ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Real-Time Operating Systems M

1. Introduction

Notice

The course material includes slides downloaded from:

http://codex.cs.yale.edu/avi/os-book/

(slides by Silberschatz, Galvin, and Gagne, associated with Operating System Concepts, 9th Edition, Wiley, 2013)

and

http://retis.sssup.it/~giorgio/rts-MECS.html

(slides by Buttazzo, associated with Hard Real-Time Computing Systems, 3rd Edition, Springer, 2011)

which has been edited to suit the needs of this course.

The slides are authorized for personal use only.

Any other use, redistribution, and any for profit sale of the slides (in any form) requires the consent of the copyright owners.

Welcome!

- Contact: by appointment

  Paolo.Torroni@unibo.it

- Weekly time table

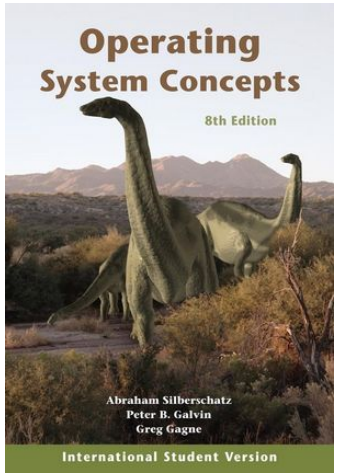
Ore	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
8-9						
9-10					42 L	
10-11					42 L	
11-12					42 E	
12-13						
13-14						
14-15			01 L			
15-16	34 L		01 E			
16-17	34 E					
17-18						
18-19						
19-20						

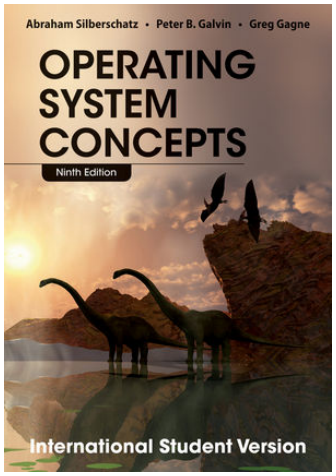
Syllabus

- Operating Systems
  - Fundamentals of general-purpose operating systems
  - Process management
  - Process coordination
  - Memory management
  - I/O-system management
- Real-Time
  - Fundamentals of real-time embedded systems
  - Aperiodic task scheduling
  - Periodic task scheduling
  - Resource access

Text books

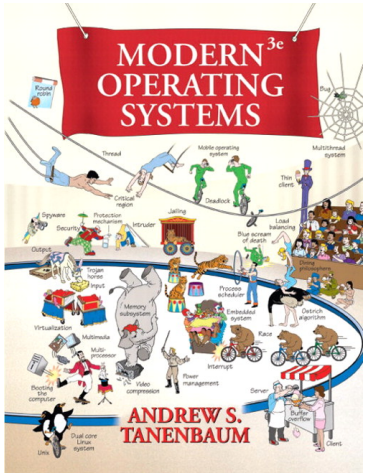
- Text Books
  - **Operating Systems**:
    - Abraham Silberschatz, Peter B. Galvin, Greg Gagne. Operating System Concepts, 8th Edition. International Student Version. Wiley 2010 (9th Edition, March 2013)
    - Chapters 1 — 9 & 13
  - **Real-Time**:
    - Giorgio C. Buttazzo. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, 3rd Edition. Springer 2011
    - Chapters 1 — 4 & 7

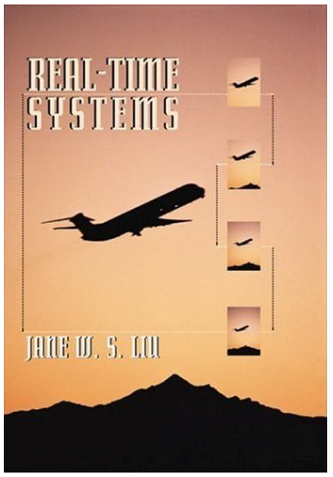




Alternative text books

- If problems in obtaining recommended text books:
  - **Operating Systems** (in place of Silberschatz):
    - Andrew S. Tanenbaum. Modern Operating Systems, 3rd Edition. Prentice Hall 2008
  - **Real-Time** (in place of Buttazzo):
    - Jane W. S. Liu. Real-Time Systems. Prentice Hall 2000





Grading

- Exam: format
  - Written & oral exam
- Grading options
  - 2 Midterms + 1 Final
    - April 5, 2013
    - May 13, 2013
    - June ??, 2013
  - Standard (all-in-one)
    - July 2013
    - September 2013
    - …

Resources

- Software
  - https://www.virtualbox.org/ (or http://www.vmware.com)
  - http://www.ubuntu.com/ & Code::Blocks
  - https://www.rtai.org/

- Prerequisites
  - Language of lectures must **not** represent a hurdle
  - Programming language – C
  - Computer architectures

Prerequisites

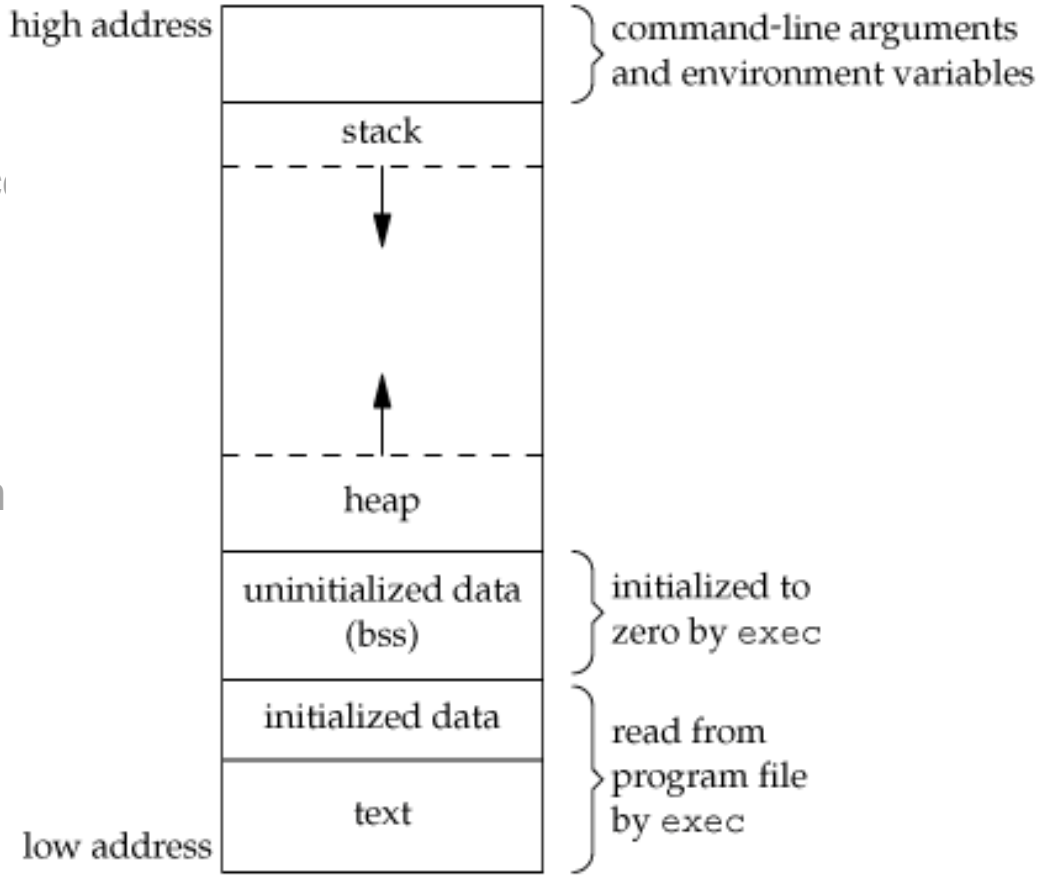
- Basics of computer architectures
  - Von Neumann architecture
    - CPU architecture
    - Fetch-Decode-Execute cycle

Prerequisites

- Basic C programming
  - Interpreted vs compiled code
  - Memory sections
  - Stack frame
  - Passing arguments
    - by value *vs* by reference
  - `Typdef struct`
  - Pointers

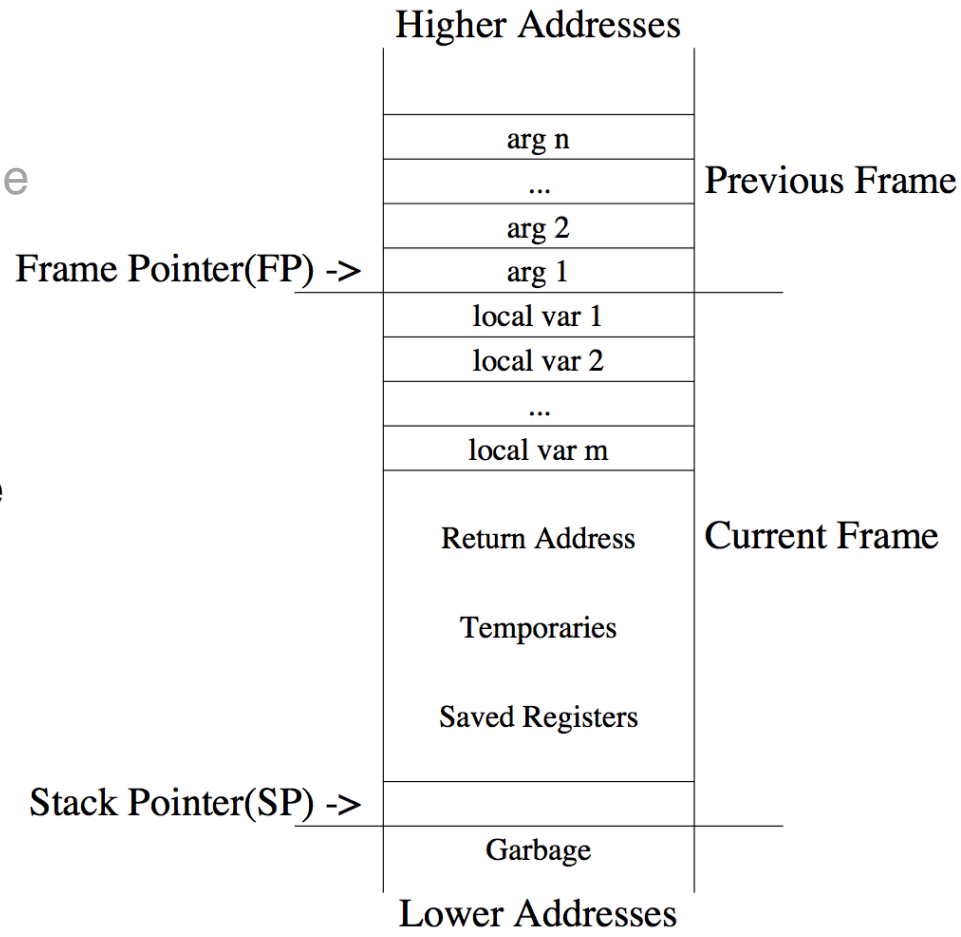
Prerequisites

- Basic C programming
  - Interpreted vs compiled c
  - Memory sections
  - Stack frame
  - Passing arguments
    - by value *vs* by referen
  - `Typdef struct`
  - Pointers



