*

Modern PC Architecture

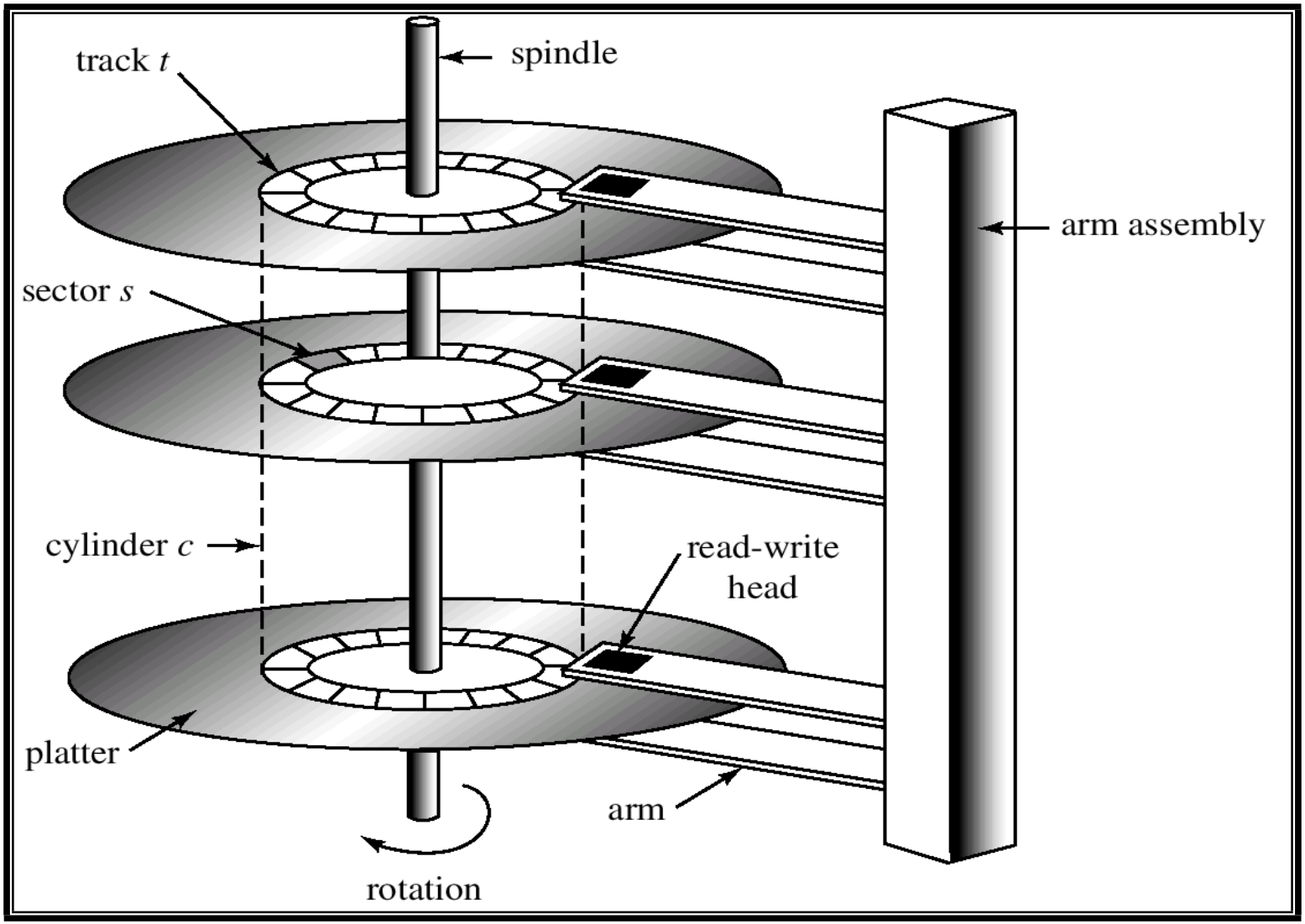


- Registers
  - ✓ Program Counter (PC)
  - ✓ Instruction Register (IR)
  - ✓ Program Status Word (PSW)
  - ✓ General-purpose registers
- Instruction Set Architecture
  - ✓ RISC vs. CISC
  - ✓ Intel, SPARC, MIPS, PowerPC, ARM, Alpha, …
- Pipelining
  - ✓ Fetch, Decode, Execute, Write Back, etc.
- Instruction-Level Parallelism(ILP)
  - ✓ Superscalar vs. VLIW
  - ✓ Simultaneous Multithreading

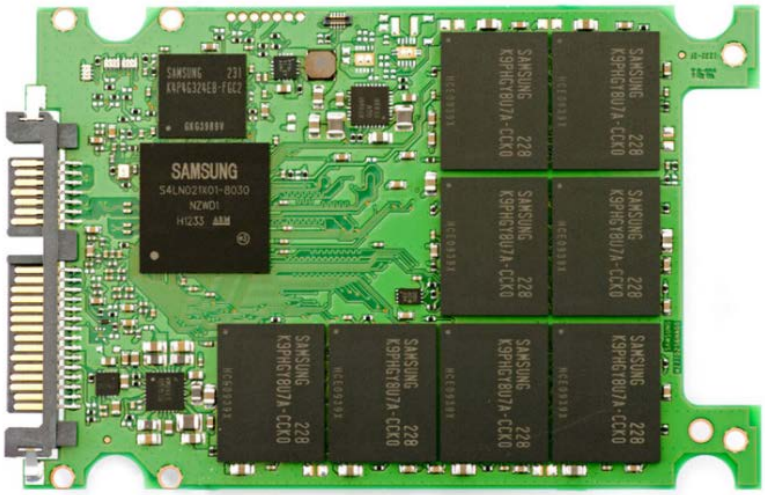
Storage Structure

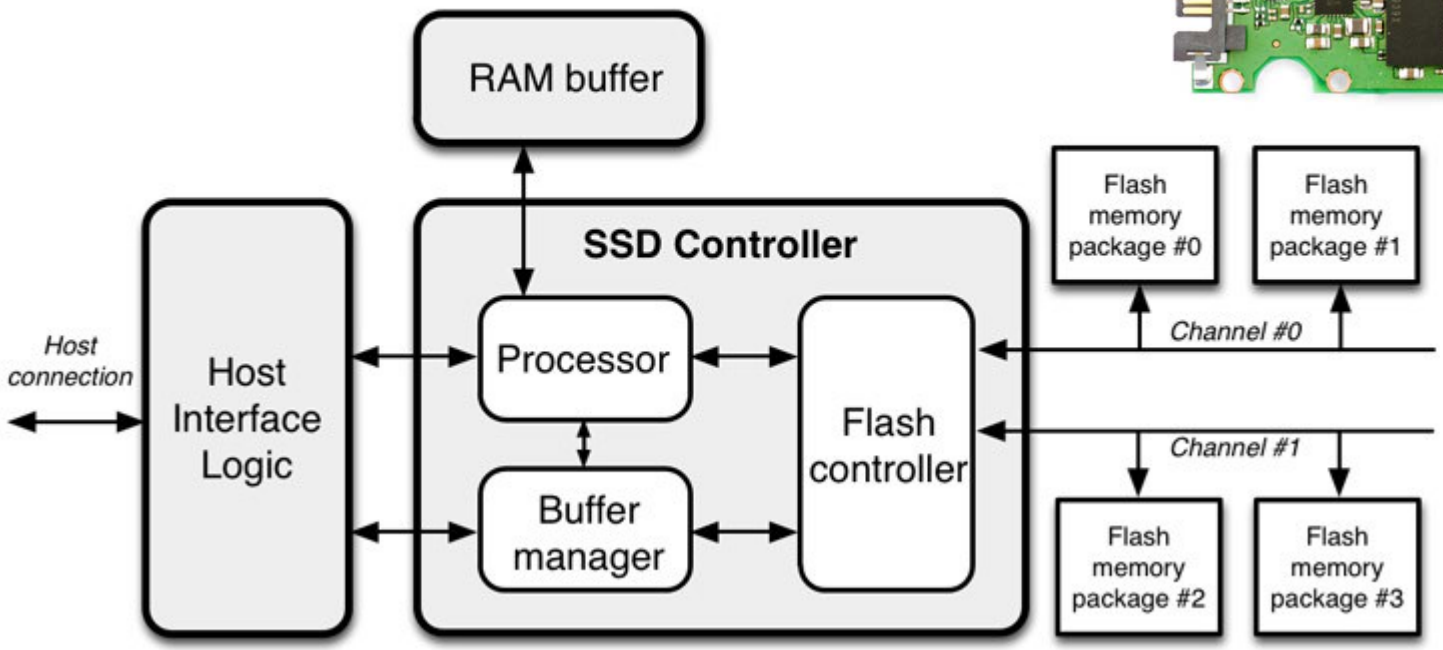
- Main memory – only large storage media that the CPU can access directly
  - ✓ von Neumann architecture vs. Harvard architecture
  - ✓ SRAM vs. DRAM
  - ✓ DDR, QDR, RDRAM
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
  - ✓ Disk surface is logically divided into *tracks*, which are subdivided into *sectors*
  - ✓ The *disk controller* determines the logical interaction between the device and the computer

