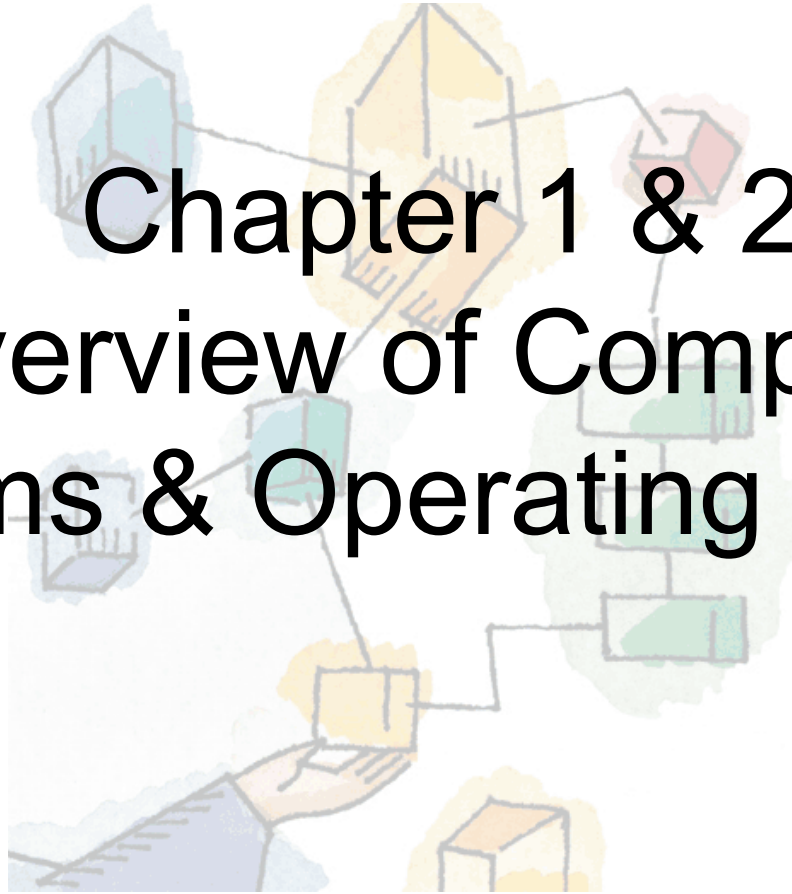
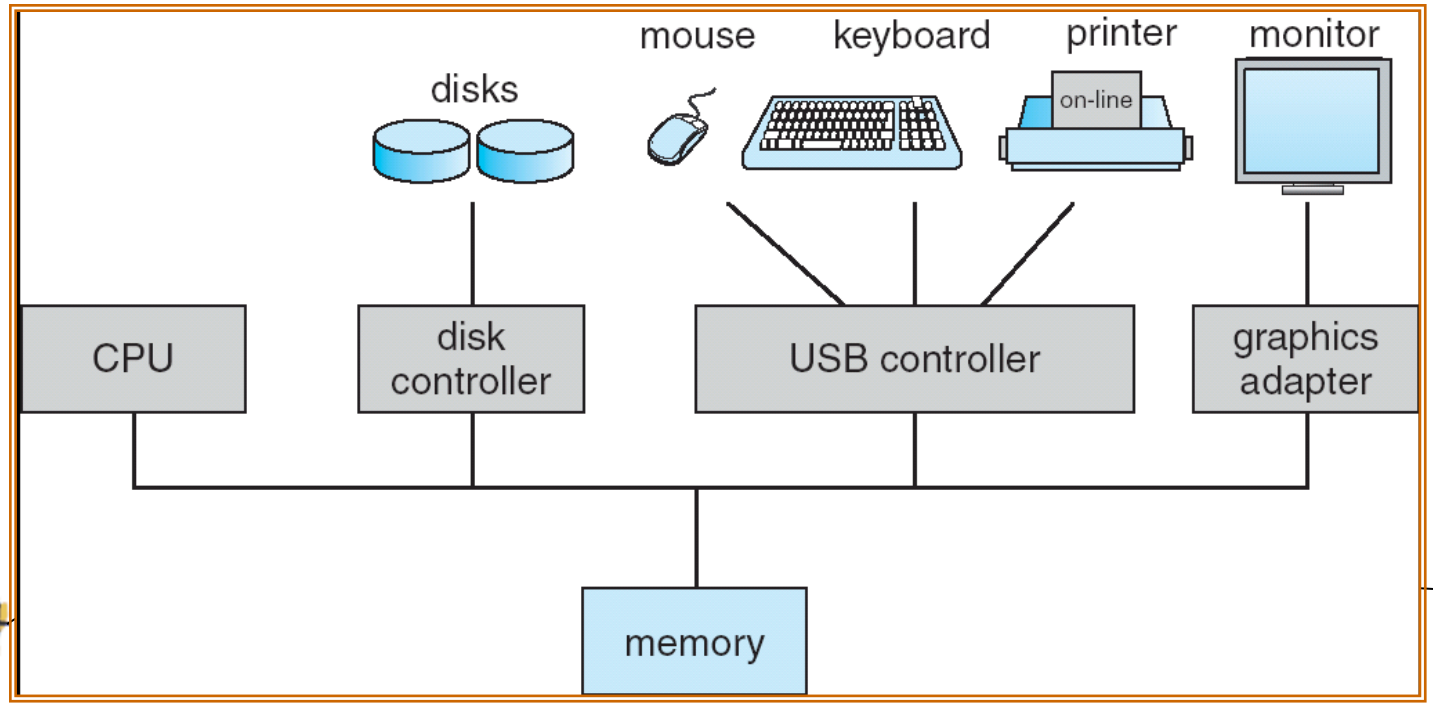
Operating Systems:
Internals and Design Principles
William Stallings