Prerequisites

- Basic C programming
  - Interpreted vs compiled code
  - Memory sections
  - Stack frame
  - Passing arguments
    - by value *vs* by reference
  - `Typdef struct`
  - Pointers



Prerequisites

- Basic C programming
  - Interpreted vs compiled code
  - Memory sections
  - Stack frame
  - Passing arguments
    - by value *vs* by reference
  - `Typdef struct`
  - Pointers

This week – synopsis

- Logistics
- **Silberschatz**: Part 1 (Overview) – Chapter 1 (Introduction)
  1. What is an operating system
     - User's view & system's view
  2. Computer system organization
     - Event-driven operation (interrupts & traps)
     - Storage device hierarchy
     - I/O Structure (polling, interrupts, DMA)
  3. Computer-system architecture
     - Single- & multi-processor systems

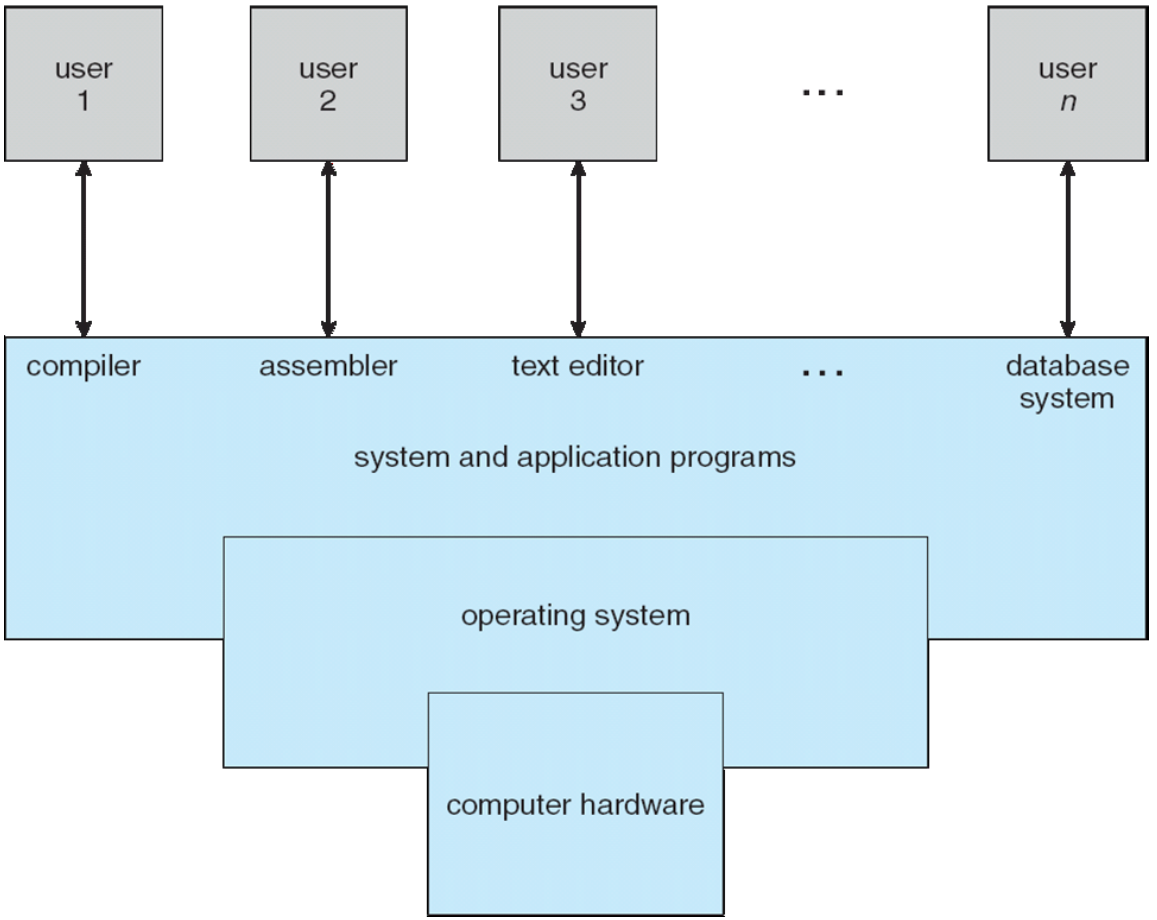
This week – synopsis

4. Operating-system structure
   - Multiprogramming & multitasking
5. Operating-system operations
   - Dual-mode operation
6. Process management
7. Memory management
8. Storage management
   - Caching
   - I/O Subsystem
9. Protection and security
10. Computing environments
    - Mobile, Distributed, Network OS, Virtualization, Cloud

What is an Operating System?

- A program that acts as an **intermediary** between a **user** of a computer and the computer **hardware**
- User's view / system's view
- 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

Computer System Structure



Operating System Definition

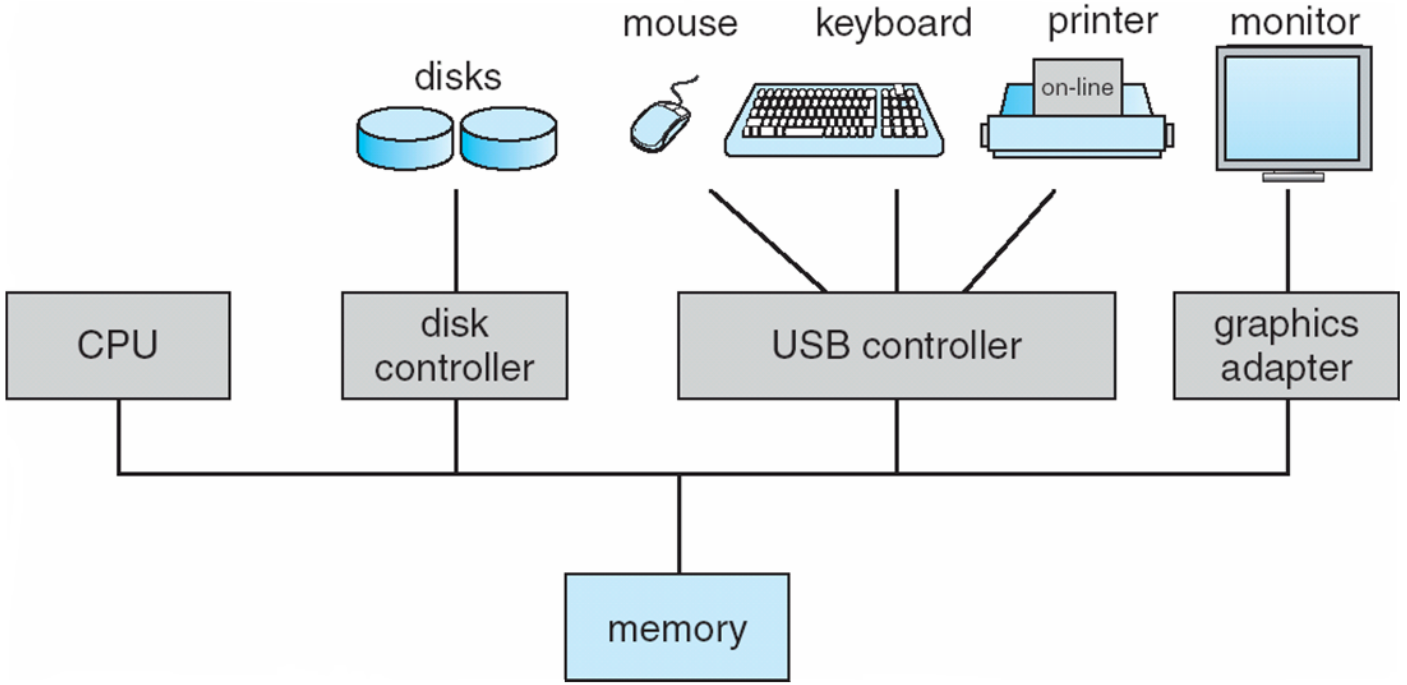
- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer
  - Focus on I/O devices

Operating System Definition

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "A piece of software bringing together all common functions of controlling and allocating resources on behalf of application programs"
- "The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



Computer Startup

- **Bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution
  - Kernel starts first process (init) and wait for events to occur

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
- 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*

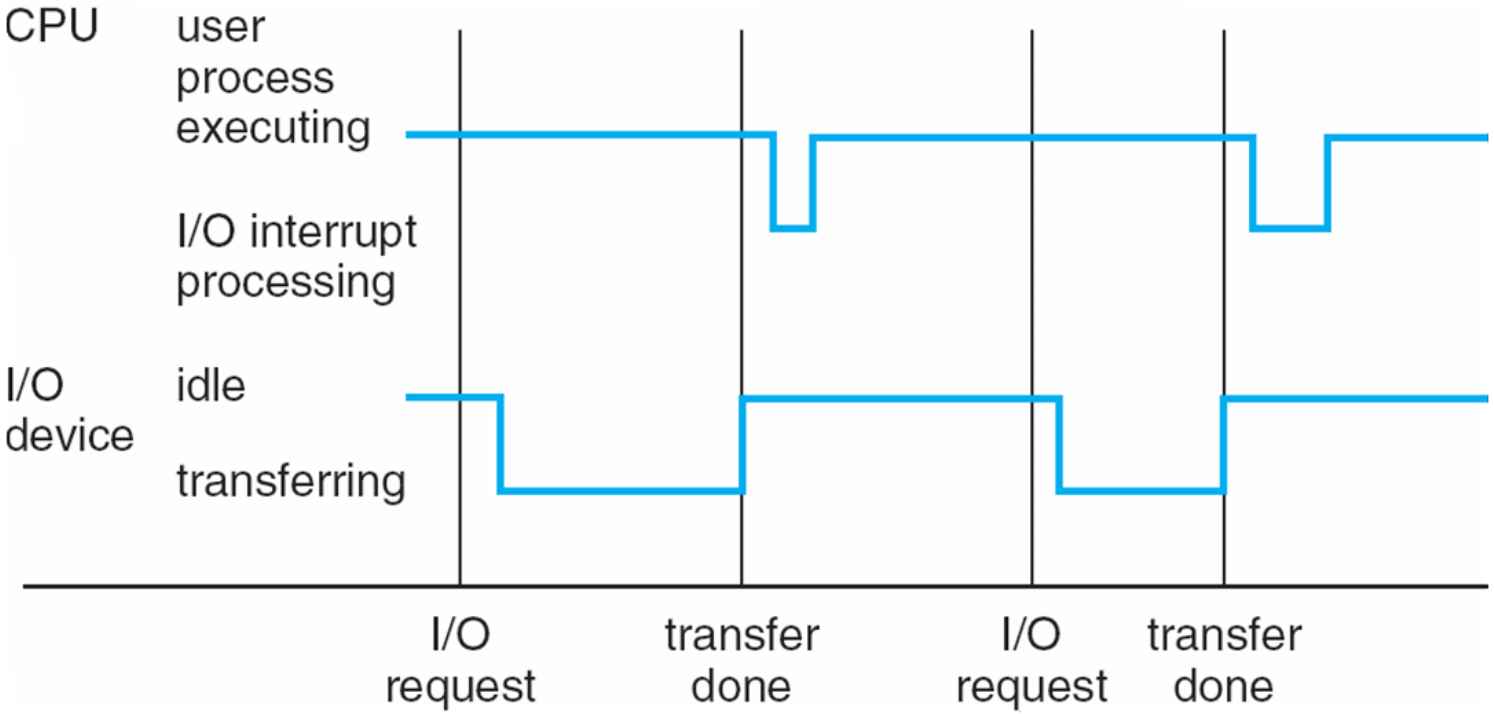
Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*
- A *trap* is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter
- Determines which type of interrupt has occurred:
  - **polling**
  - **vectored** interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt Timeline