Moving-Head Disk Mechanism

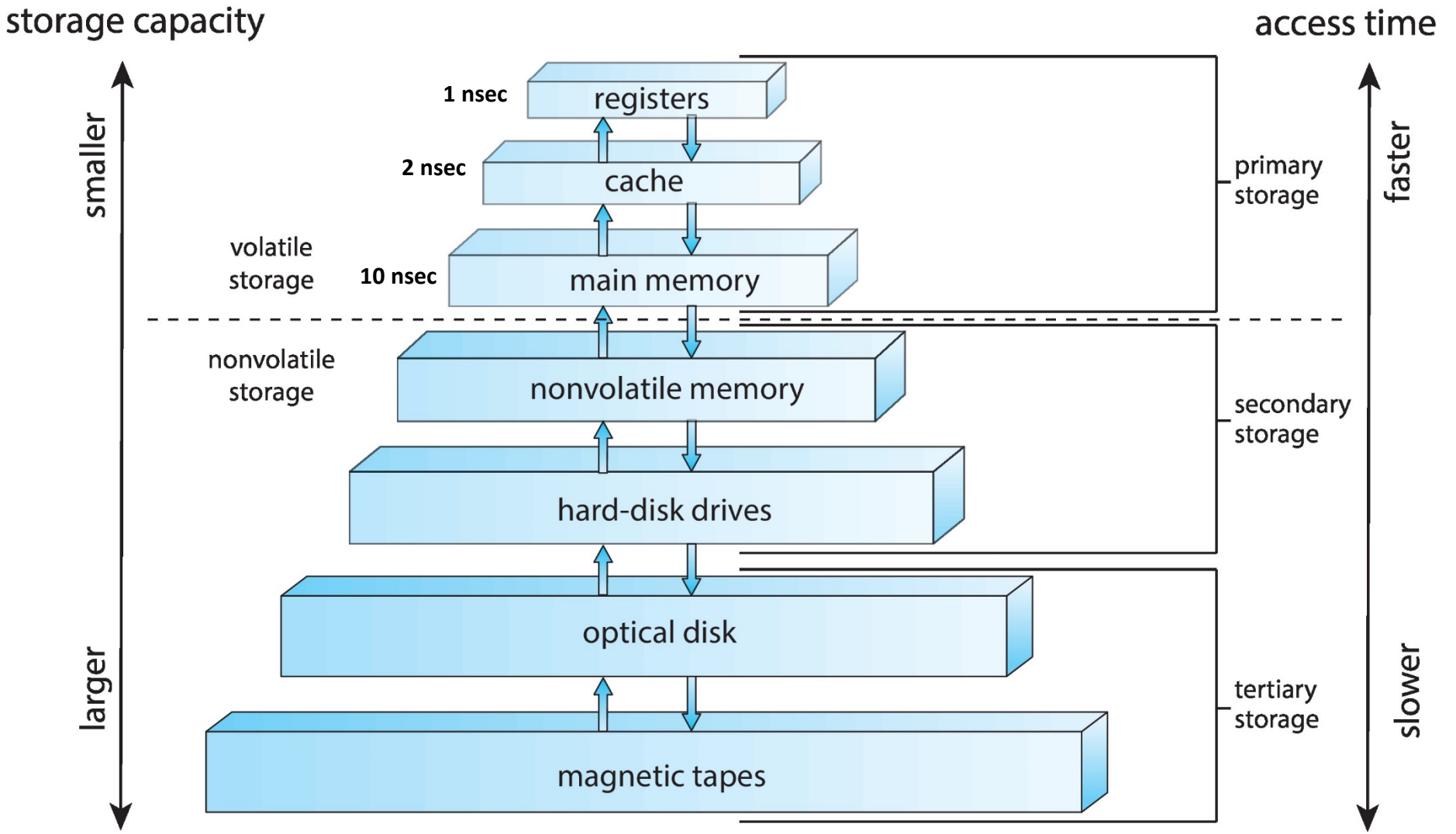


Compact Flash card internals (SSD)





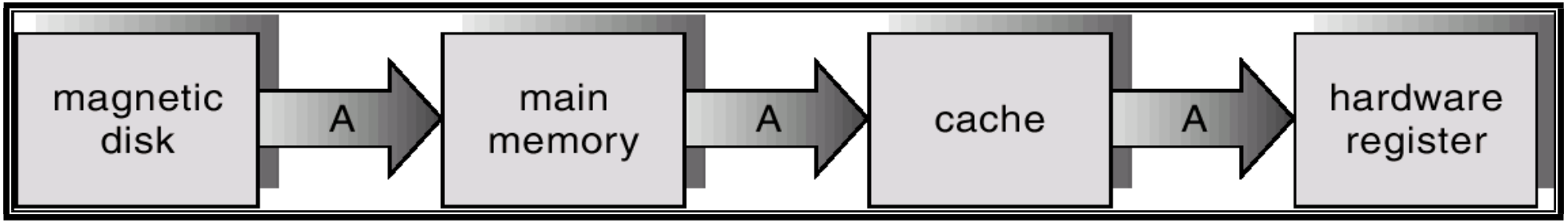
Storage-Device Hierarchy



Caching

- Use of high-speed memory to hold recently-accessed data
- Requires a *cache management* policy
  - ✓ Write-through vs. Write-back
- Caching introduces another level in storage hierarchy. This requires data that is simultaneously stored in more than one level to be *consistent*
  - ✓ Cache coherency

Migration of A From Disk to Register

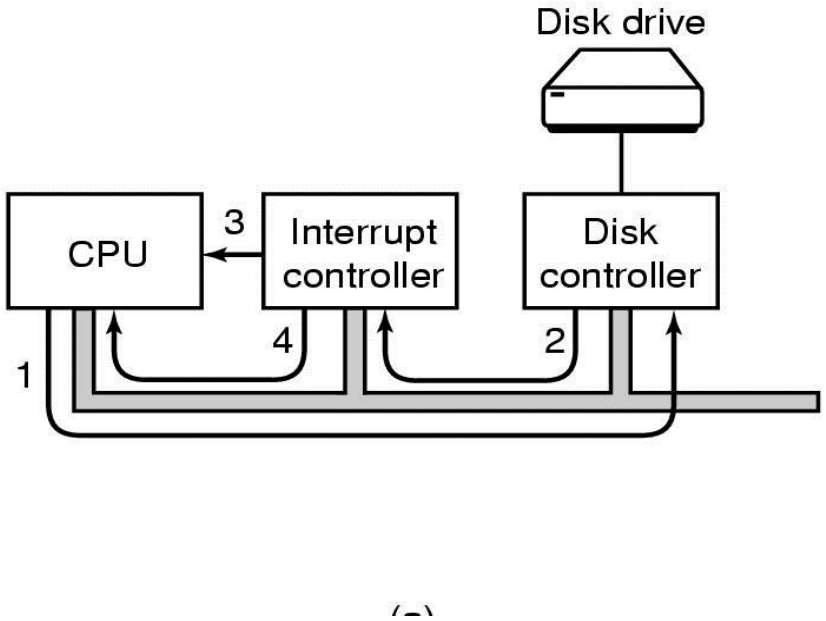


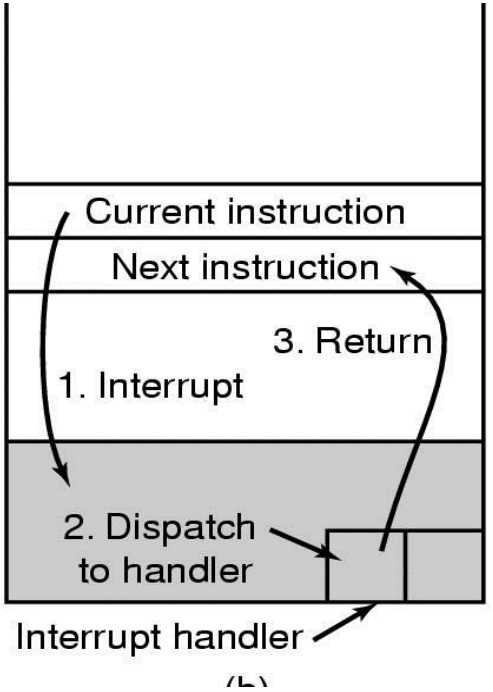
I/O Structure

- After I/O starts, control returns to user program only upon I/O completion
  - ✓ Wait instruction idles the CPU until the next interrupt
  - ✓ Wait loop (contention for memory access)
  - ✓ At most one I/O request is outstanding at a time, no simultaneous I/O processing

- After I/O starts, control returns to user program without waiting for I/O completion
  - ✓ *System call* – request to the operating system to allow user to wait for I/O completion
  - ✓ *Device-status table* contains entry for each I/O device indicating its type, address, and state
  - ✓ Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt

- How does the kernel notice an I/O has finished?
  - ✓ Polling
  - ✓ Hardware interrupt





I/O Control (Cont'd)

Device interrupts

↓

CPU stops current operation, switches to kernel mode, and saves current PC and other state on kernel stack

↓

CPU fetches proper vector from vector table and branches to that address (to routine to handle interrupt)

↓

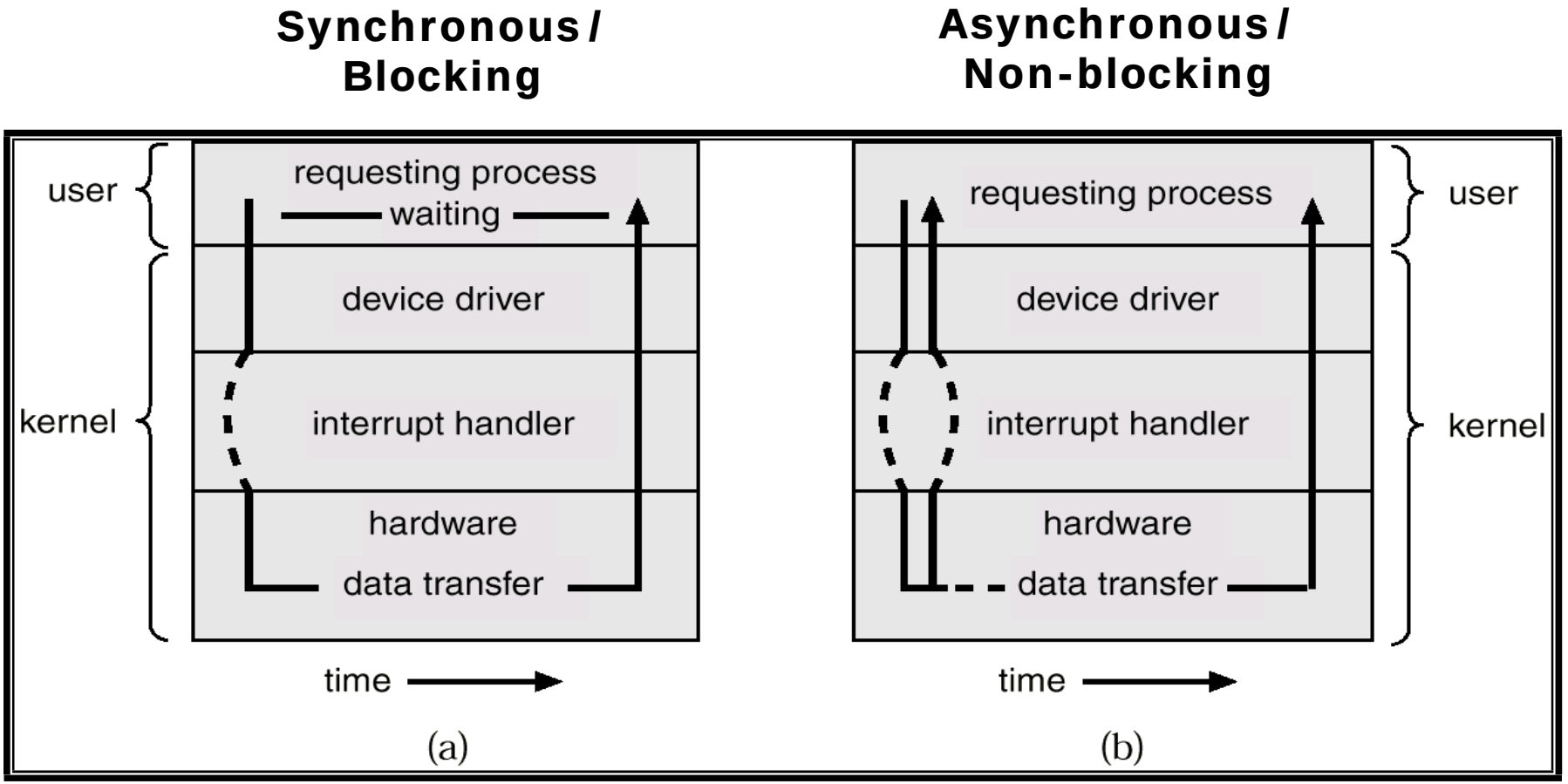
Interrupt routine examines device database and performs action required by interrupt