Chapter 1 & 2
Overview of Computer Systems & Operating systems

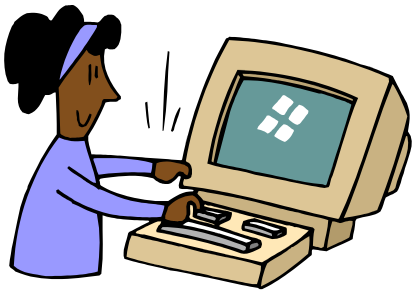
Computer System Organization

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



Basic Elements

- Processor
- I/O Modules
- Main Memory
- System Bus



Computer Components: Top-Level View

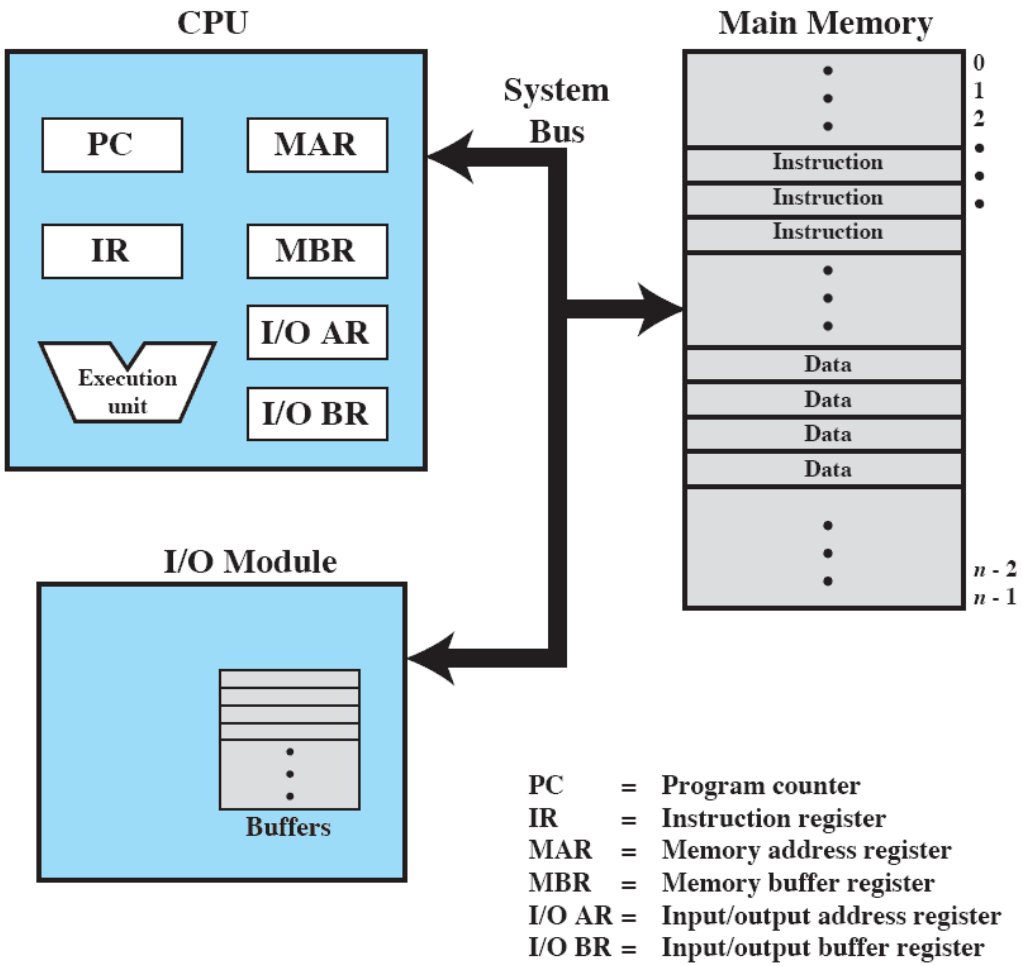
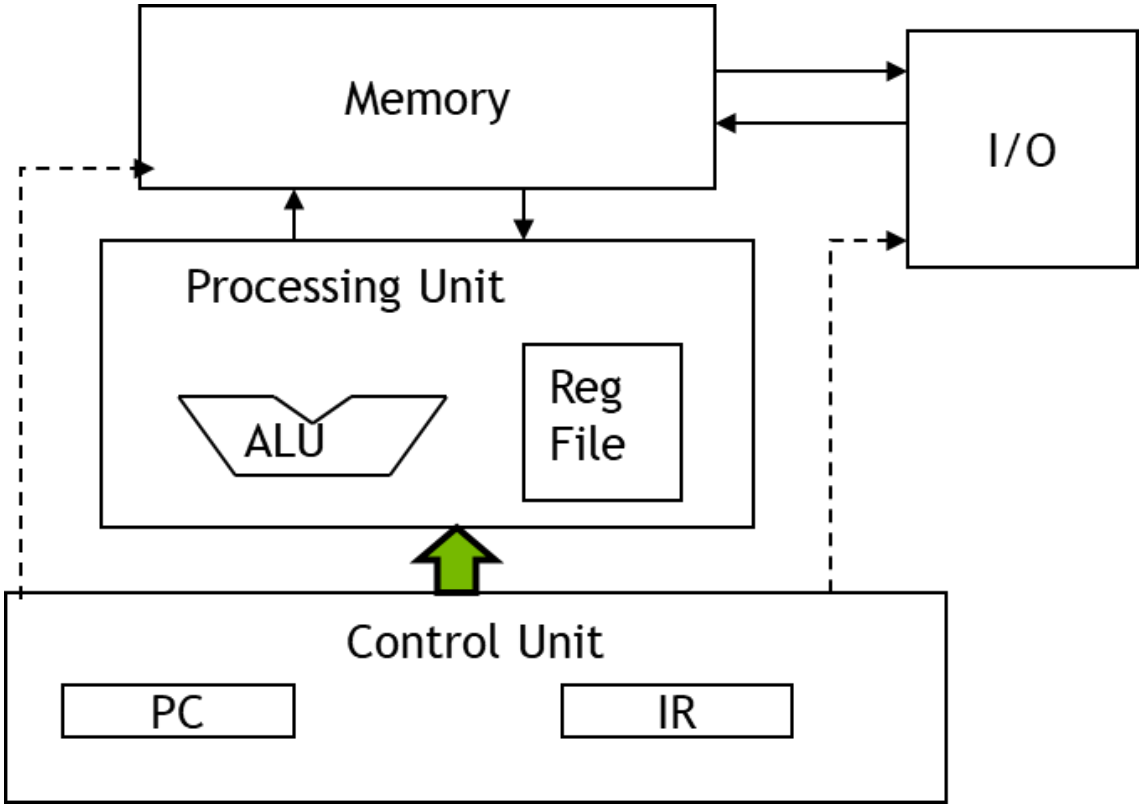
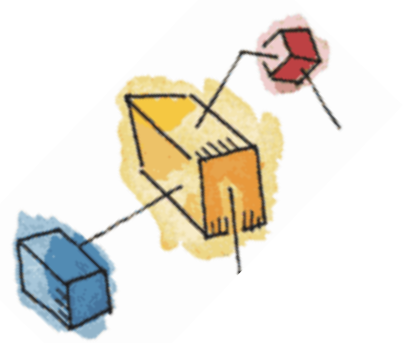


Figure 1.1 Computer Components: Top-Level View

The Von-Neumann Model





Instruction Execution

- A program consists of a set of instructions stored in memory

Two steps:

  - processor reads (fetches) instructions from memory
  - processor executes each instruction

- Program counter (PC) holds address of the instruction to be fetched next
  - PC is incremented after each fetch