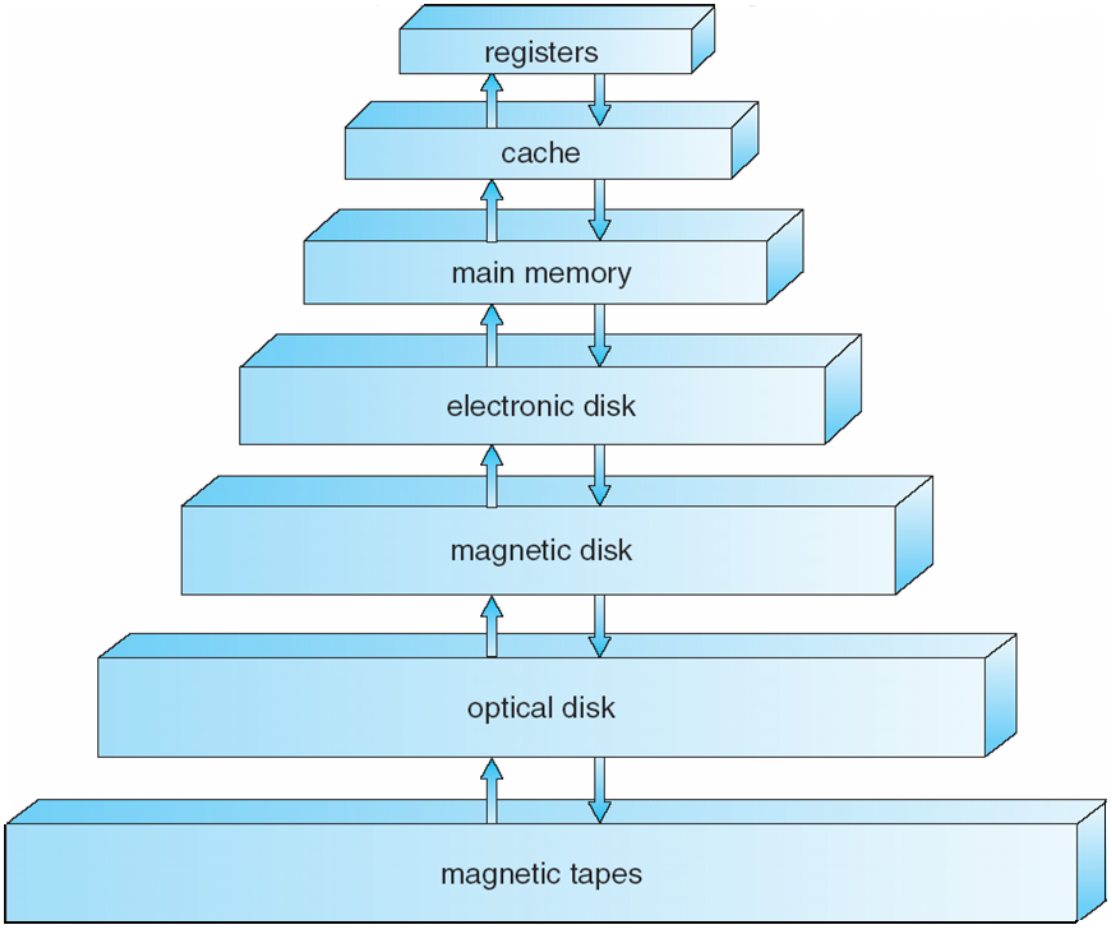
Storage Structure

- Main memory – only large storage media that the CPU can access directly
- 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

Storage Hierarchy

- Bit, Byte, Word, K, M, G, T
- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage

Storage-Device Hierarchy



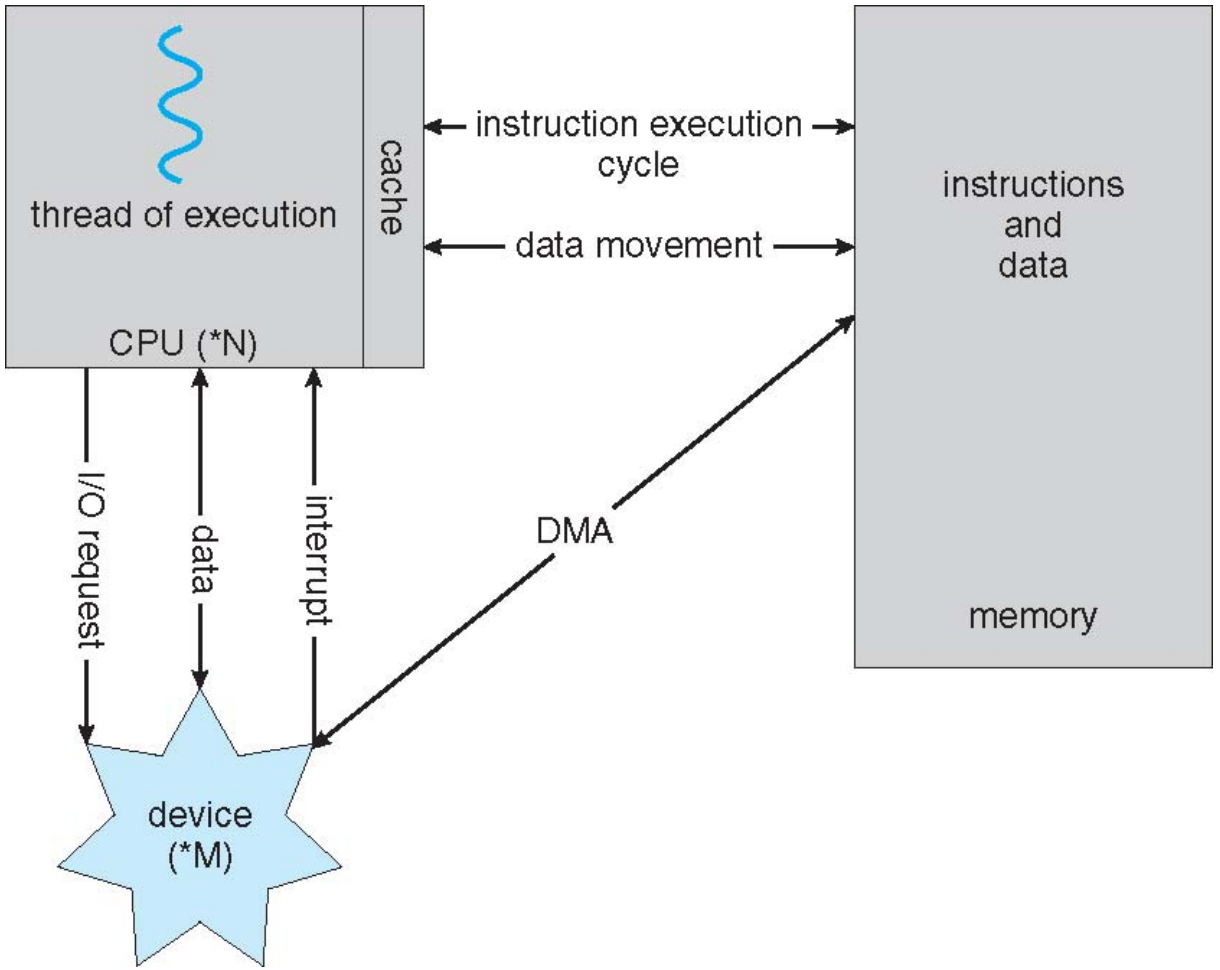
I/O Structure

- 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

Direct Memory Access Structure

- 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 one interrupt is generated per block, rather than the one interrupt per byte