↓

Handler completes operation, restores saved (interrupt state) and returns to user mode (or calls scheduler to switch to anot her program)

Two I/O Methods

- From the perspective of applications



Interrupts and Exceptions

- Interrupts
  - ✓ Generated by hardware devices
    - Triggered by a signal in INTR or NMI pins (Pentium)
  - ✓ Asynchronous

- Exceptions
  - ✓ Generated by software executing instructions
    - INT instruction in IA32
    - Page fault, protection fault
  - ✓ Synchronous
  - ✓ Trap (expected) or fault (unexpected)

Data Transfer Modes in I/O

- ✓ Programmed I/O (PIO)
  - By special I/O instructions
  - Memory-mapped I/O
- ✓ DMA (Direct Memory Access)
  - Used for high-speed I/O devices able to transmit information at close to memory speeds
  - Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
  - Only an interrupt is generated per block

Hardware Protection

- I/O Protection
- Memory Protection
- CPU Protection

Dual-Mode Operation

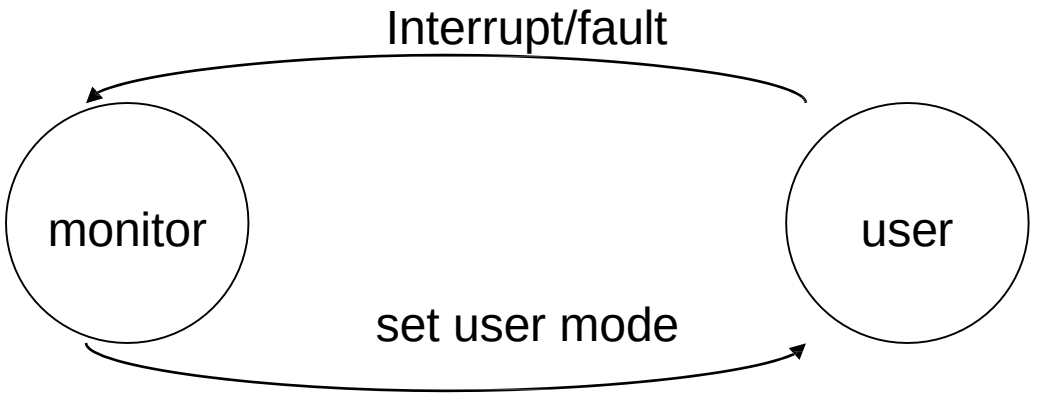
- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly

- Provide hardware support to differentiate between at least two modes of operations
  1. *User mode*
     - execution done on behalf of a user
  2. *Monitor mode* (also *kernel mode* or *system mode*)
     - execution done on behalf of operating system

Dual-Mode Operation (Cont'd)

- *Mode bit* added to computer hardware to indicate the current mode:
  - ✓ monitor (0) or user (1)

- When an interrupt or fault occurs hardware switches to monitor mode



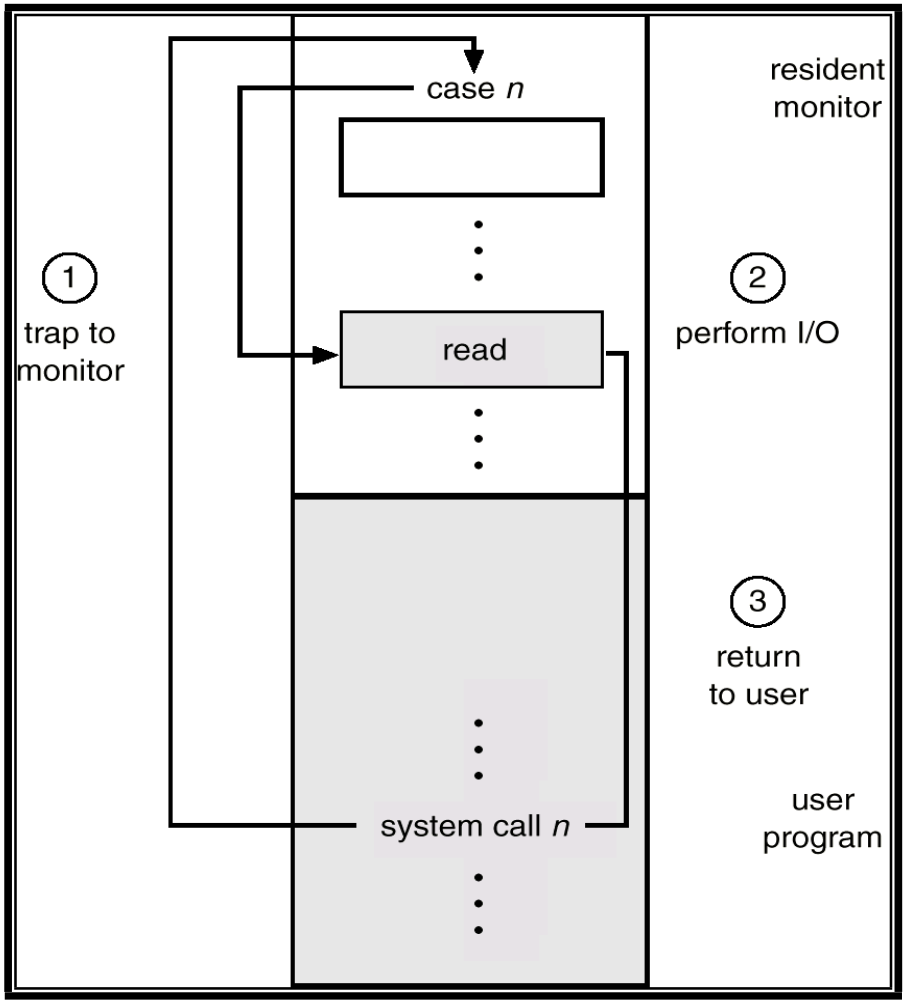
Privileged instructions can be issued only in monitor mode

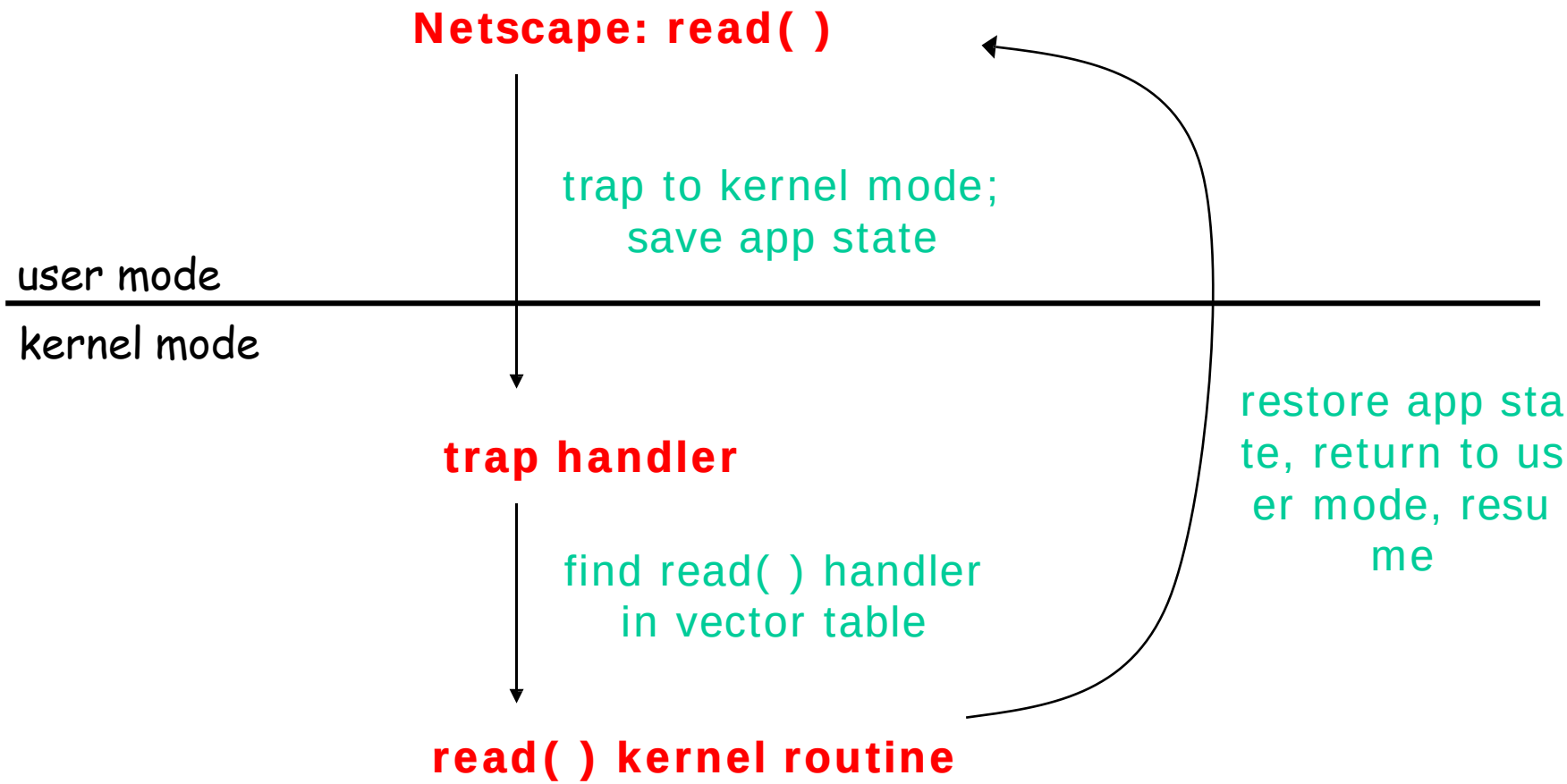
I/O Protection

- All I/O instructions are privileged instructions

- Must ensure that a user program could never gain control of the computer in monitor mode (I.e., a user program that, as part of its execution, stores a new address in the interrupt vector)