Basic Instruction Cycle

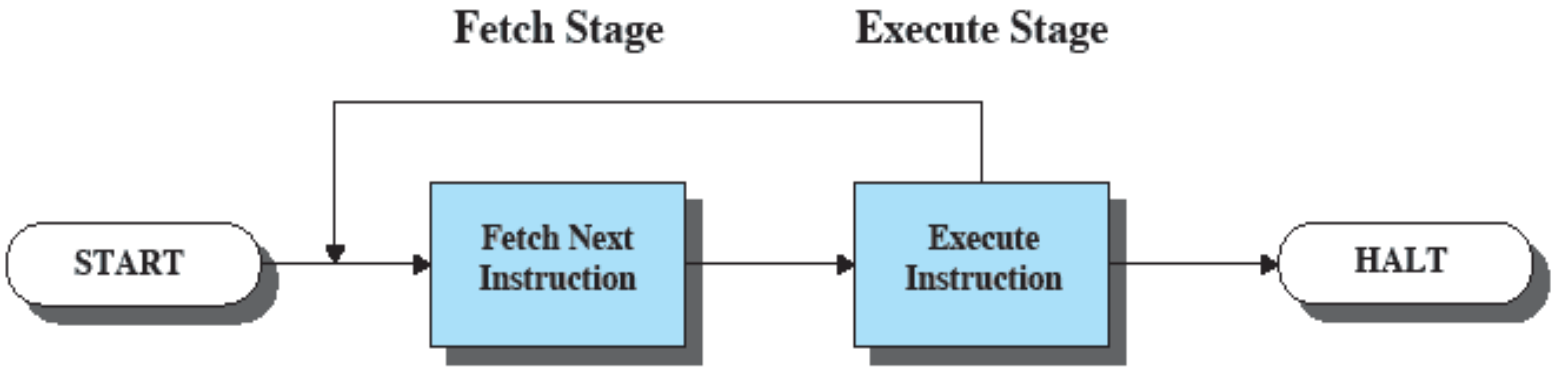


Figure 1.2 Basic Instruction Cycle

Example of Instruction Execution

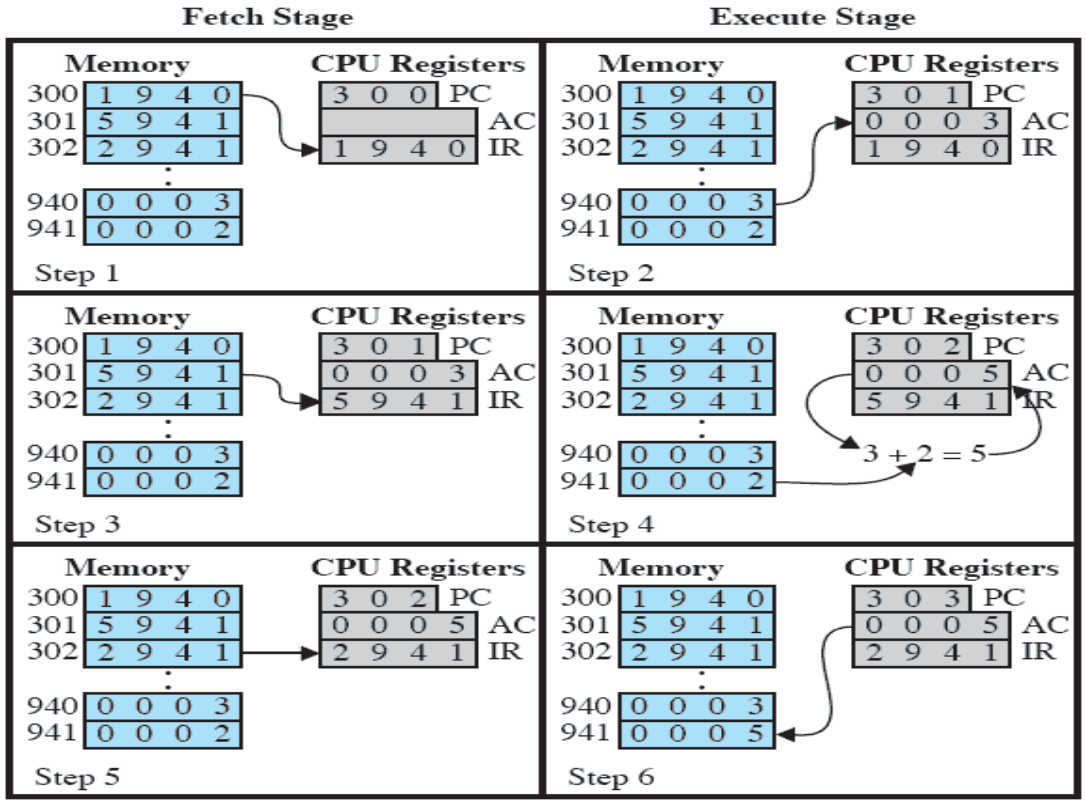


Figure 1.4 Example of Program Execution (contents of memory and registers in hexadecimal)

Interrupts

- Interrupt the normal sequencing of the processor
- Provided to improve processor utilization
  - most I/O devices are slower than the processor
  - processor must pause to wait for device
  - wasteful use of the processor

Common Classes of Interrupts

Table 1.1 Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure, such as power failure or memory parity error.

Instruction Cycle with Interrupts

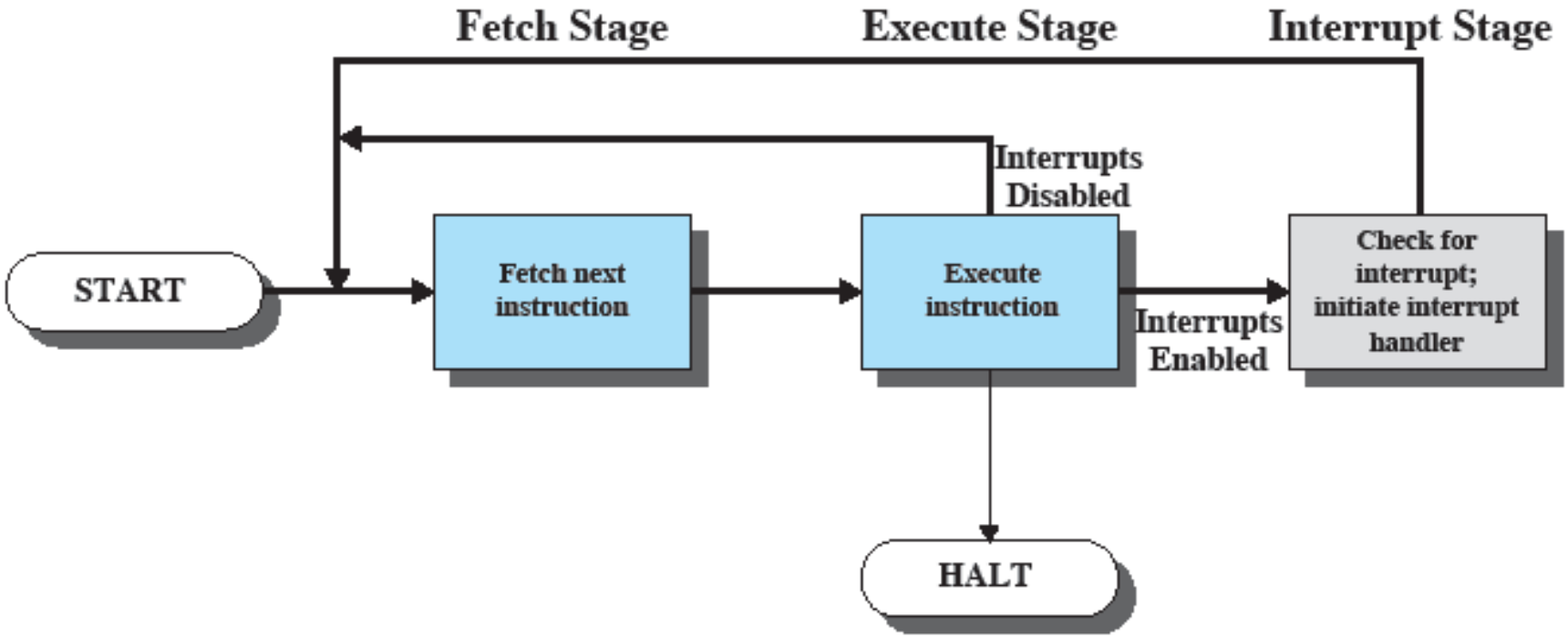
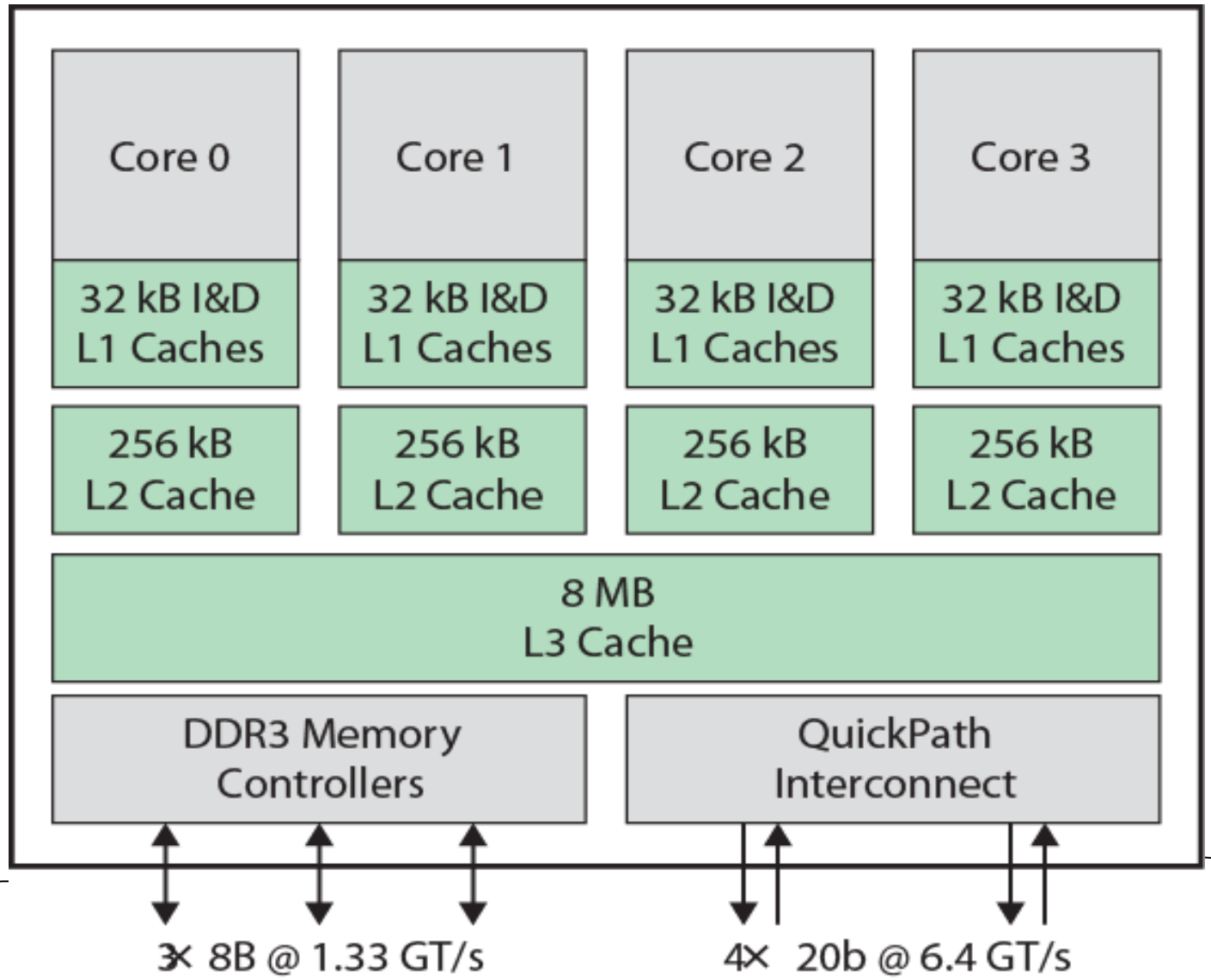