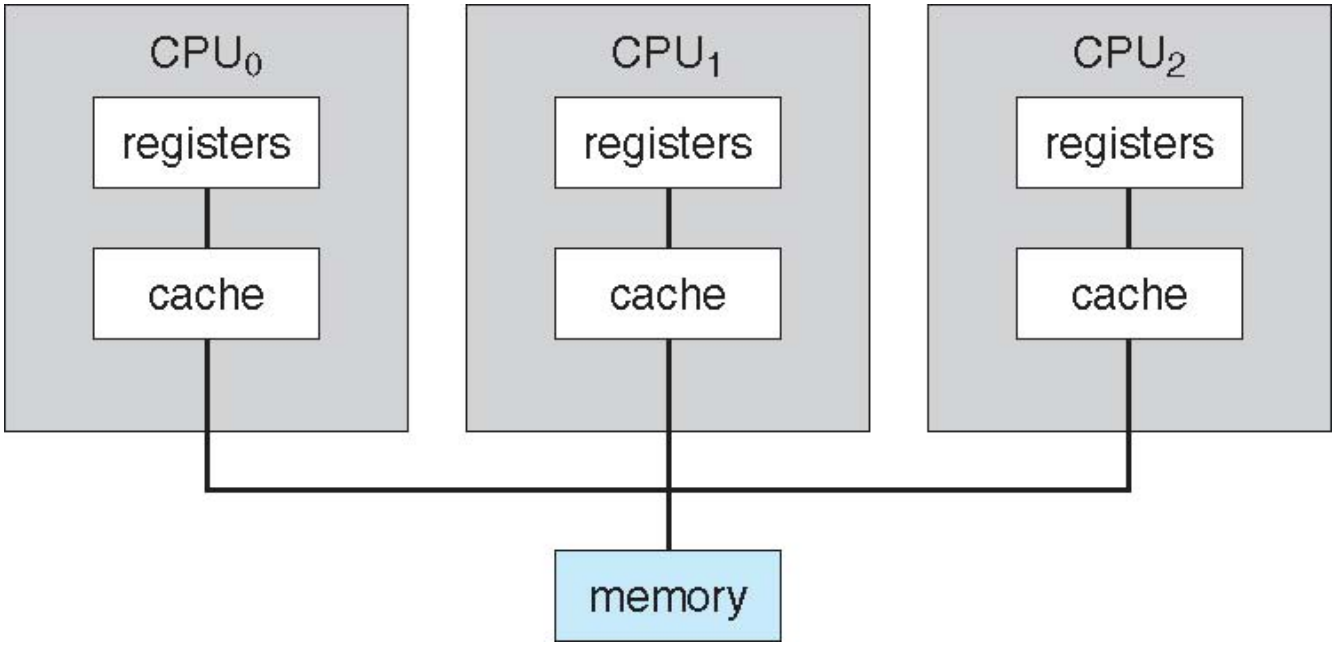
How a Modern Computer Works



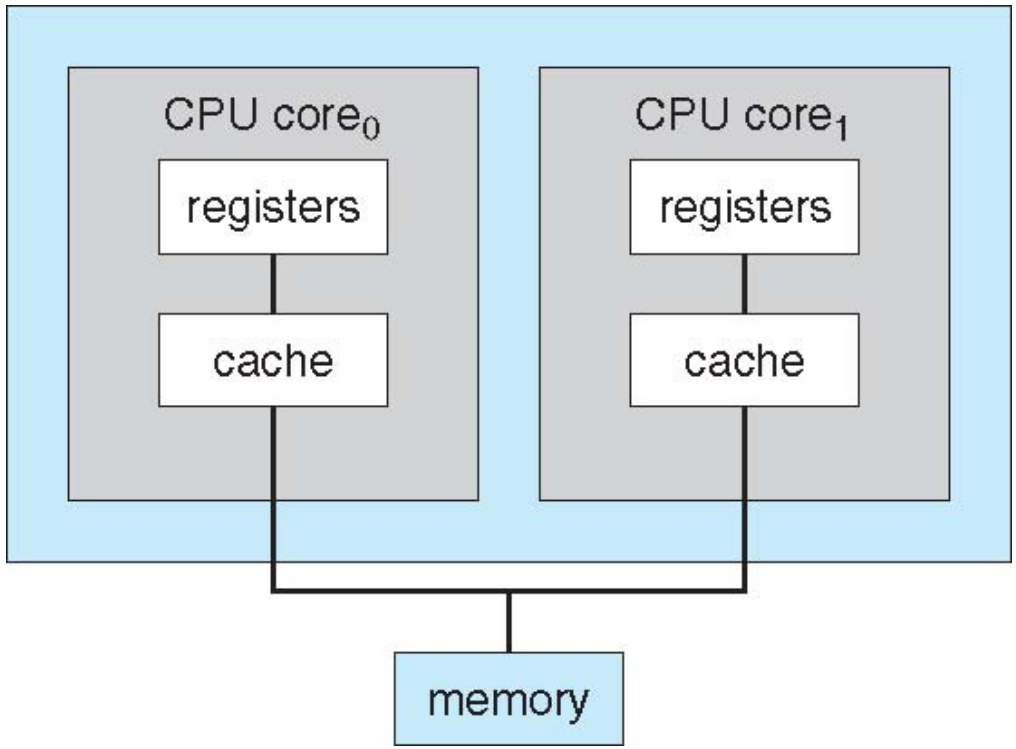
Computer-System Architecture

- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- Multiprocessors systems growing in use and importance
  - Also known as parallel systems, tightly-coupled systems
  - Advantages include
    1. Increased throughput
    2. Economy of scale
    3. Increased reliability – graceful degradation or fault tolerance
  - Two types
    1. Asymmetric Multiprocessing (master-slave)
    2. Symmetric Multiprocessing

Symmetric Multiprocessing Architecture



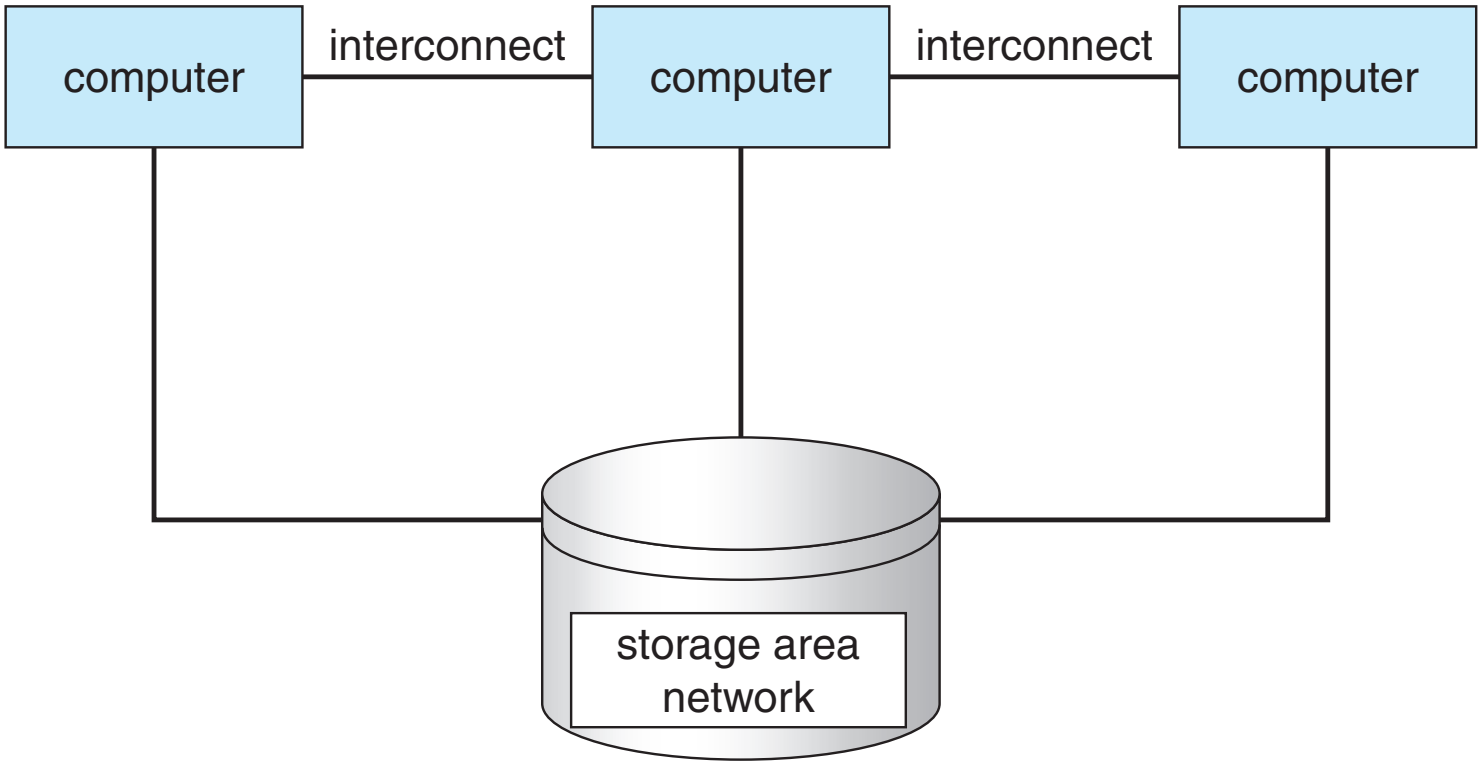
A Dual-Core Design



Clustered Systems

- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a storage-area network (SAN)
  - Provides a high-availability service which survives failures
    - Asymmetric clustering has one machine in hot-standby mode
    - Symmetric clustering has multiple nodes running applications, monitoring each other
  - Some clusters are for high-performance computing (HPC)
    - Applications must be written to use parallelization

Clustered Systems



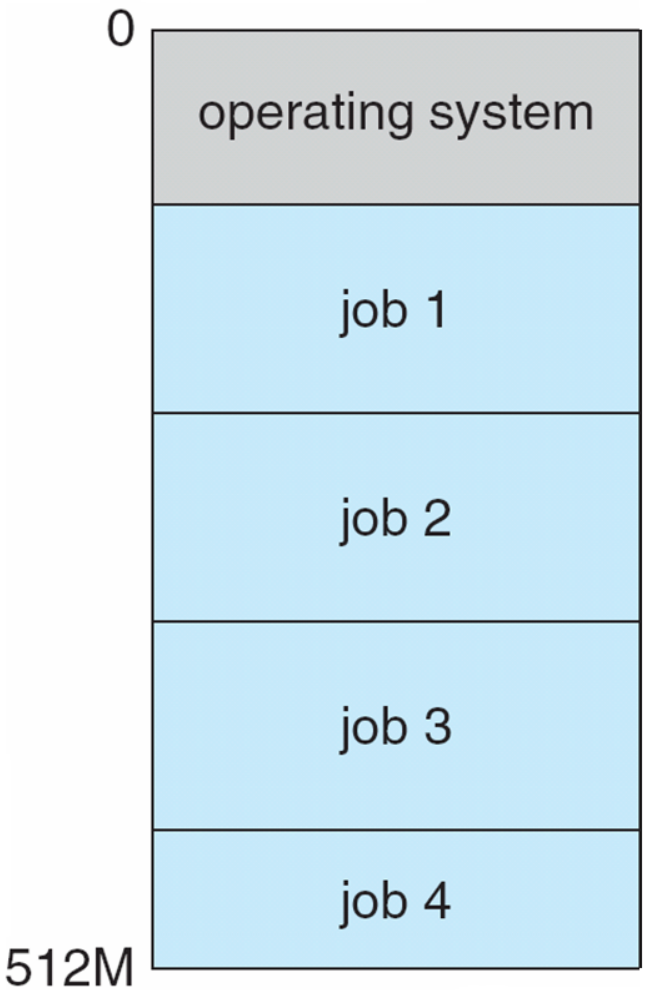
Quizzes

- An operating system is everything a vendor ships when you order an operating system
- The main goal of an operating system is to manage the hardware resources efficiently
- A trap can be generated intentionally by a user program
- Direct Memory Access (DMA) is effective especially with very slow devices
- Increased throughput, economy of scale, and increased reliability are three advantages of multiprocessor systems
- Symmetric multiprocessing is only possible with computer systems that have an even number of CPUs