Use of A System Call to Perform I/O




Netscape: read( )
trap to kernel mode;
save app state
user mode
kernel mode
trap handler
find read( ) handler
in vector table
read( ) kernel routine
restore app sta
te, return to us
er mode, resu
me

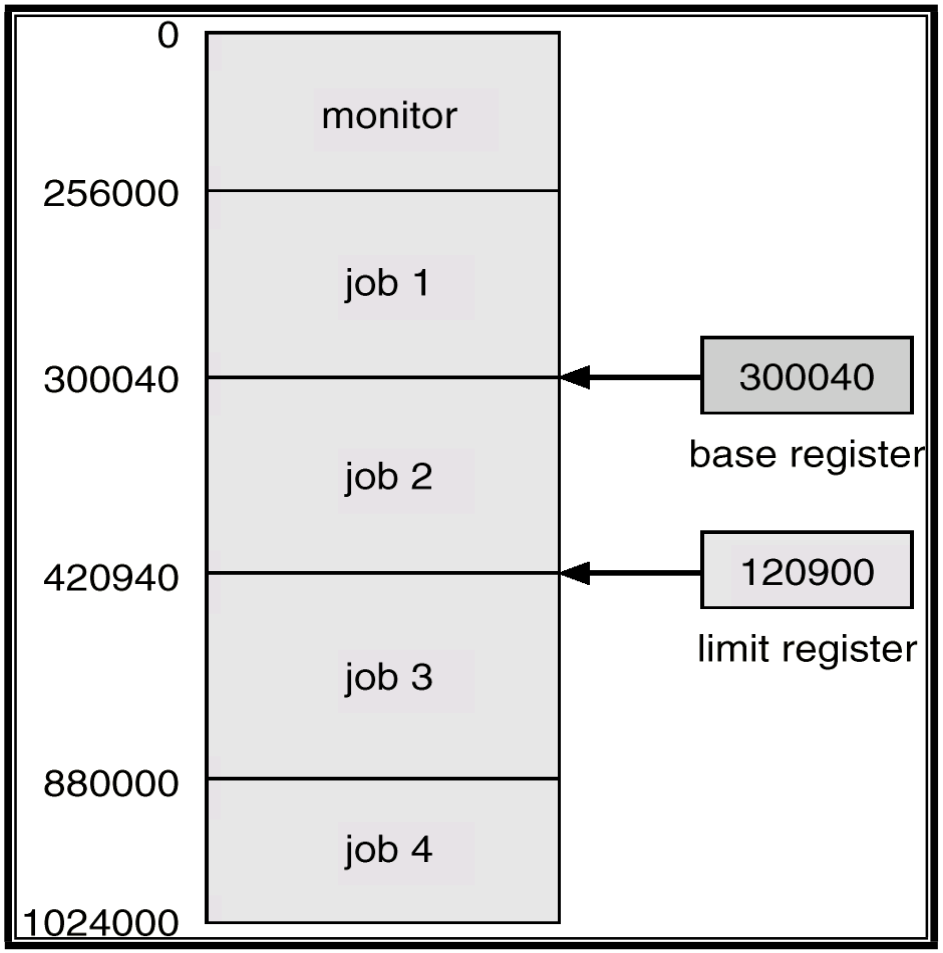
Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines

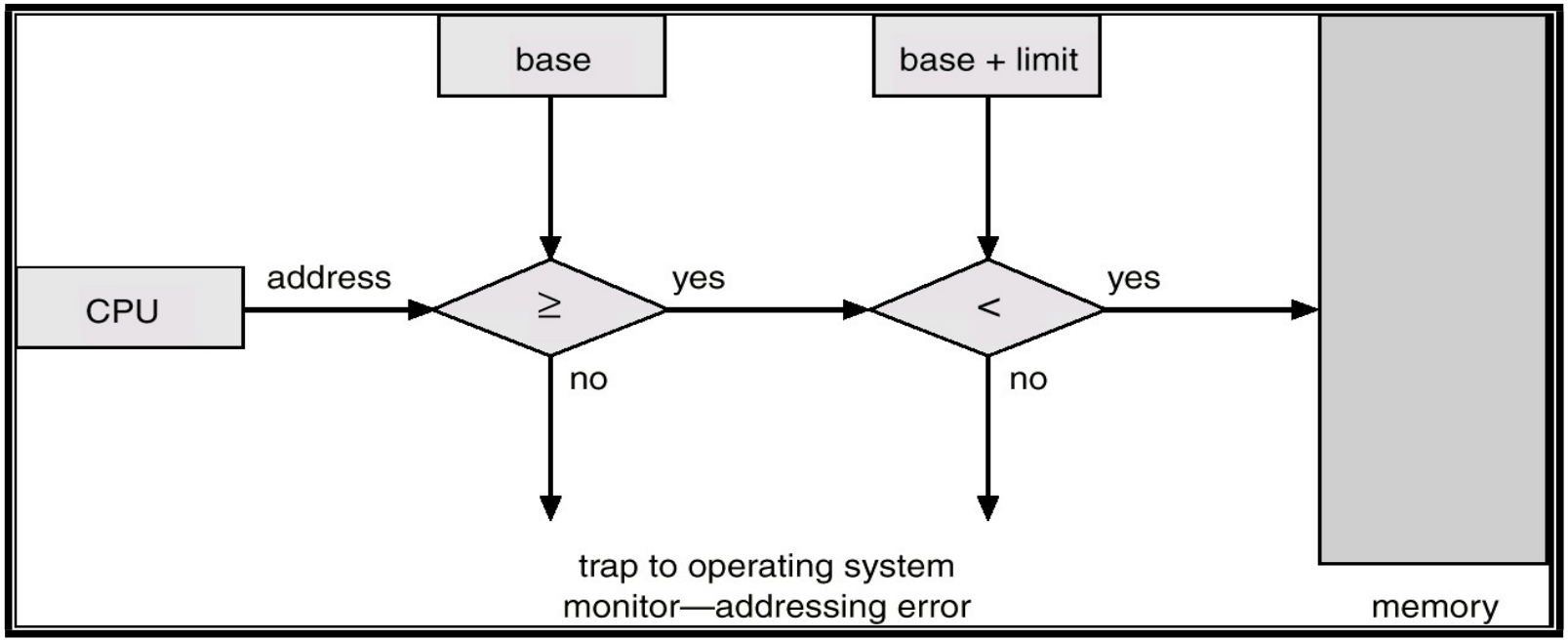
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - ✓ **Base register** – holds the smallest legal physical memory address
  - ✓ **Limit register** – contains the size of the range

- Memory outside the defined range is protected

Use of A Base and Limit Register



Hardware Address Protection



Memory Protection

- MMU (Memory Management Unit)
  - ✓ Memory management hardware provides more sophisticated memory protection mechanisms
    - base and limit registers
    - page table pointers, page protection, TLBs
    - virtual memory
    - segmentation
  - ✓ Manipulation of memory management hardware are protected (privileged) operations

CPU Protection

- *Timer* – interrupts computer after specified period to ensure operating system maintains control
  - ✓ Timer is decremented every clock tick
  - ✓ When timer reaches the value 0, an interrupt occurs
- Timer commonly used to implement time sharing
- Time also used to compute the current time
- Load-timer is a privileged instruction