Figure 1.7 Instruction Cycle with Interrupts

Multicore Computer

- Also known as a chip multiprocessor
- Combines two or more processors (cores) on a single piece of silicon (die)
  - each core consists of all of the components of an independent processor
- In addition, multicore chips also include L2 cache and in some cases L3 cache

Intel Core i7



Memory Hierarchy

- Major constraints in memory
  - Amount
  - speed
  - expense
- Memory must be able to keep up with the processor
- Cost of memory must be reasonable in relationship to the other components

The Memory Hierarchy

- Going down the hierarchy:
  - decreasing cost per bit
  - increasing capacity
  - increasing access time
  - decreasing frequency of access to the memory by the processor

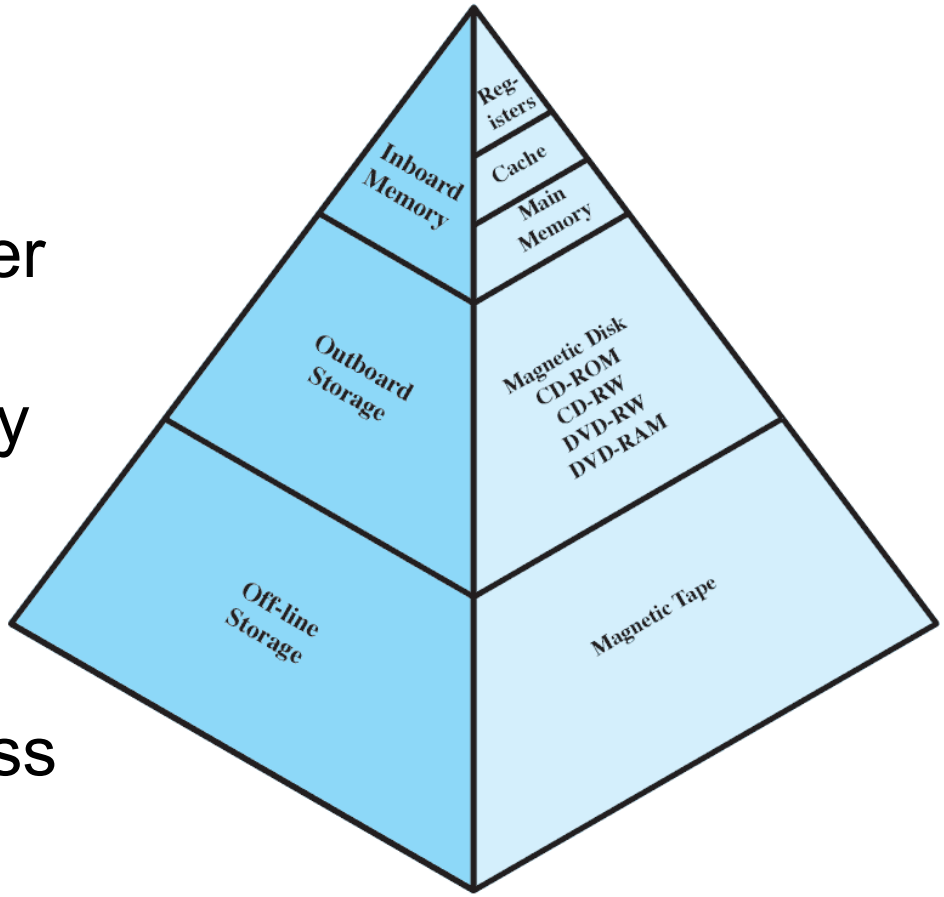
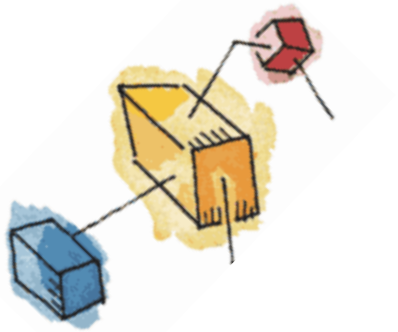


Figure 1.14 The Memory Hierarchy



Principle of locality

- Memory references by the processor tend to cluster
  - Temporal locality and spatial locality
- Data is organized so that the percentage of accesses to each successively lower level is substantially less than that of the level above
- Can be applied across more than two levels of memory

I/O Techniques

- When the processor encounters an instruction relating to I/O, it executes that instruction by issuing a command to the appropriate I/O module

Three techniques are possible for I/O operations:

Programmed I/O	Interrupt-Driven I/O	Direct Memory Access (DMA)