Operating System Structure

- **Multiprogramming** needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job
- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨**process**
  - If several jobs ready to run at the same time ⇨ **CPU scheduling**
  - If processes don't fit in memory, **swapping** moves them in and out to run
  - **Virtual memory** allows execution of processes not completely in memory

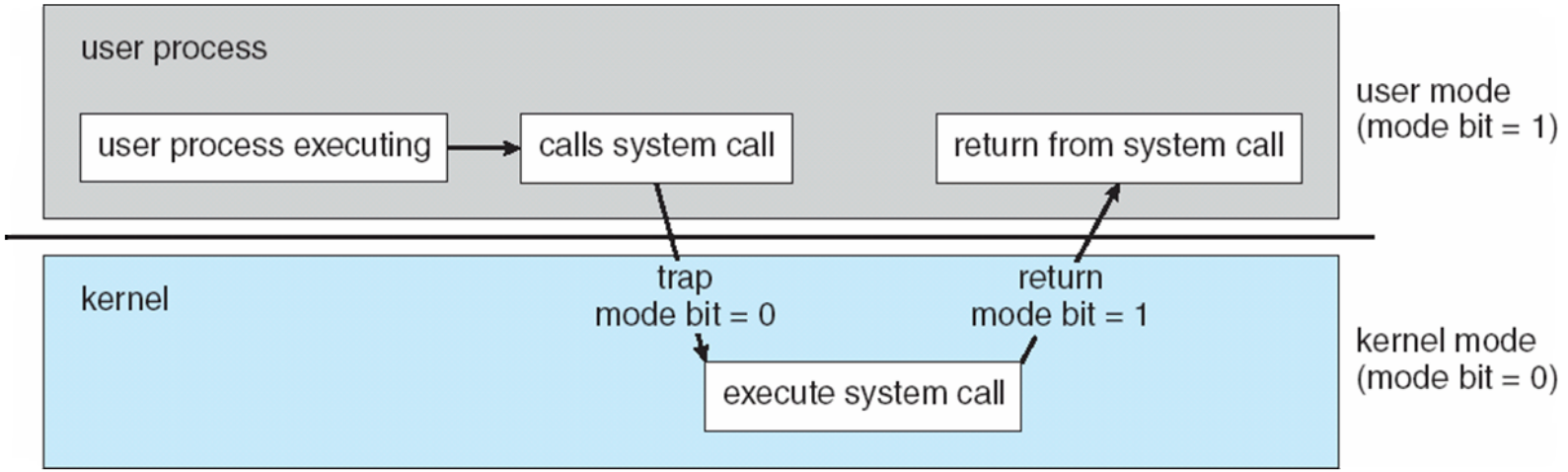
Memory Layout for Multiprogrammed System



Operating-System Operations

- Interrupt driven by hardware
- Software error or request creates **exception** or **trap**
  - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other's data, or code, or the operating system
- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

Transition from User to Kernel Mode



- **Mode bit**
  - Provides ability to distinguish when system is running user code or kernel code
  - Some instructions designated as **privileged**, only executable in kernel mode
  - System call changes mode to kernel, return from call resets it to user

Timer

- To prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

Quizzes

- Which of the following instructions should be **privileged**?
  - Set value of timer
  - Read the clock
  - Clear memory
  - Issue a trap instruction
  - Turn off interrupts
  - Modify entries in device-status table
  - Access I/O device

Process Management

- A process is **a program in execution**. It is **a unit of work within the system**. Program is a *passive entity*, process is an *active entity*.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPU(s) among the processes or threads

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

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

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users

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

Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
- OS responsible for:
  - Creating and deleting files and directories
  - Primitives to manipulate files and directories
  - Mapping files onto secondary storage
  - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) 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
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

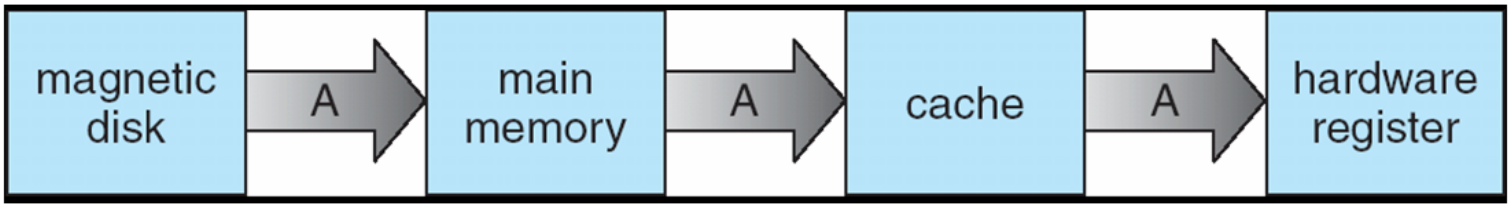
Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit
  - Controlled by operating system, or hardware function transparent to OS

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

Migration of Integer A from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including:
    - buffering (storing data temporarily while it is being transferred),
    - caching (storing parts of data in faster storage for performance),
    - spooling (overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among 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 files, 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, file
  - **Privilege escalation** allows user to change to effective ID with more rights

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has "copyleft" **GNU Public License (GPL)**
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

Computing Environments - 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
- **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

Computing Environments - Mobile

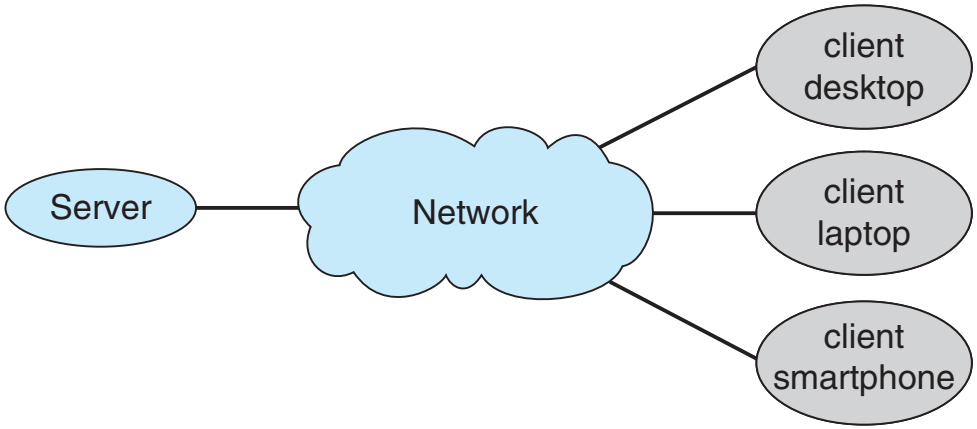
- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

Computing Environments – Distributed

- Distributed
  - Collection of separate, possibly heterogeneous, systems networked together
    - **Network** is a communications path, **TCP/IP** most common
      - **Local Area Network** (**LAN**)
      - **Wide Area Network** (**WAN**)
      - **Metropolitan Area Network** (**MAN**)
      - **Personal Area Network** (**PAN**)
  - **Network Operating System** provides features between systems across network
    - Communication scheme allows systems to exchange messages
    - Illusion of a single system

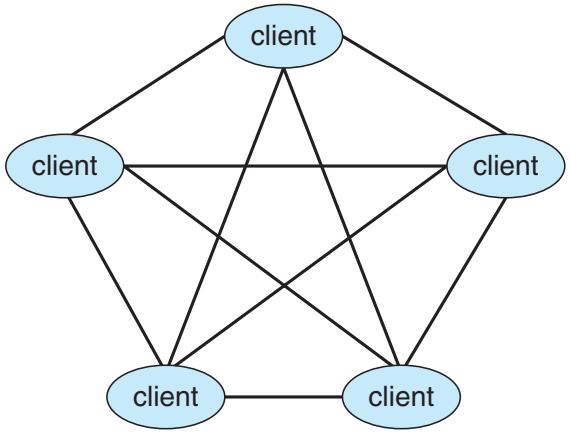
Computing Environments – Client-Server

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems 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



Computing Environments - Peer-to-Peer

- Another model of distributed system



- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via ***discovery protocol***
  - Examples include Napster and Gnutella, **Voice over IP** *(***VoIP***)* such as Skype

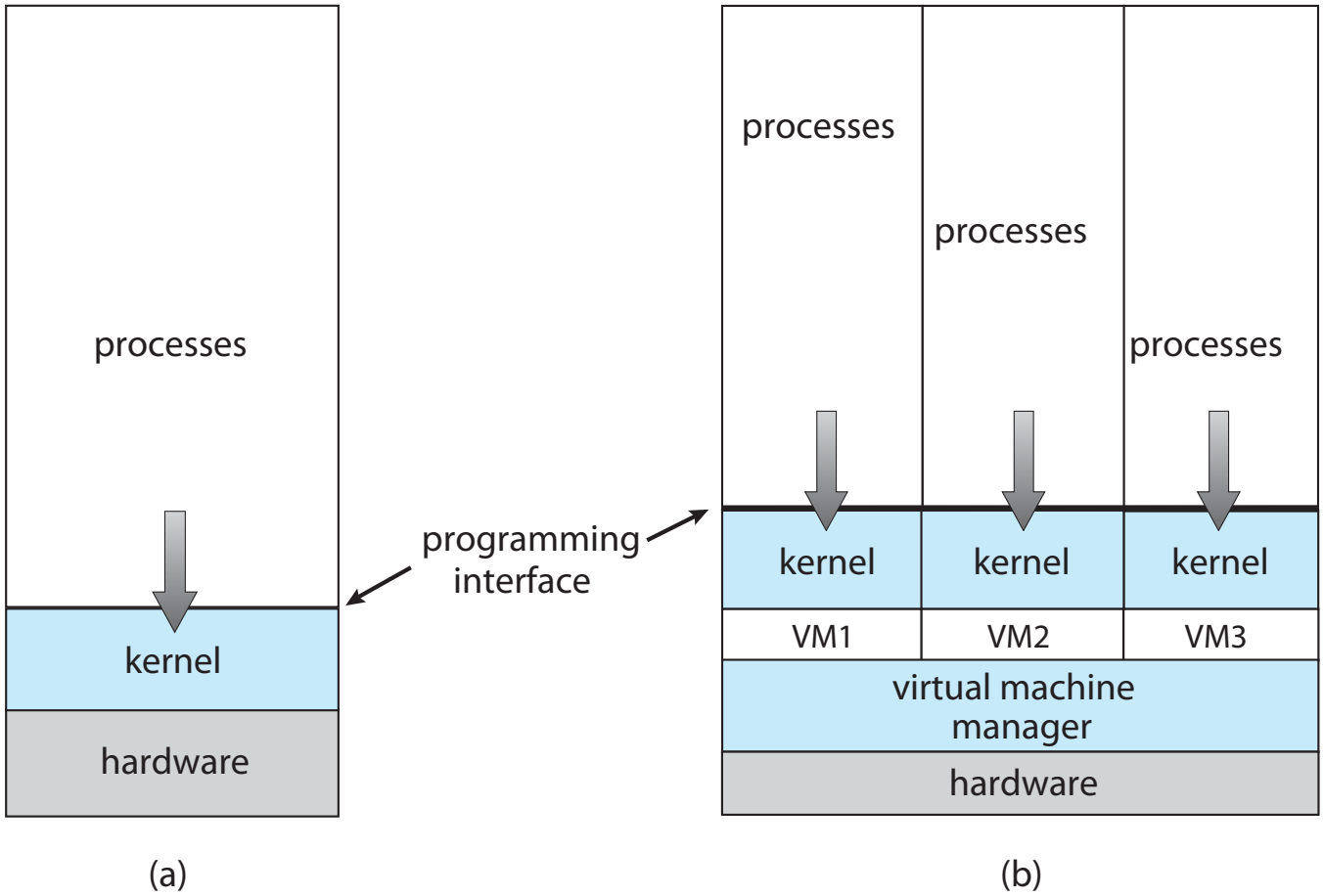
Computing Environments - Virtualization

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Emulation** used when source CPU type different from target type (i.e. PowerPC to Intel x86)
  - Generally slowest method
  - When computer language not compiled to native code – **Interpretation**
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - **Virtual Machine Monitor (VMM)** provides virtualization services