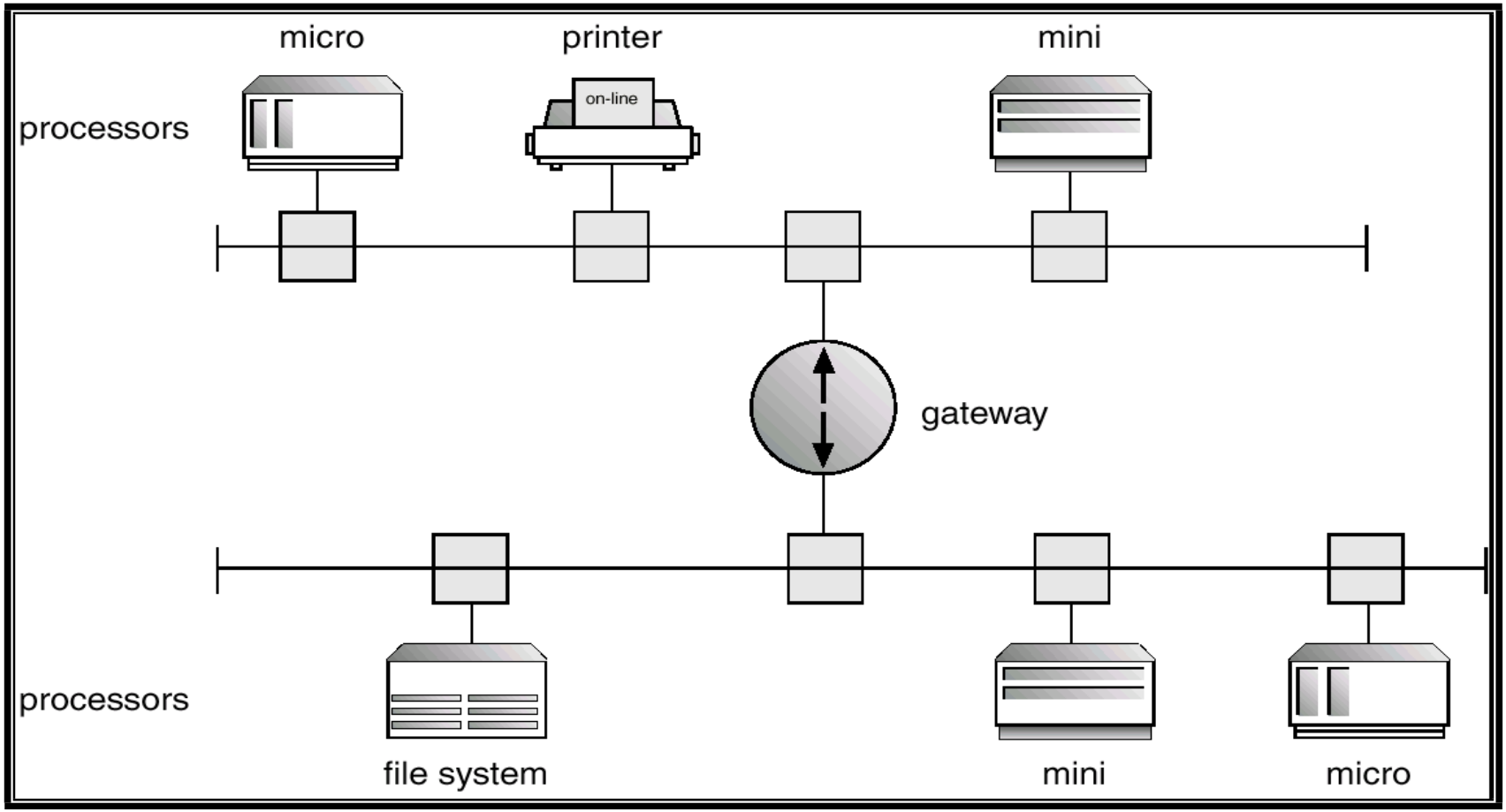
Timers

- How does the OS take control of CPU from the running programs?
  - ✓ Use a hardware timer that generates a periodic interrupt
  - ✓ The timer interrupt transfers control back to OS
  - ✓ The OS preloads the timer with a time to interrupt. (“quantum”)
    - 10ms for Linux
    - Cf) time slice or time quantum
  - ✓ The timer is privileged.
    - Only the OS can load it

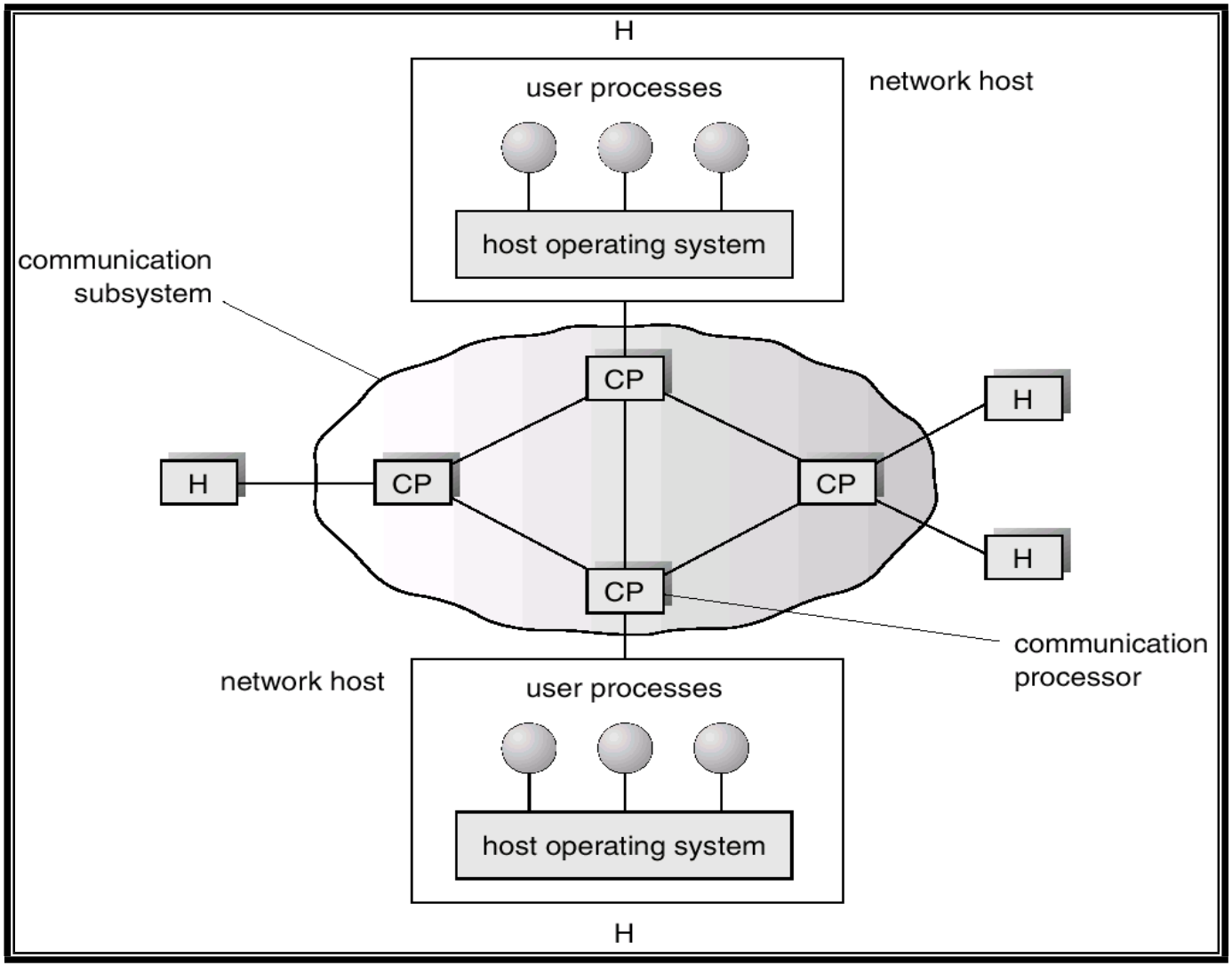
Network Structure

- Local Area Networks (LAN)
  - ✓ Relay, bridge, gateway, router, switch, hub
- Wide Area Networks (WAN)

Local Area Network Structure



Wide Area Network Structure



Computer Systems

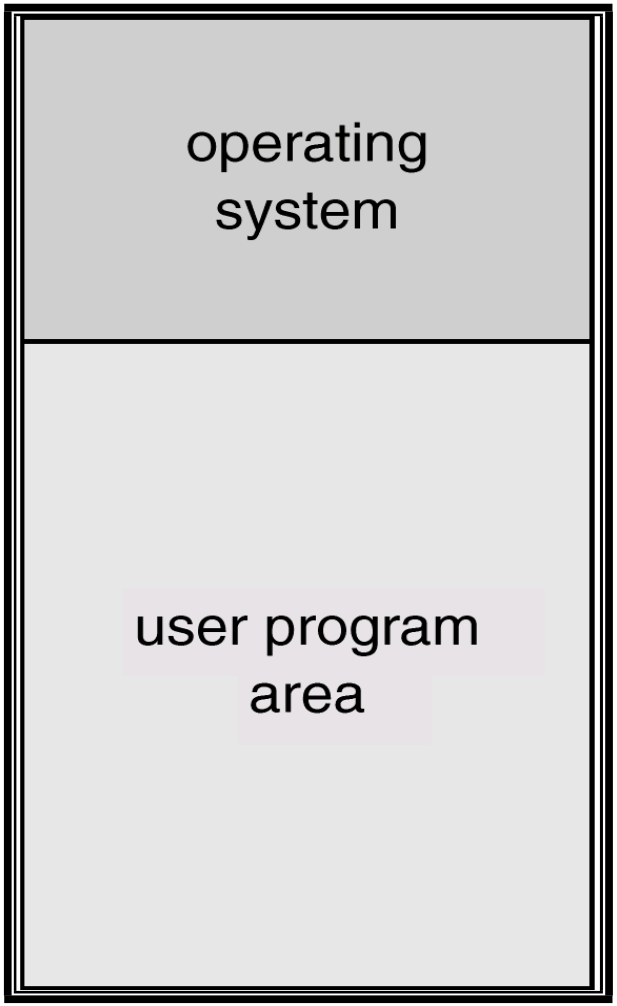
- Mainframe systems
- Multiprogramming systems
- Time-sharing systems
- Desktop systems
- Parallel systems
- Distributed systems
- Clustered systems
- Real-time systems
- Embedded systems
- Handheld systems

Mainframe Systems

- Reduce setup time by batching similar jobs
- Automatic job sequencing – automatically transfers control from one job to another. First rudimentary operating system
- Resident monitor
  - ✓ initial control in monitor
  - ✓ control transfers to job
  - ✓ when job completes, control transfers back to monitor

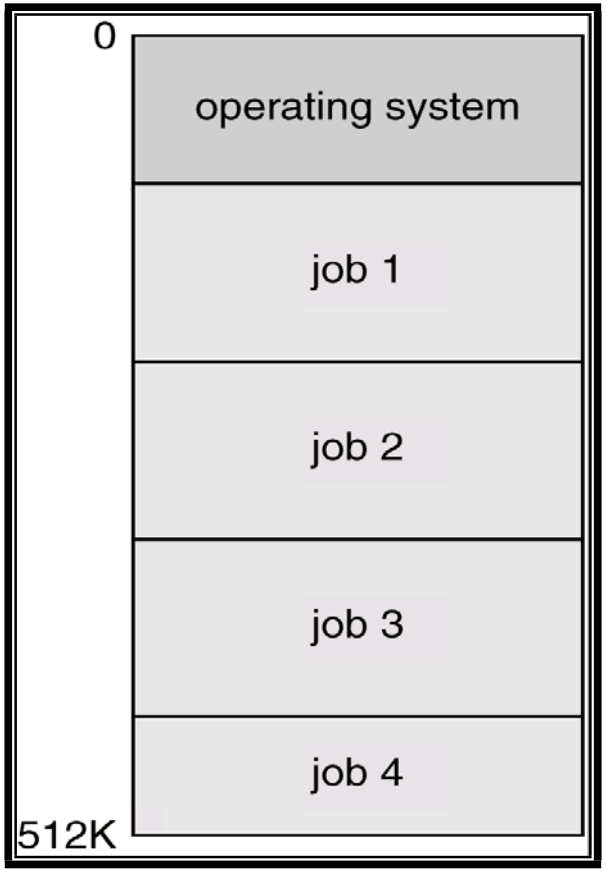
IBM Supercomputer Watson

Memory Layout for a Simple Batch System



Multiprogrammed Batch Systems

Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.