Layers and Views

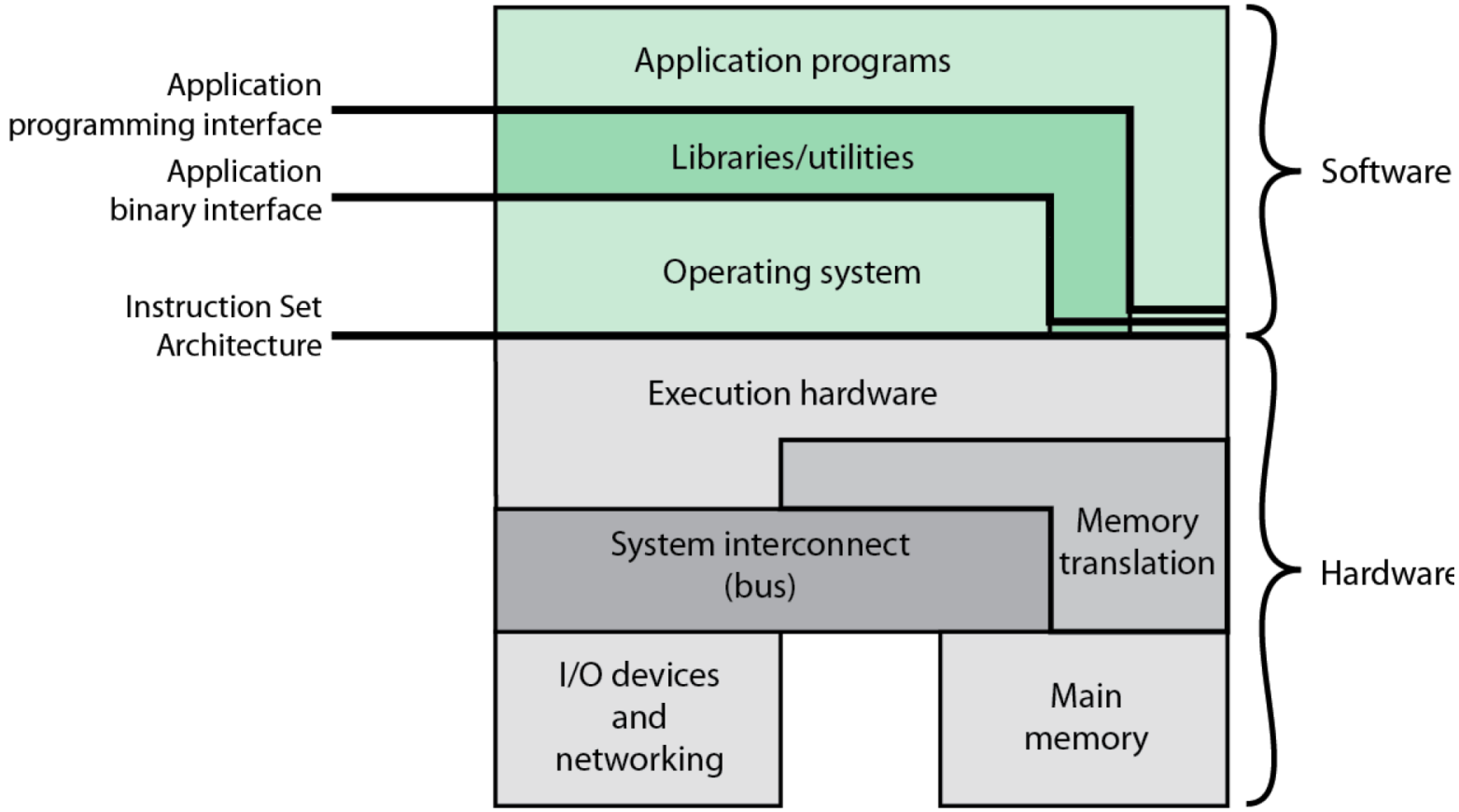
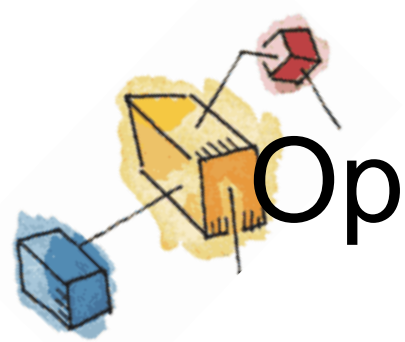


Figure 2.1 Computer Hardware and Software Infrastructure



Operating System Services

- Operating systems provide an environment for the execution of programs.
- Operating systems provide certain services to:
  - Programs
  - Users of those programs
- Basically two types of services:
  - Set of operating-system functions for ensuring the efficient operation of the system itself via resource sharing
  - Set of operating-system services provides functions that are helpful to the user

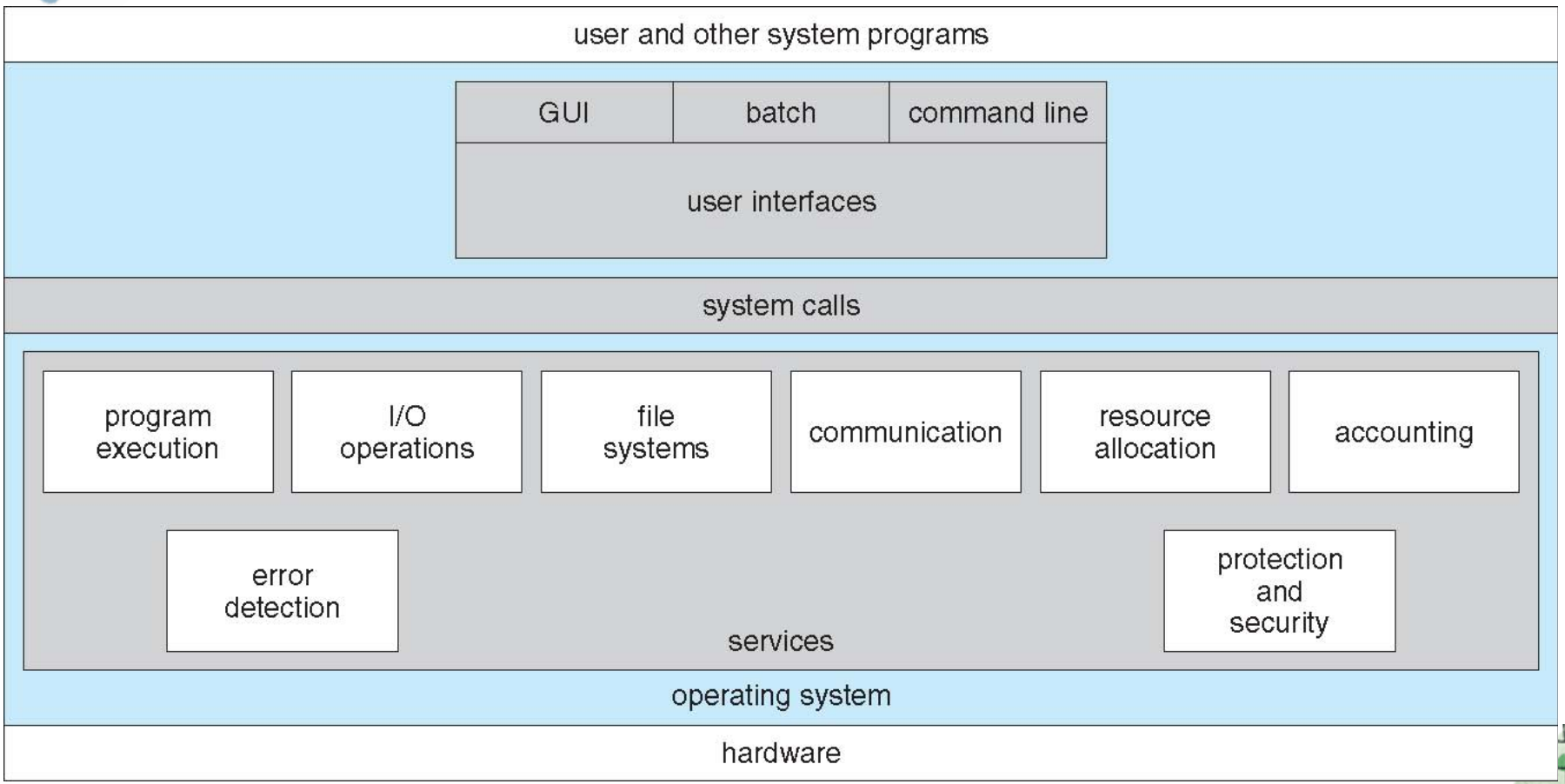
System Calls

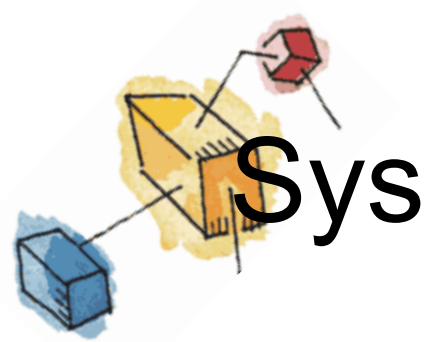
- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level **Application Programming Interface (API)** rather than direct system call use
- Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)

Types of System Calls

- Process control
  - load, execute; create process, terminate process; get process attributes, set process attributes; wait for time, wait for events; allocate and free memory; end, abort;
- File management
  - ...
- Device management
  - ...
- Information maintenance
  - ...
- Communications
  - ...