Computing Environments - Virtualization

- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing compute environments within data centers

Computing Environments - Virtualization

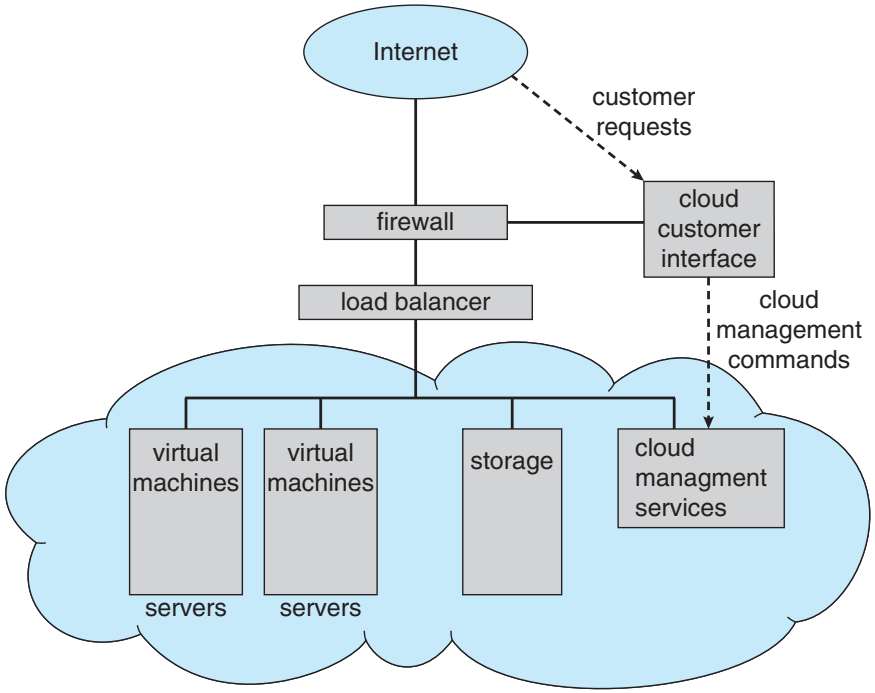


Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization as based on virtualization
  - Amazon **EC2** has thousands of servers, millions of VMs, PBs of storage available across the Internet, pay based on usage
- 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. word processor)
  - 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)

Computing Environments – Cloud Computing

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



Computing Environments – Real-Time Embedded Systems

- Coming soon…

In the first part of this course…

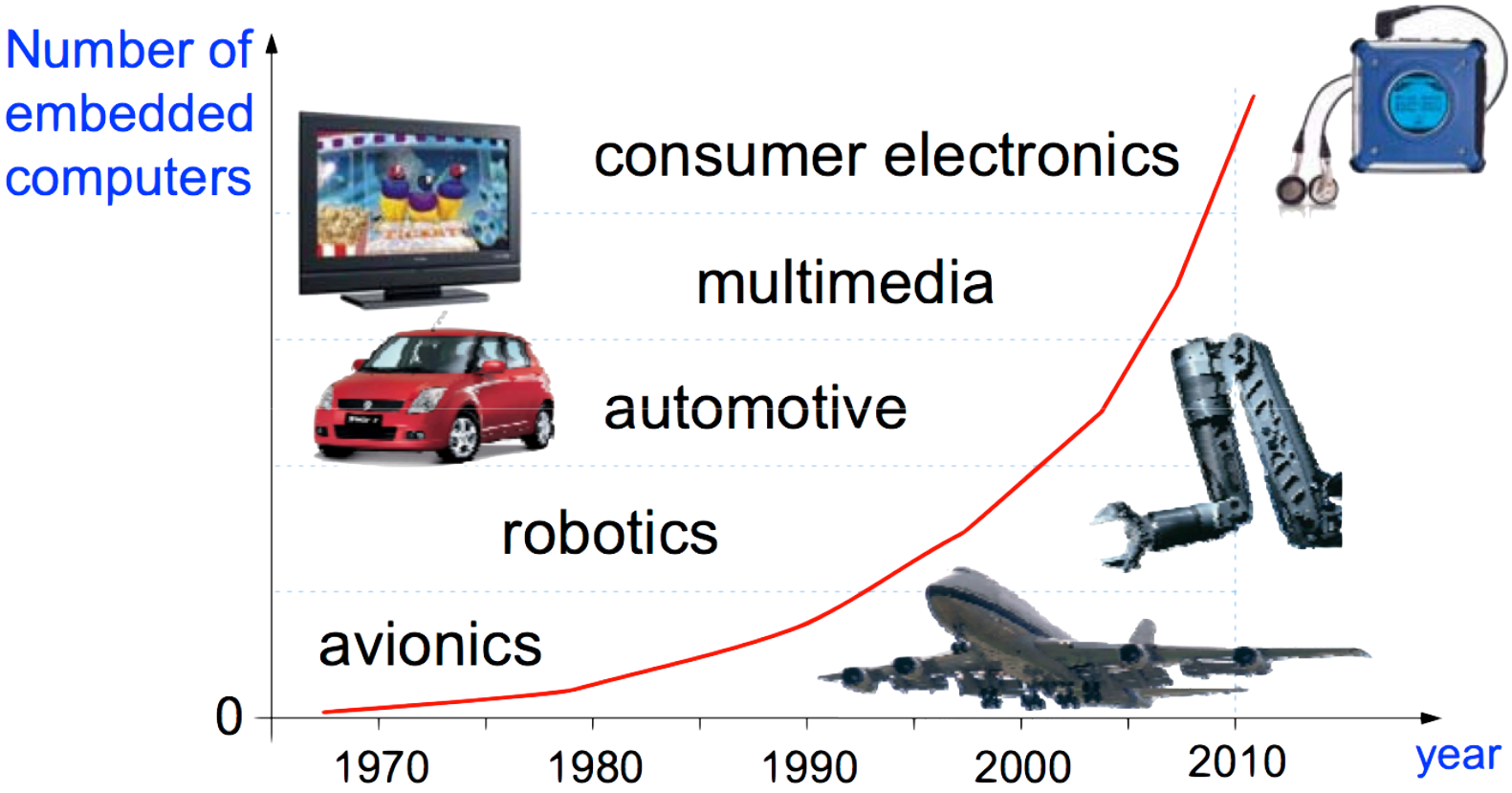
- Fundamentals of general-purpose Operating Systems
- Process management & coordination
  - Process concept
  - Multi-threaded programming
  - Process scheduling
  - Synchronization
  - Deadlocks
- Memory management
  - Memory-Management Strategies
  - Virtual-Memory Management
- Storage & device management
  - I/O Systems

Real-Time Embedded Systems

- **Special-purpose** computing systems
- Must guarantee **bounded** and **predictable response times**
- Predictability of response times must be guaranteed
  - For each critical activity
  - For all possible combinations of events