OS Features Needed for Multiprogramming

- I/O routine supplied by the system
- Memory management – the system must allocate the memory to several jobs
- CPU scheduling – the system must choose among several jobs ready to run
- Allocation of devices

Time-Sharing Systems–Interactive Computing

- The CPU is multiplexed among several jobs that are kept in memory and on disk (the CPU is allocated to a job only if the job is in memory)
- A job swapped in and out of memory to the disk
- On-line communication between the user and the system is provided; when the operating system finishes the execution of one command, it seeks the n ext “control statement” from the user’s keyboard
- On-line system must be available for users to access data and code

Terminology

- Batch, Multiprogramming, Time-sharing(or Multitasking)
- Job scheduling vs. CPU scheduling
- Job, Task, Process
- Concurrent, Simultaneous, Parallel

Desktop Systems

- *Personal computers* – computer system dedicated to a single user
- I/O devices – keyboards, mice, display screens, small printers
- User convenience and responsiveness
- Can adopt technology developed for larger operating system
- Individuals have sole use of computer and do not need advanced CPU utilization of protection features
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux)

Parallel Systems

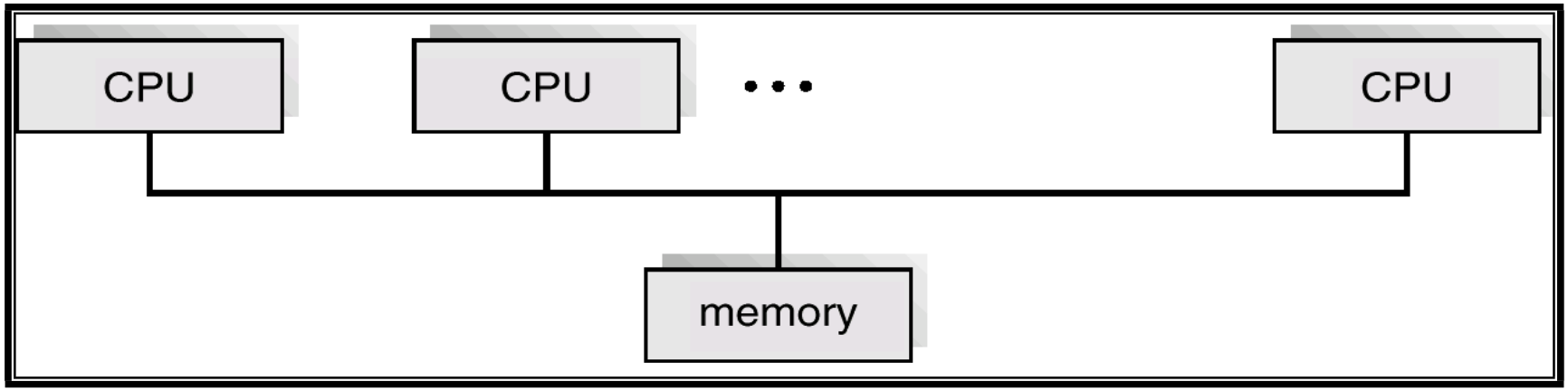
- Multiprocessor systems with more than one CPU in close communication
- *Tightly coupled system* – processors share memory and a clock; communication usually takes place through the shared memory
- Advantages of parallel system:
  - ✓ Increased *throughput*
  - ✓ Economical
  - ✓ Increased reliability
    - graceful degradation
    - fail-soft systems (fault-tolerant systems)
- *Cf) Co-processor or controller*

Parallel Systems (Cont'd)

- *Symmetric multiprocessing (SMP)*
  - ✓ Each processor runs and identical copy of the operating system
  - ✓ Many processes can run at once without performance deterioration
  - ✓ Most modern operating systems support SMP

- *Asymmetric multiprocessing*
  - ✓ Each processor is assigned a specific task; master processor schedules and allocated work to slave processors
  - ✓ More common in extremely large systems

Symmetric Multiprocessing Architecture



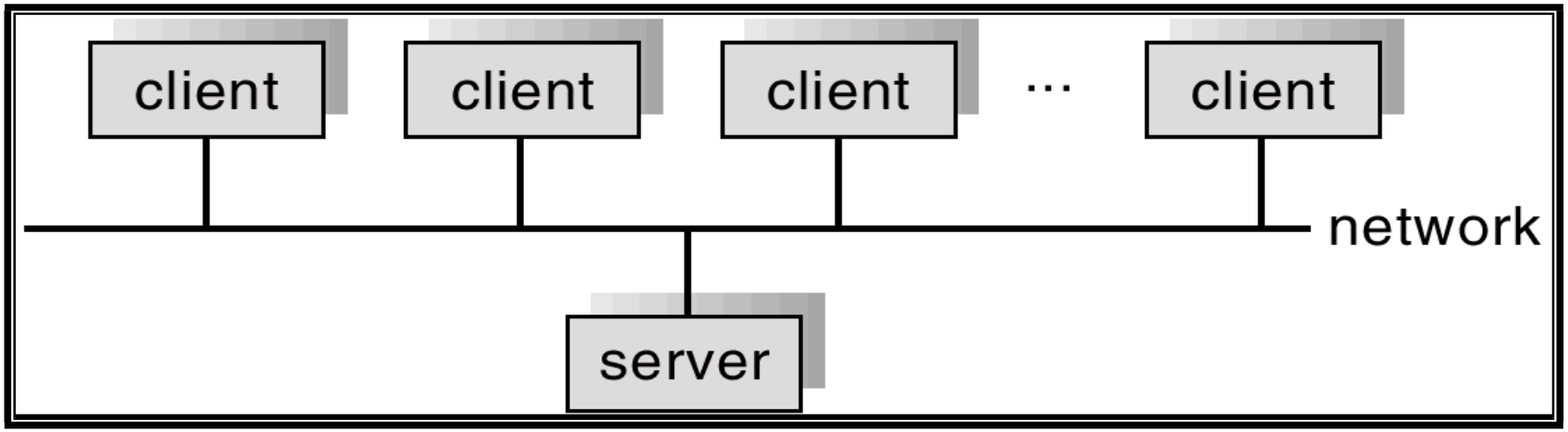
Distributed Systems

- Distribute the computation among several physical processors
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines
- Advantages of distributed systems
  - ✓ Resources Sharing
  - ✓ Computation speed up – load sharing
  - ✓ Reliability
  - ✓ Communications

Distributed Systems (Cont'd)

- Requires networking infrastructure
- Local area networks (LAN) or Wide area networks (WAN)
- May be either client-server or peer-to-peer systems

General Structure of Client-Server



Clustered Systems

- Clustering allows two or more systems to share storage
- Provides high reliability (or high availability)
- *Asymmetric clustering*: one server runs the application while other servers standby
- *Symmetric clustering*: all N hosts are running the application
- Cf) Grid Computing
- Cf) Distributed Lock Manager (DLM), Storage Area Network (SAN)

Large-scale Server in Data Center

Real-Time Embedded Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems
- Well-defined fixed-time constraints
- Real-Time systems may be either *hard* or *soft* real-time

Real-Time Embedded Systems (Cont'd)

- Hard real-time:
  - ✓ Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
  - ✓ Conflicts with time-sharing systems, not supported by general-purpose operating systems

- Soft real-time
  - ✓ Limited utility in industrial control of robotics
  - ✓ Useful in applications (multimedia, virtual reality) requiring advanced operating-system features

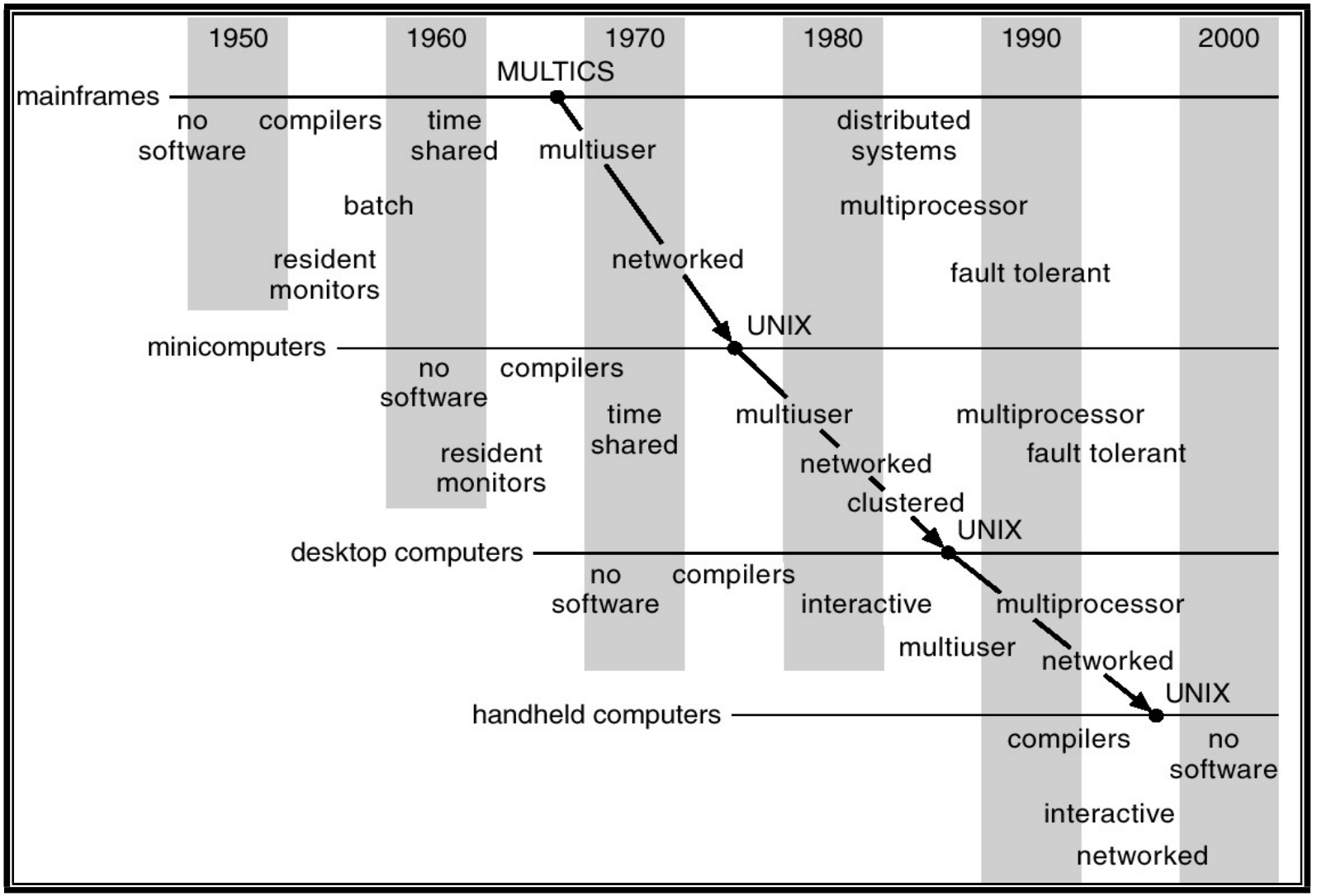
Handheld Systems

- Personal Digital Assistants (PDAs)
- Cellular telephones
- Issues:
  - ✓ Limited memory
  - ✓ Slow processors
  - ✓ Small display screens