Operating System Services

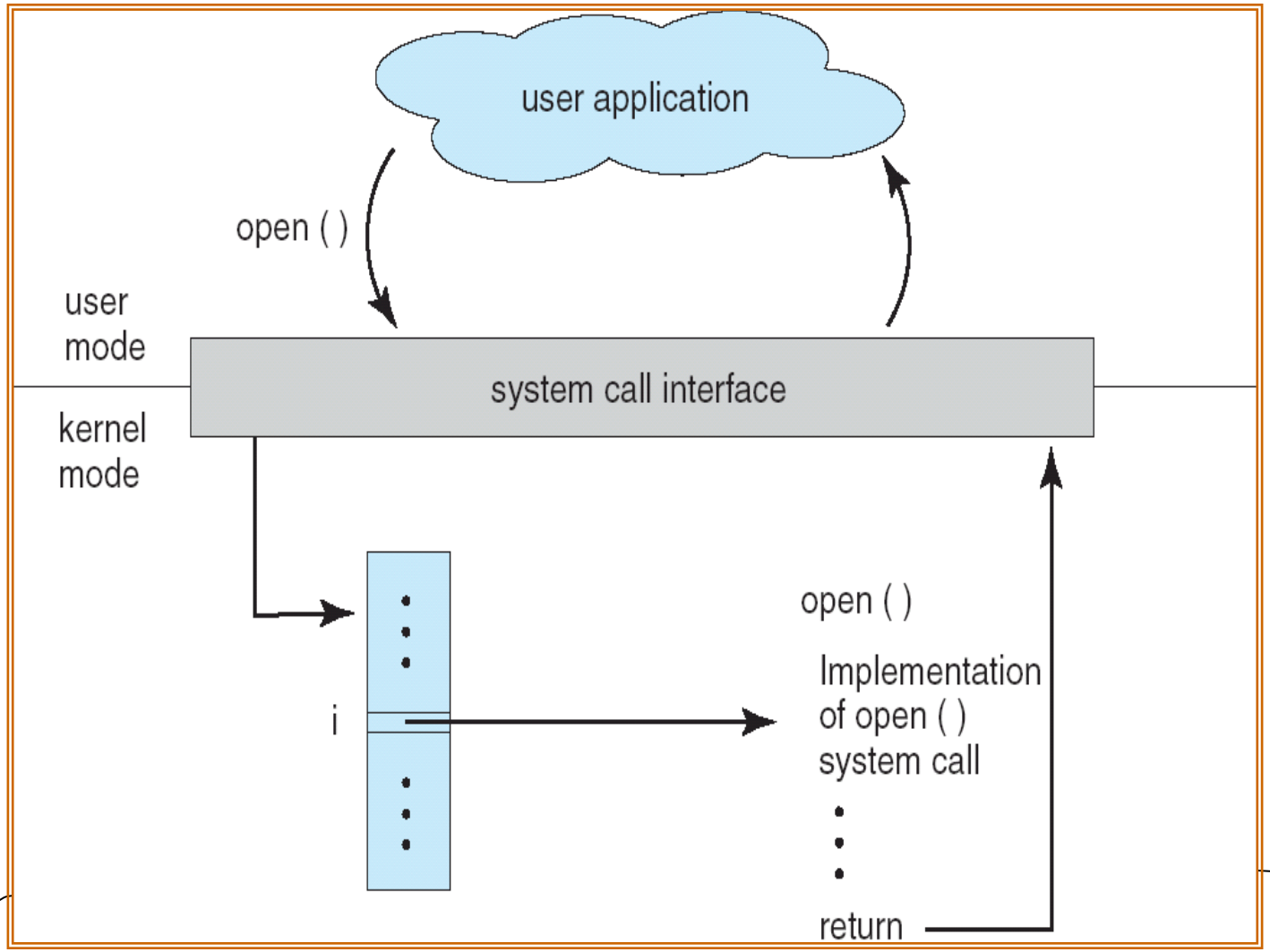




System Call Implementation

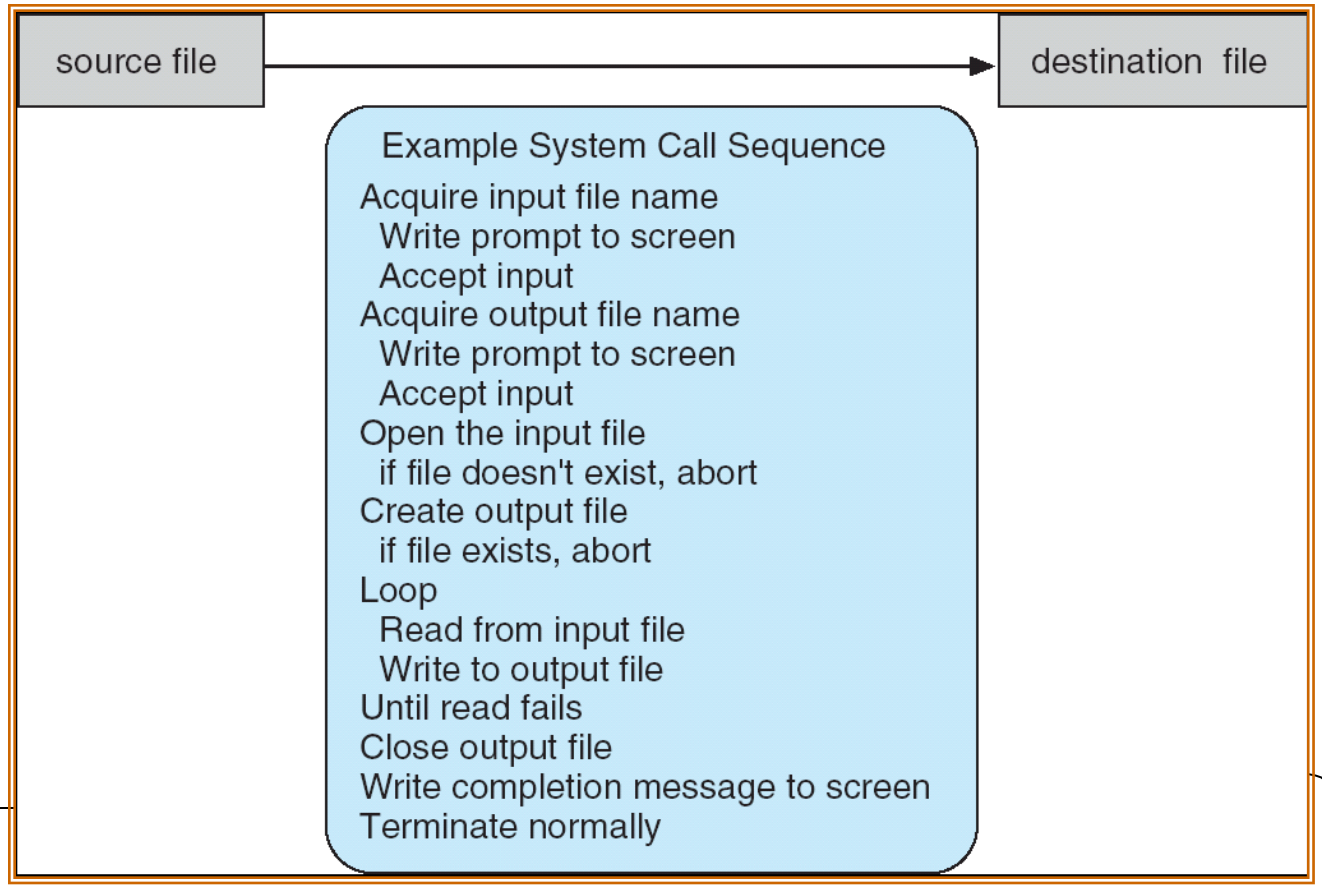
- Typically, a number associated with each system call
  - System-call interface maintains a table indexed according to these numbers
- The system call interface invokes intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented
  - Just needs to obey API and understand what OS will do as a result call
  - Most details of OS interface hidden from programmer by API
    - Managed by run-time support library (set of functions built into libraries included with compiler)