Evolution of Embedded Systems

- Grown exponentially in several application domains



Computers everywhere

- Today, **98%** of all processors in the planet are embedded in other objects

Smart objects

- This number is expected to grow in the future

Electronic Key

Step counter

Smart shoes

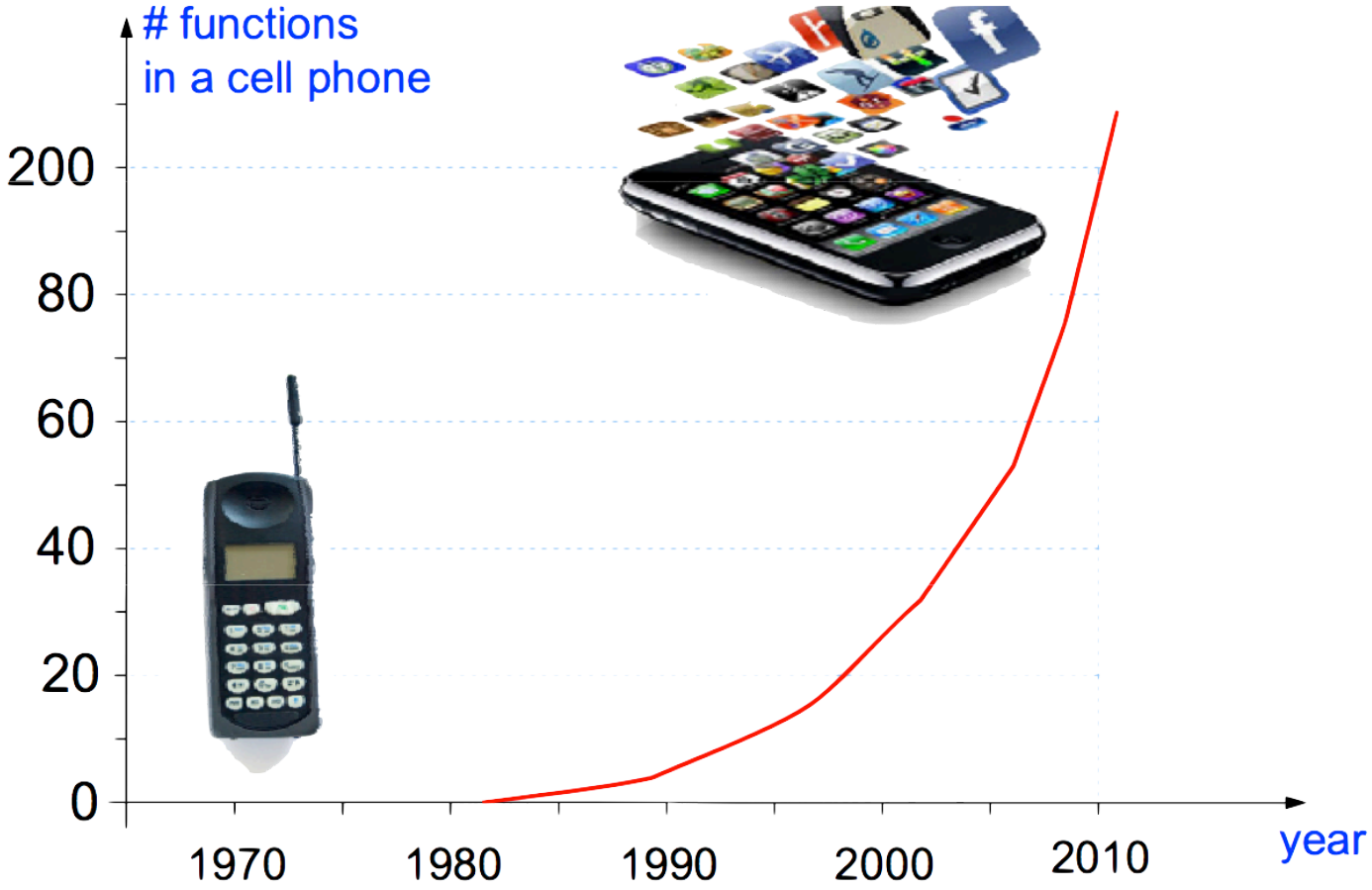
recording pen

Cardio
pulse meter

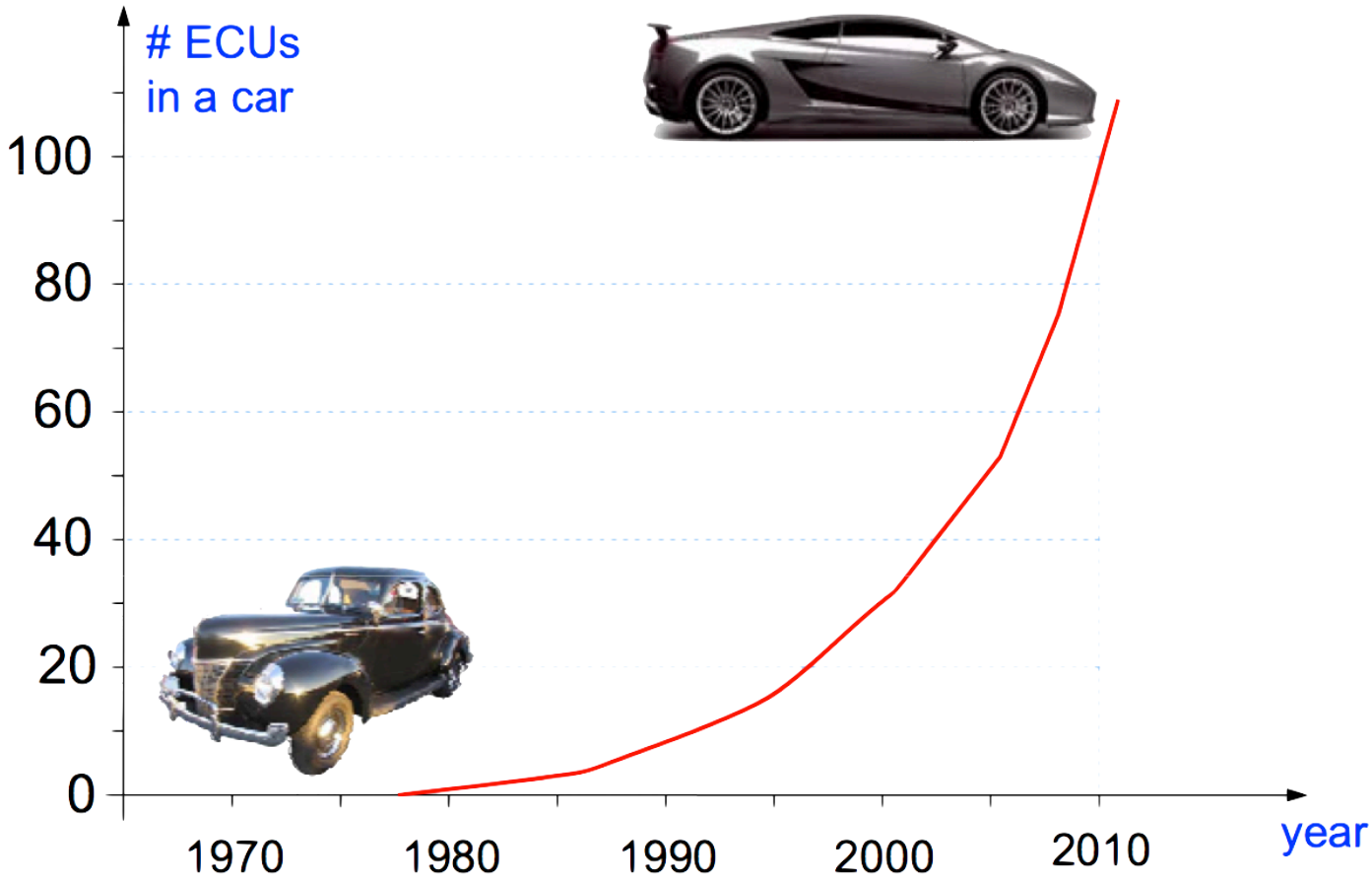
Watch computer

GPS localizer

Increasing complexity



Engine Control Units growth in a car

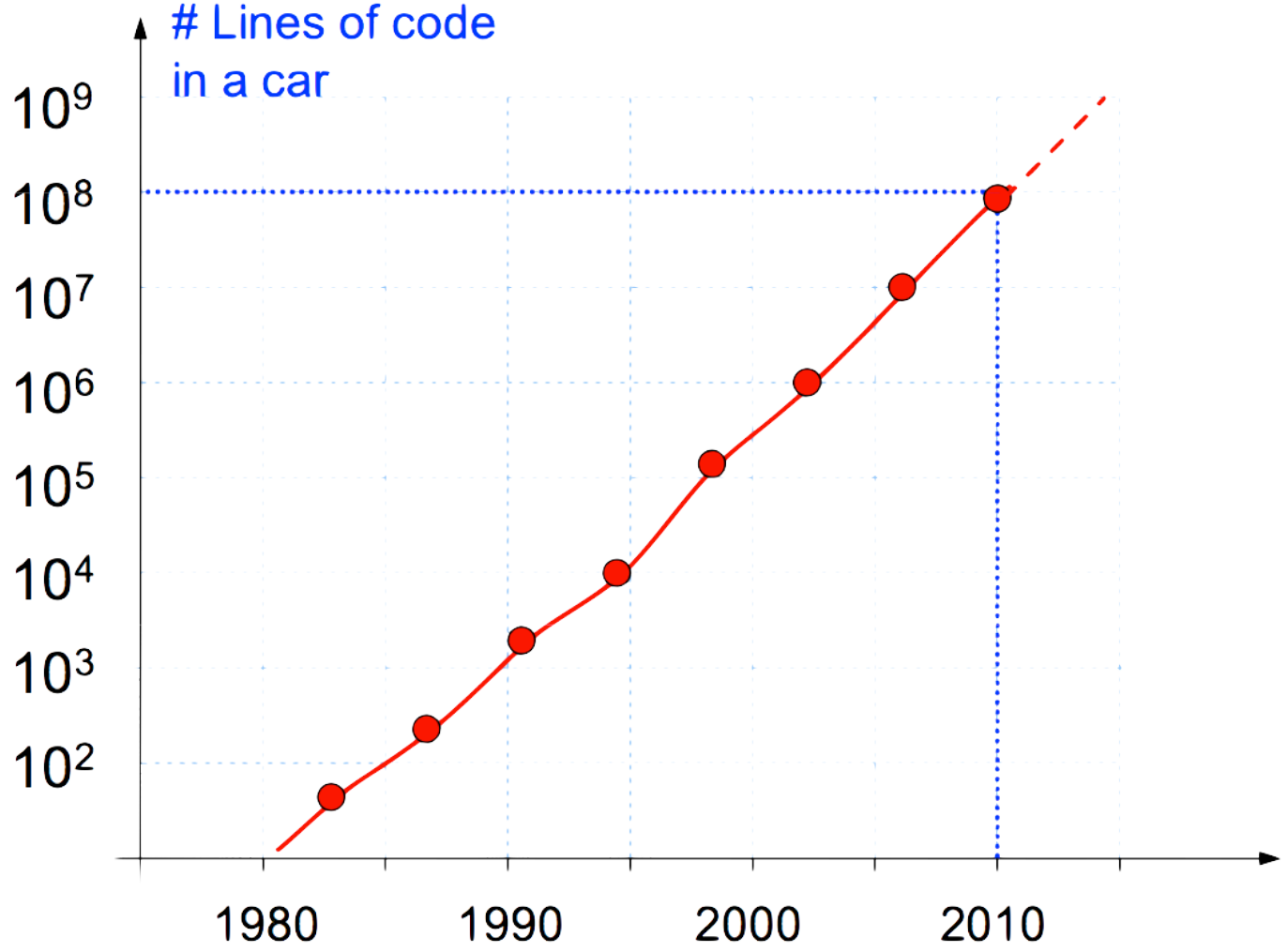


Real-Time Systems Laboratory

Software in a car

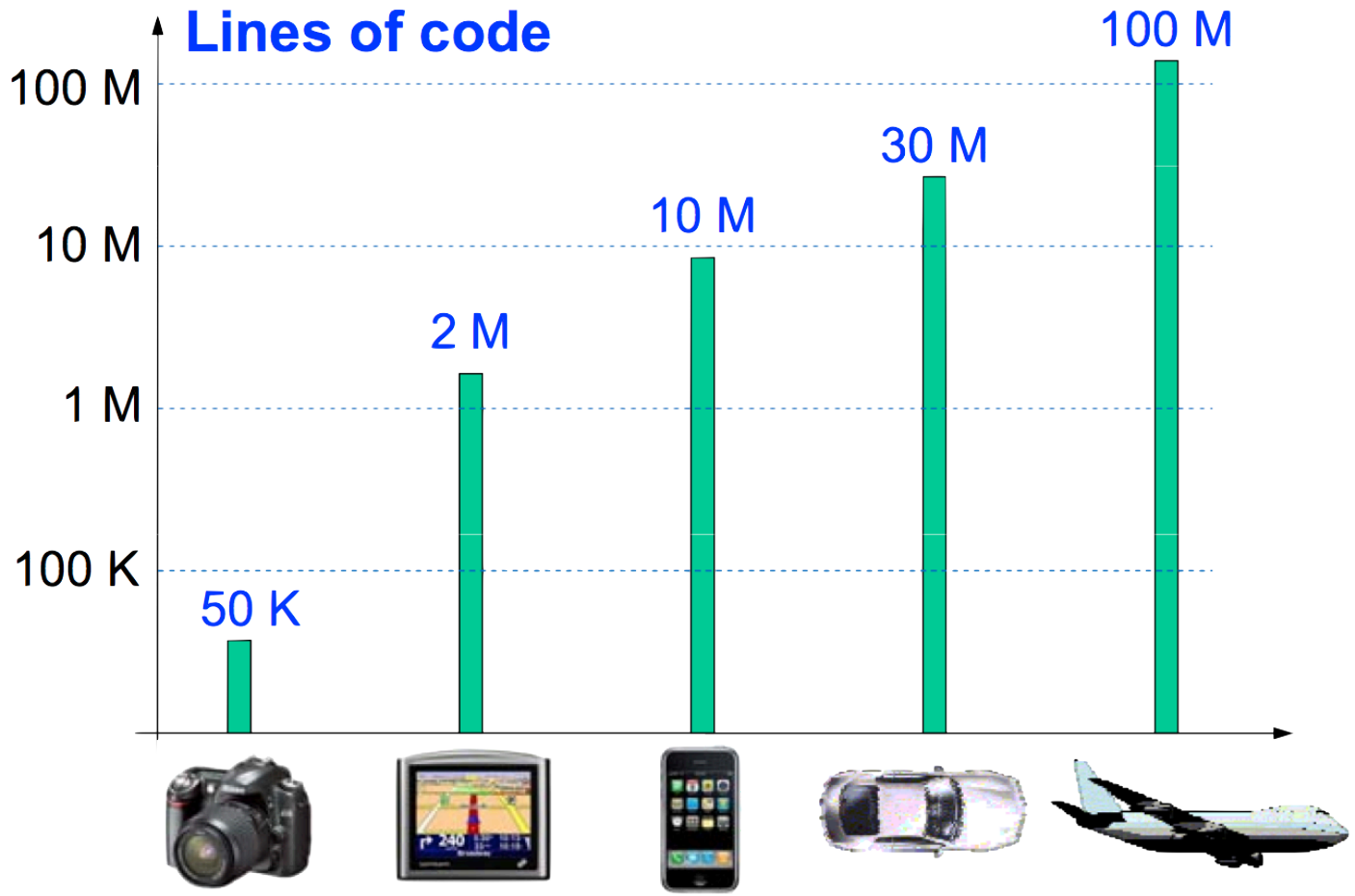
- Car software controls almost everything:
  - **Engine**: ignition, fuel pressure, water temperature, valve control, gear control
  - **Dashboard**: engine status, message display, alarms
  - **Diagnostics**: failure signaling and prediction
  - **Safety**: ABS, ESC, CBC, TCS, …
  - **Assistance**: power steering, navigation, sleep sensors, parking, night vision, collision detection
  - **Comfort**: fan control, air conditioning, music, regulations: steer/ lights/seats/mirrors/glasses

Software evolution in a car



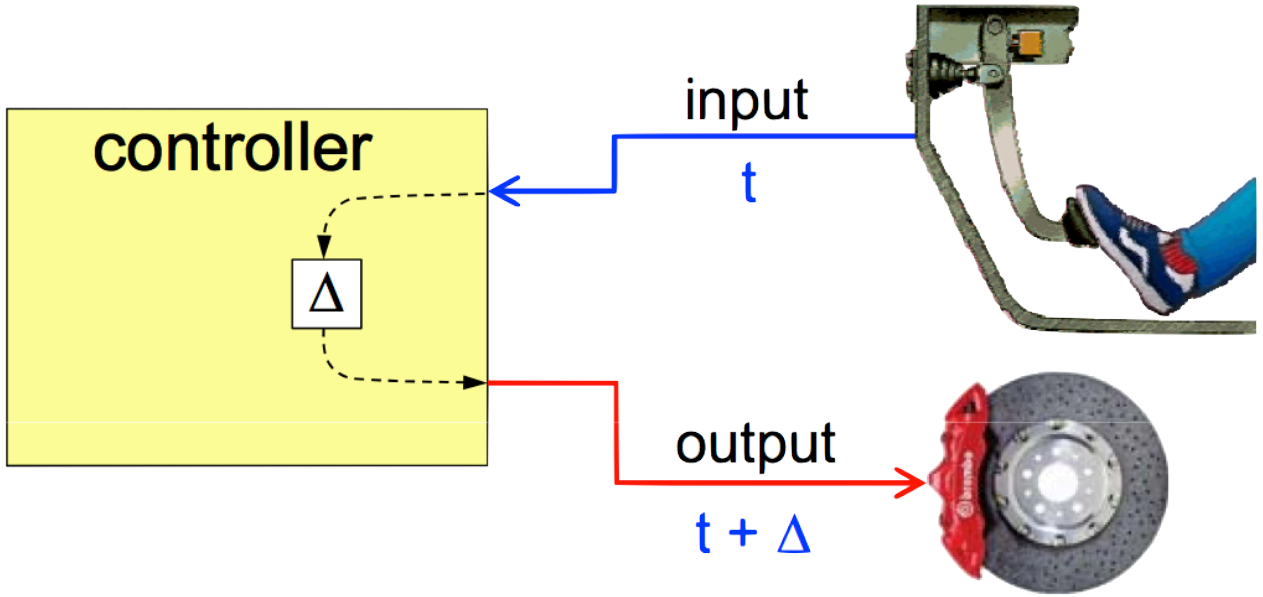
Real-Time Systems Laboratory

Comparing software complexity



Software reliability

- **Reliability** does not only depend on the correctness of single instructions, but also on **when** they are executed:



- A correct action executed **too late** can be **useless** or even **dangerous**

Criticality

QoS management | High performance | Safety critical