Computing Environments

- Traditional computing
- Client-Server computing
- Peer-to-Peer computing
- Web-based computing

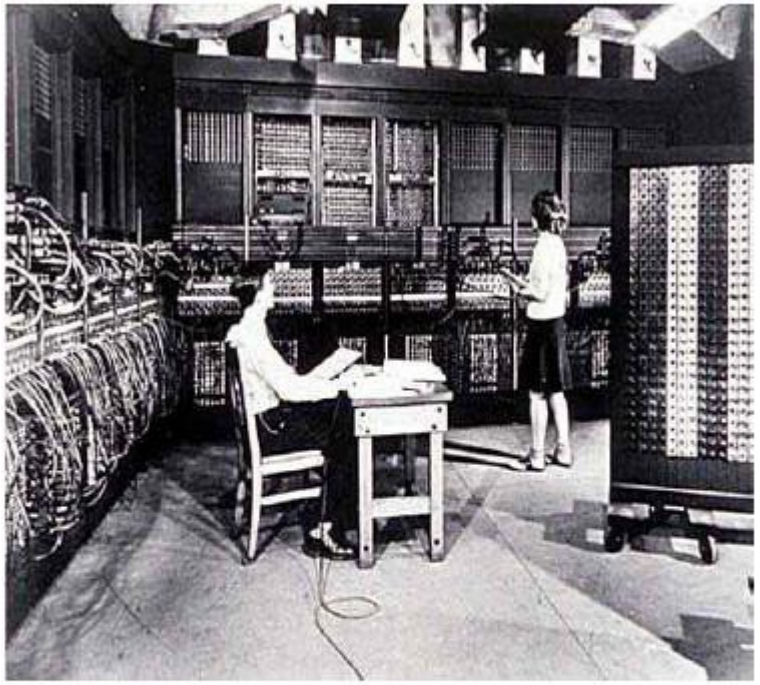
Migration of Operating-System Concepts and Features



1st Generation (1945-55)

- *Vacuum Tubes and Plugboards*

- No OS
- No Programming Languages
- No Assembly Languages



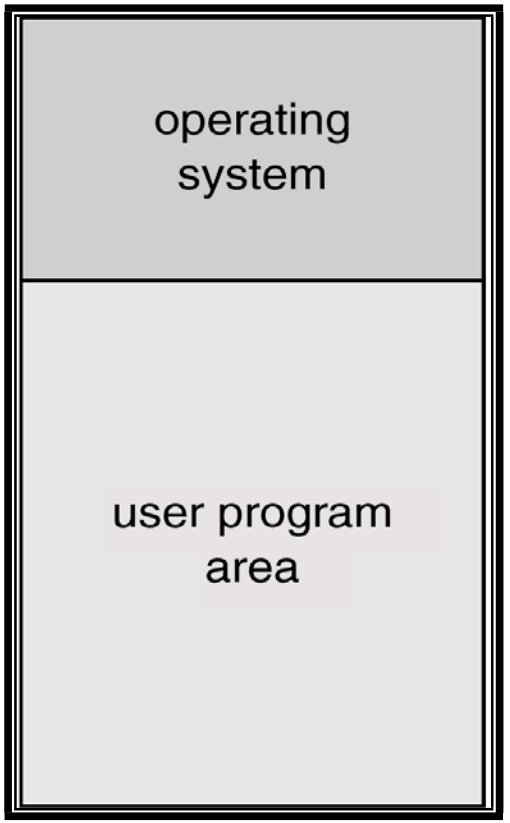
Eniac, 1940's

John Von Newman

2nd Generation (1955-65)

- *Transistors and Mainframes*

- Batch systems
  - ✓ One job at a time
  - ✓ Card readers, tape drives, line printers
  - ✓ OS is always resident in memory and merely transfers a control.
  - ✓ CPU is underutilized due to the bottleneck in I/O



3rd Generation (1965-80)

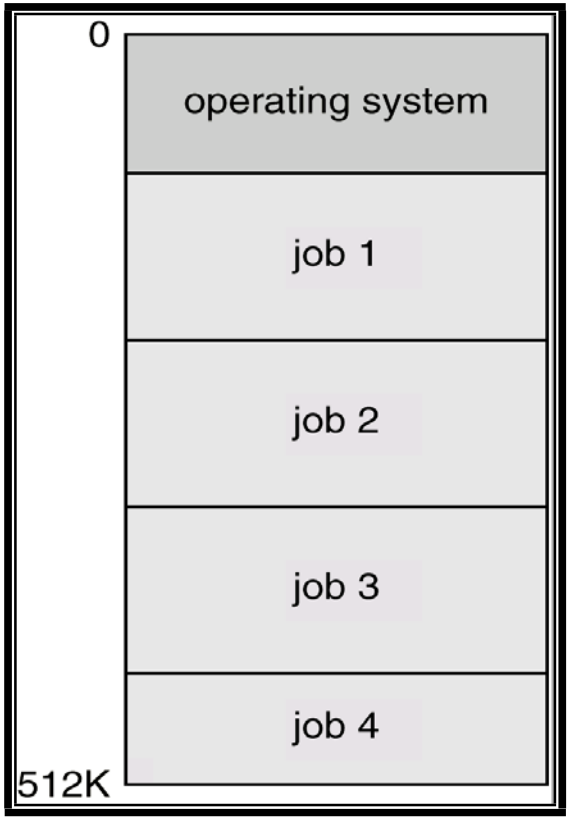
- *Integrated Circuits (ICs)*

- Architectural Advances
  - ✓ Using ICs: better price/performance
  - ✓ Disk drives
  - ✓ On-line terminals

  - ✓ The notion of “Computer Architecture”
    - IBM System/360 family

3rd Generation (1965-80)

- Multiprogrammed Systems
  - ✓ Increase CPU utilization
  - ✓ OS features
    - Job scheduling
    - Memory management
    - CPU scheduling
    - Protection
  - ✓ Spooling (Simultaneous Peri pheral Operation On-Line)



3rd Generation (1965-80)

- Time-sharing Systems
  - ✓ Improve response time
  - ✓ OS features
    - Swapping
    - Virtual memory
    - File system
    - Sophisticated CPU scheduling
    - Synchronization
    - Interprocess communication
    - Interactive shell
    - More protection, …

4th Generation (1980-)

- *LSIs & VLSIs*

- Architectural Advances
  - ✓ Microprocessors: smaller and faster
  - ✓ Storages: larger and faster
  - ✓ Personal computers
  - ✓ CPU work is offloaded to I/O devices

- Modern OS Features
  - ✓ GUI (Graphical User Interface)
  - ✓ Multimedia
  - ✓ Internet & Web
  - ✓ Networked / Distributed, etc.

A long time ago,
in a galaxy far, far away, …

- IBM OS/360: Multiprogramming
- MIT CTSS (Compatible Time-Sharing System)
- MIT, Bell Labs, GE, MULTICS
  (MULTiplexed Information and Computing Service)

And UNIX was born in 1969

OS History: UNIX (1969-85)