API – System Call – OS Relationship

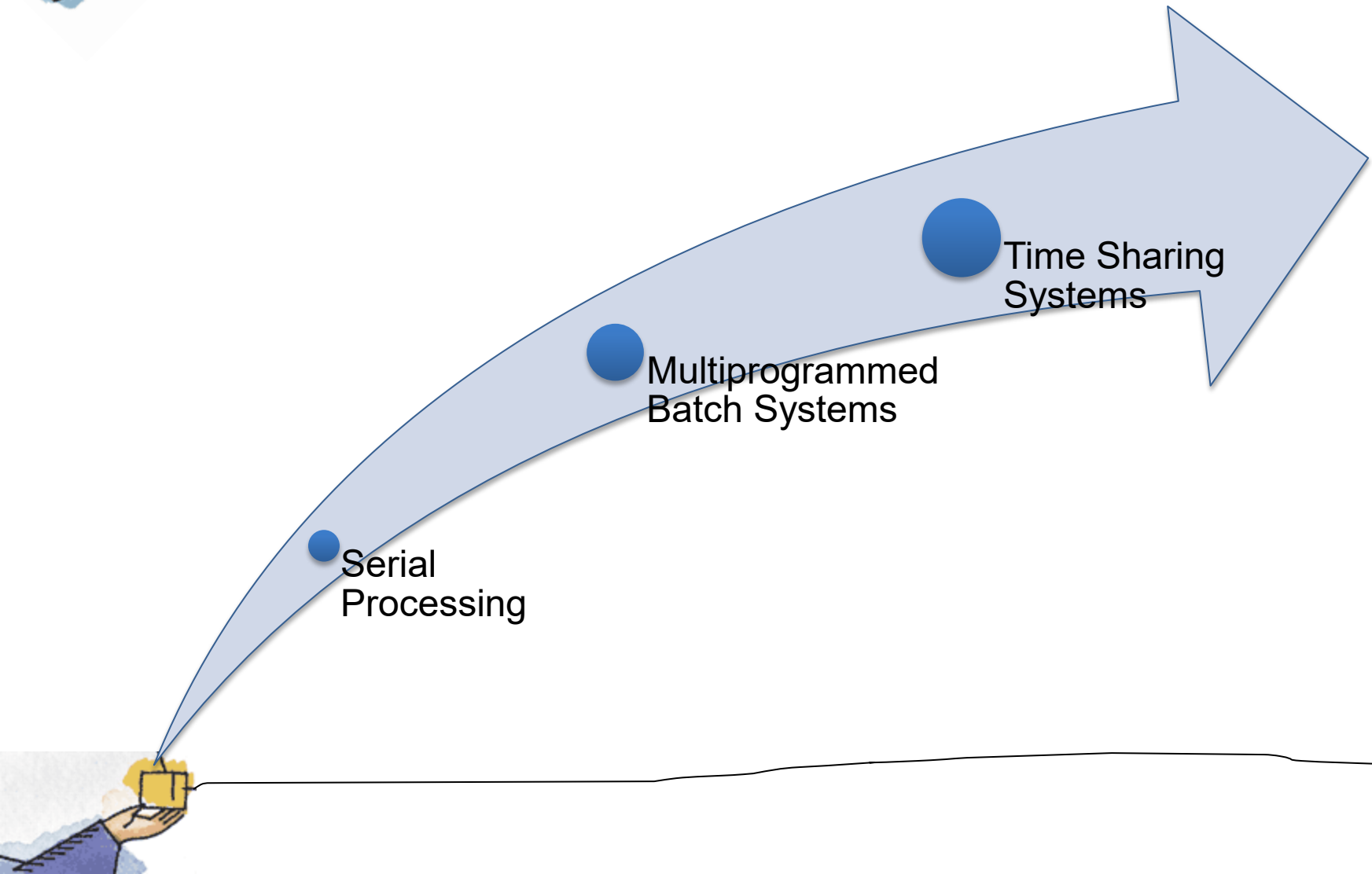


Example of System Calls

- System call sequence to copy the contents of one file to another file



Evolution of Operating Systems

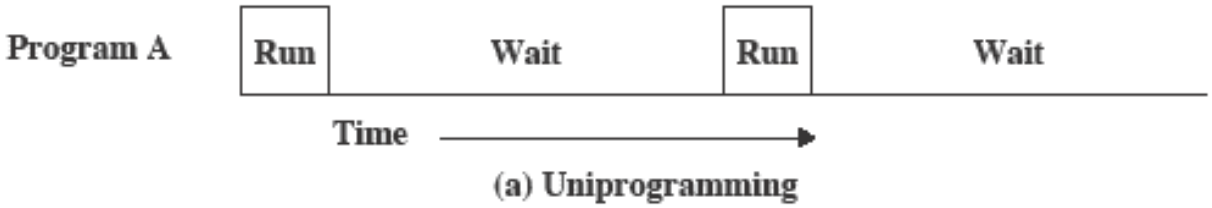


Evolution of Operating Systems

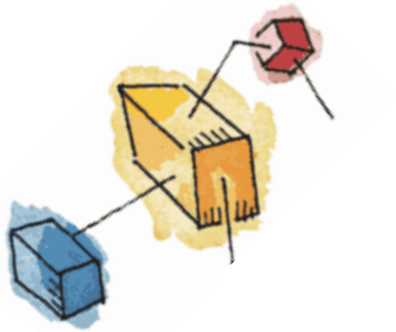
- A major OS will evolve over time for a number of reasons:
  - Hardware upgrades plus new types of hardware
  - New services
  - Fixes