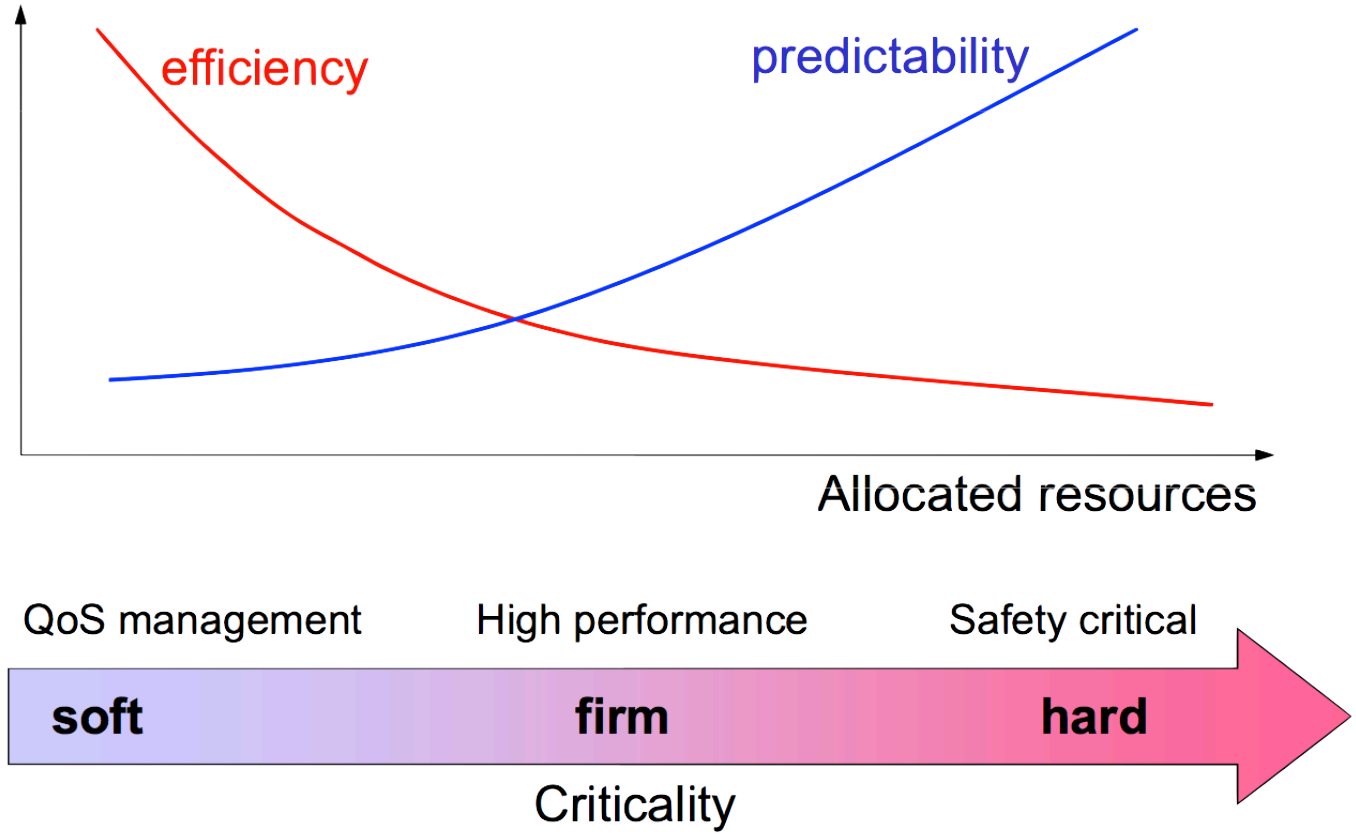
soft → **firm** → **hard**

Criticality

What is special in embedded systems?

FEATURES	REQUIREMENTS
Scarce resources (space, weight, time, memory, energy)	High efficiency in resource management
High concurrency and resource sharing (high task interference)	Limit interference by temporal isolation
Tight environment interaction (causing timing constraints)	High predictability in the response time
High variability on workload and resource demand	Robustness (overload handling and adaptivity)

Requirements



In the last part of this course…

- Study software methodologies and algorithms to increase **predictability** in computing systems
- We consider embedded computing system consisting of several concurrent activities subject to **timing constraints**
- We will see how to **model** and **analyze** a real-time application to predict worst-case response times and **verify its feasibility** under a set of constraints

Quizzes

- Multiprogramming and multitasking are the same thing
- An operating system's process management activities include suspending and resuming processes
- An operating system should never keep track of which parts of memory are currently being used and by whom
- Caching is used to increase the size of a system's memory
- Buffering is used for transferring data
- Protection is concerned with defending a system against internal and external attacks
- Client-server and Peer-to-peer are two models of distributed computing environments
- What matters in real-time embedded systems is performance