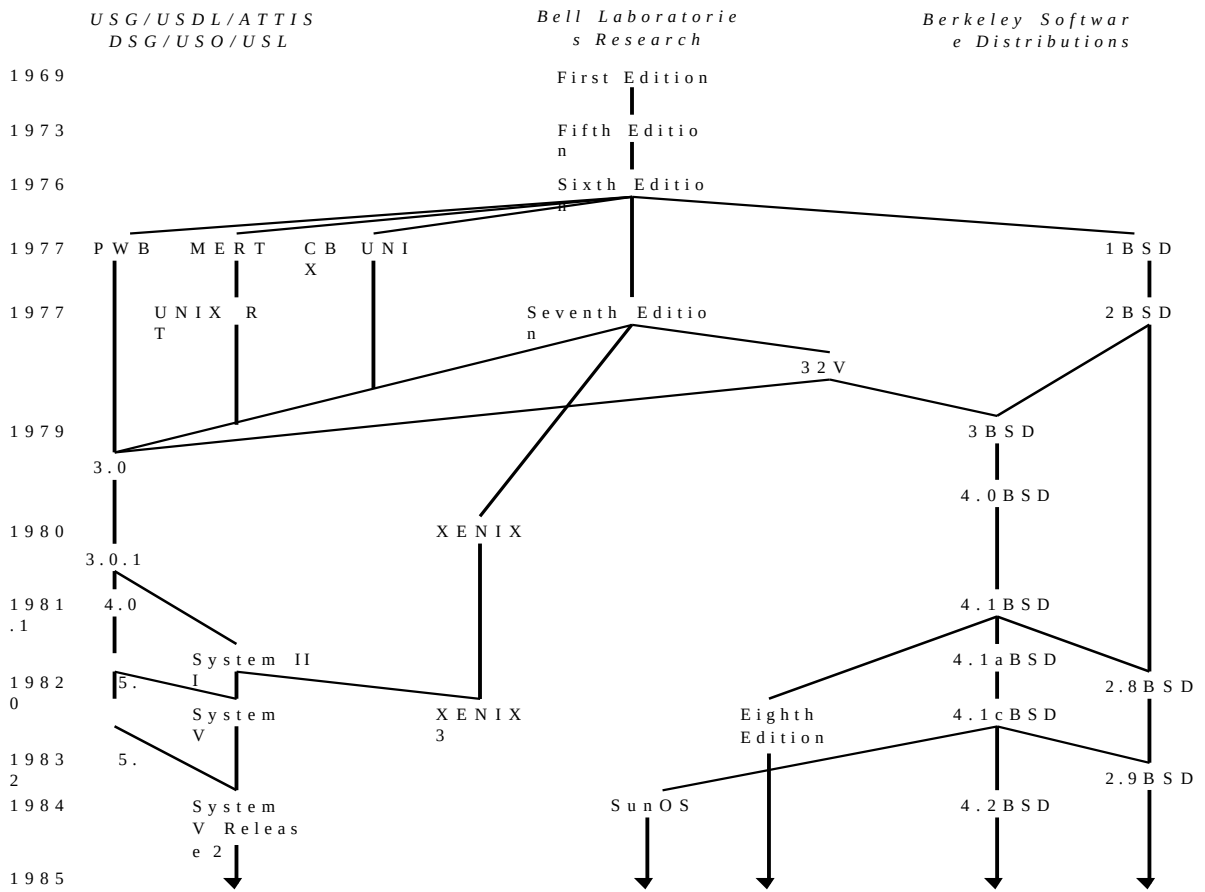


Figure 1.1 The UNIX system family tree, 1969-1985

OS History: UNIX (1985-96)

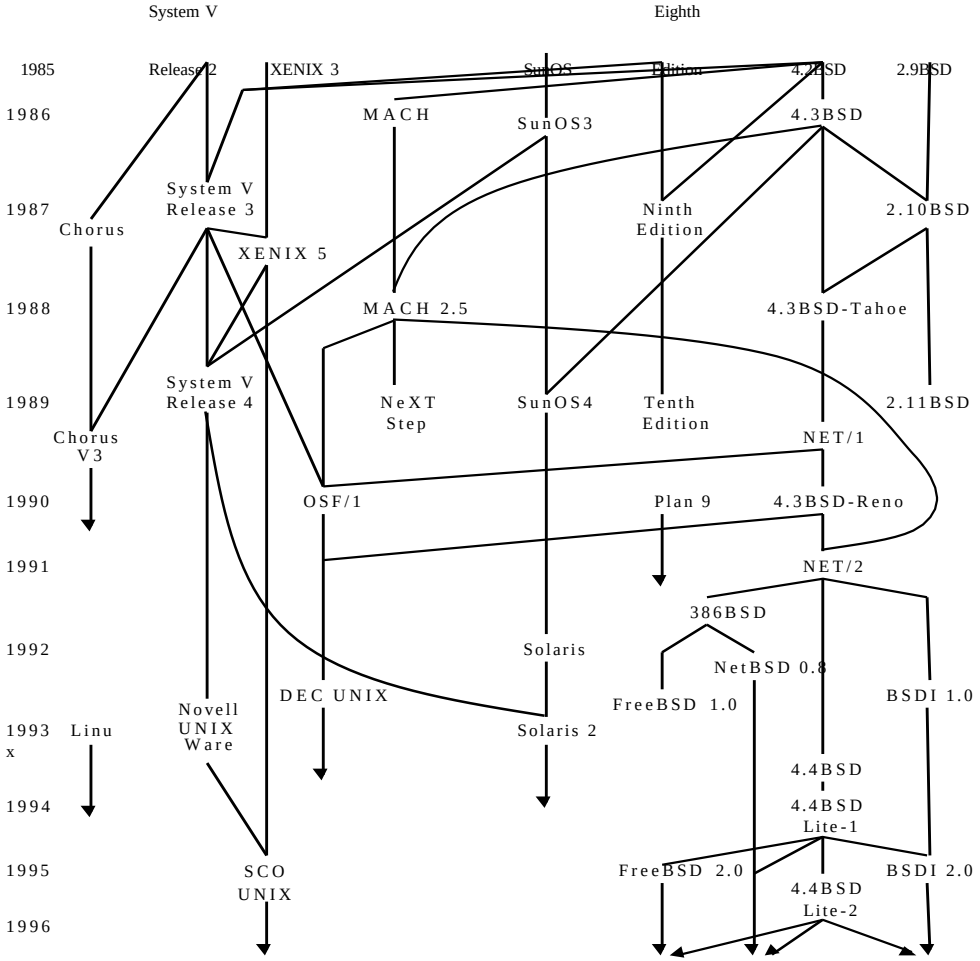


Figure 1.2 The UNIX system family tree, 1986-1996

OS History: UNIX (Current)

- Sun Solaris
- HP HP-UX
- IBM AIX
- Caldera (SCO) Unixware
- Compaq (Digital) Tru64
- SGI Irix
- Linux, FreeBSD, NetBSD
- Apple Mac OS X, etc.

- Cf) POSIX

OS History: Windows & Linux

1981 82 84 86 88 90 91 92 93 94 95 96 97 98 99 00 01

MS-DOS Family

1.0 2.0 3.0 3.2 4.0 5.0 6.0 7.0

Windows Family

1.0 2.0 3.0 3.1 95 98 98 SE ME

Windows for WorkGroup Family

3.1 3.11

Windows NT Family

3.1 3.51 4.0 2K XP

Linux

0.01 0.99 1.0 1.2 2.0 2.1 2.2 2.4

OS History: Windows & Linux (Cont'd)

04 07 09 10 12 ?

Windows Vista

Windows 7

Windows 8

2.6.0

A.B.C
A: Kernel version
B: Major revision
C: Minor revision

2.6.33

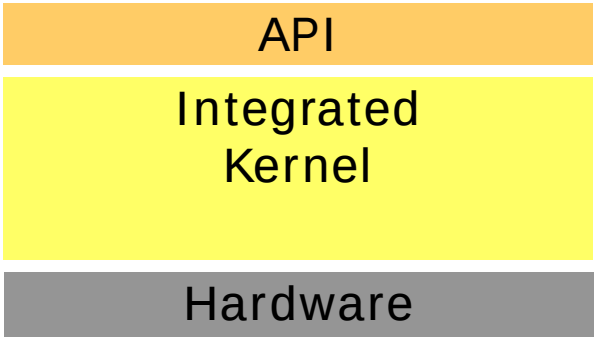
3.2.0

OS Classification (1)

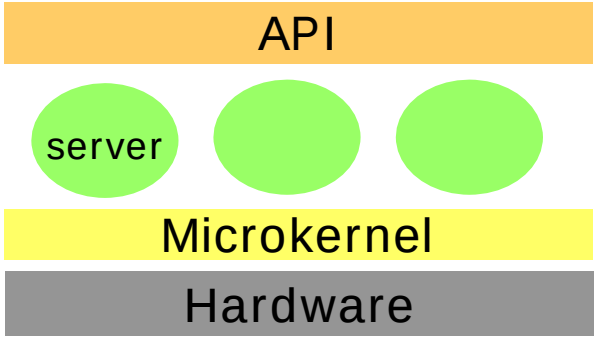
	MS-DOS	Windows 98	Windows NT	Linux
Multi-user	X	X	O	O
Multi-task Multi-process	X	O	O	O
Multi-processor	X	X	O	O
Multi-thread	X	O	O	O

OS Classification (2)

- Monolithic Kernel
  - ✓ Function calls
  - ✓ Unixware, Solaris, AIX, HP-UX, Linux, etc.



- Micro kernel
  - ✓ Multiple servers
  - ✓ Message passing
  - ✓ Mach, Chorus, Linux mk, etc.



OS Classification (3)

- Mainframe systems
  - ✓ CTS, MULTICS, IBM MVS, VM
- Desktop systems
  - ✓ DOS, Windows, MacOS, Unix/Linux
- Multiprocessor systems
- Cluster systems
- Distributed systems
  - ✓ Amoeba(Vrije Univ.), Locus(UCLA), Grapevine(Xerox), V(Stanford), Eden(U. of Washington), Chorus/Nucleus(Inria)
- Embedded systems
  - ✓ Vertex, pSOS, VxWorks, OSE, Windows-CE, Embedded Linux
  - ✓ Company-proprietary OS (Cisco, Qualcomm, Palm, Cellvic)
- Real-time systems
  - ✓ Real-Time Linux, Spring(U. of Massachusetts), HARTS(U. of Michigan), MARUTI(U. of Maryland)

Summary

- Operating system
  - ✓ An intermediary software between users and computer hardware

- Goals of operating system
  - ✓ Use the computer hardware in efficient manners
    - Convenient programming & use through encapsulation of complexity of H/Ws
    - Maximization of performance of computer hardware resources

Summary (Cont'd)

- Computer system architecture & Operation
  - ✓ CPU, Memory, I/O devices
  - ✓ Instructions
    - Arithmetic instructions: add, subtract, …
    - Logical instructions: and, or, not, …
    - Control flow instructions: goto, if, call, return, …
    - Data instructions: load, store, move, input, output, …
  - ✓ Interrupt
    - When I/O events occur, I/O devices trigger an interrupt
    - CPU saves its operation and runs the corresponding interrupt service routine (ISR)
  - ✓ DMA
    - Transfers blocks of data from I/O devices to main memory without CPU intervention
  - ✓ Storage hierarchy
    - Register / Cache / Main memory / Secondary storage / Tertiary storage

Summary (Cont'd)

- Computer system architecture & Operation (Cont'd)
  - ✓ System call
    - An interface to the service made available by operating system
    - When a user process calls a system call, controls are transferred from the user process to operating system → user mode vs. kernel mode
  - ✓ Protection
    - I/O protection: Dual-mode operation
    - Memory protection: MMU (Memory Management Unit)
    - CPU protection: Timer interrupt
- History of computer systems
  - ✓ Mainframe systems or batch systems
  - ✓ Multiprogramming systems
  - ✓ Time-sharing systems
  - ✓ Desktop systems
  - ✓ Parallel / Distributed / Clustered / Cloud systems
  - ✓ Real-time / Embedded / Handheld systems