Uniprogramming

- Processor must wait for I/O instruction to complete before proceeding

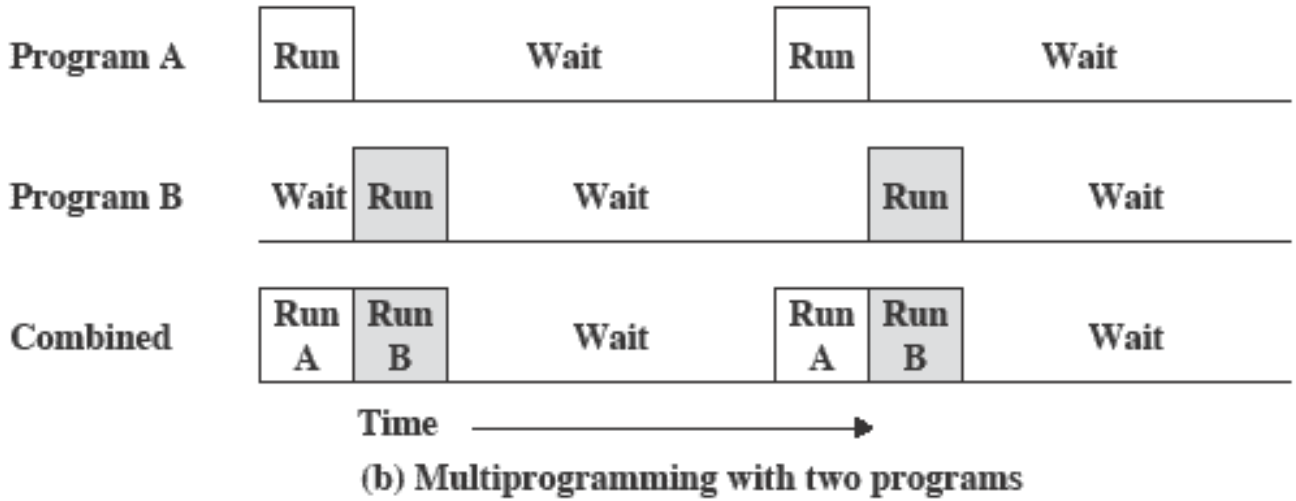


(a) Uniprogramming



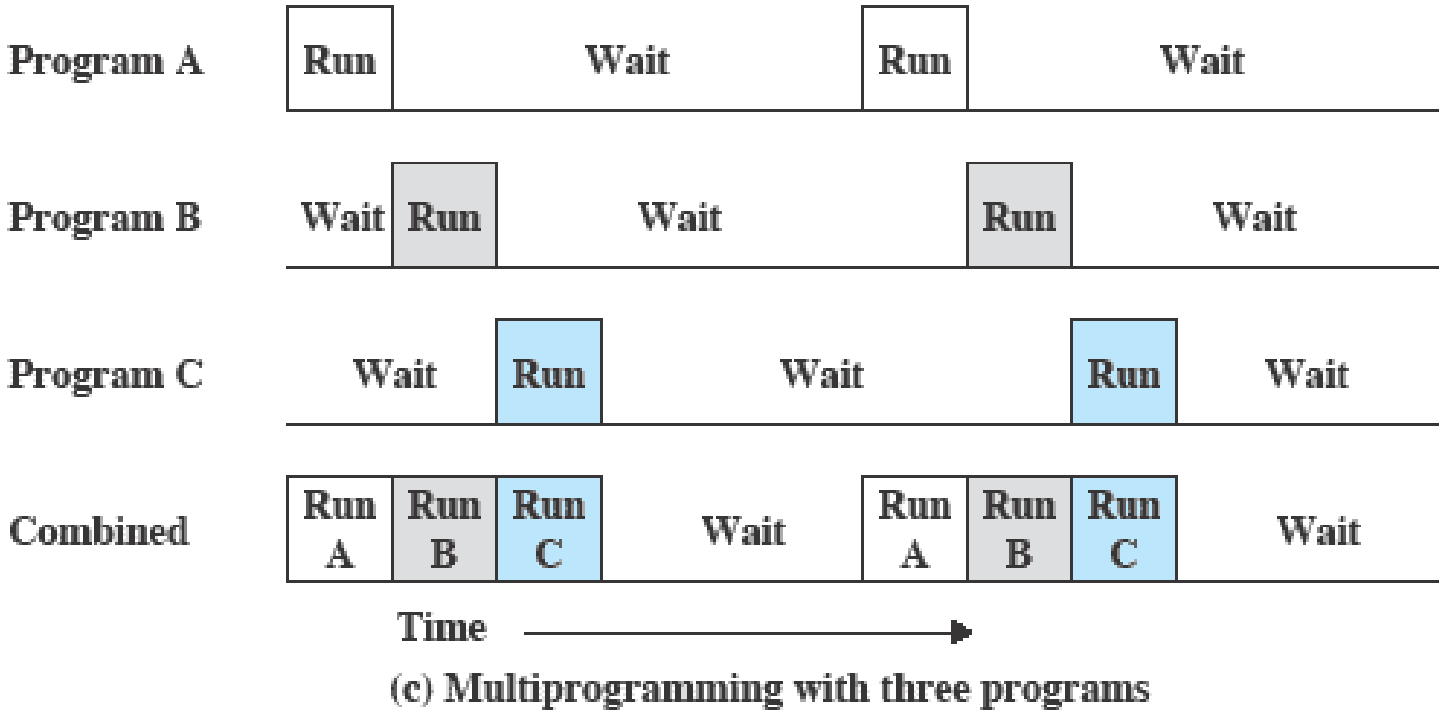
Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job

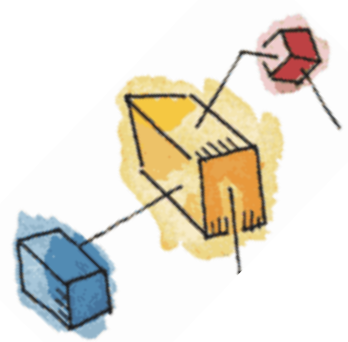


(b) Multiprogramming with two programs

Multiprogramming

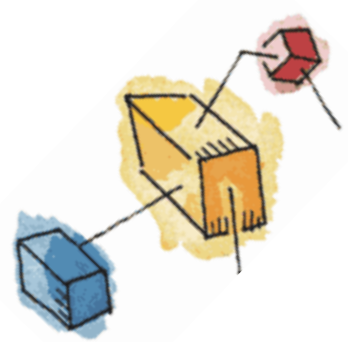


(c) Multiprogramming with three programs



Time Sharing Systems

- Using multiprogramming to handle multiple interactive jobs
- Processor’s time is shared among multiple users
- Multiple users simultaneously access the system through terminals



Time Sharing Systems

- **Timesharing (multitasking)**: 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**
  - To ensure orderly execution ⇨ **Synchronization** and **Communication**
  - **Virtual memory** allows execution of processes not completely in memory
  - Also need mechanisms for **Security and Protection**

Hardware Features to Support OS

- Memory protection
  - Does not allow the memory area containing the OS to be altered
- Timer
  - Prevents a job from monopolizing the system
- Interrupts
  - Early computer models did not have this capability
- Privileged instructions
  - Certain machine level instructions can only be executed by the OS

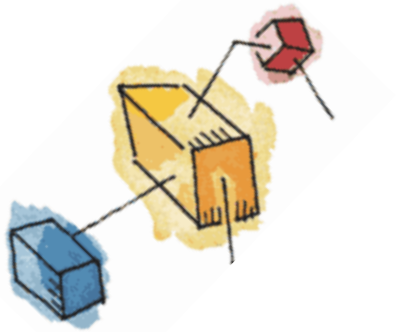
Dural Mode Operation

User Mode

- user program executes in user mode
- certain areas of memory are protected from user access
- certain instructions may not be executed

Kernel Mode

- OS executes in kernel mode
- privileged instructions may be executed
- protected areas of memory may be accessed



System Boot

- When power initialized on system, execution starts at a fixed memory location
  - Firmware ROM used to hold initial boot code
- Operating system must be made available to hardware so hardware can start it
  - Small piece of code – **bootstrap loader**, stored in **ROM** or **EEPROM** locates the kernel, loads it into memory, and starts it
  - Sometimes two-step process where **boot block** at fixed location loaded by ROM code, which loads bootstrap loader from disk
- Common bootstrap loader, **GRUB**, allows selection of kernel from multiple disks, versions, kernel options
- Kernel loads and system is then **running**

Summary

- Basic Elements
  - processor, main memory, I/O modules, system bus
- Instruction execution
- Memory Hierarchy
- I/O techniques
- Multicore

Summary

- Operating System services
  - System calls
- Multiprogramming and time sharing
- Hardware features to support OS
- System boot