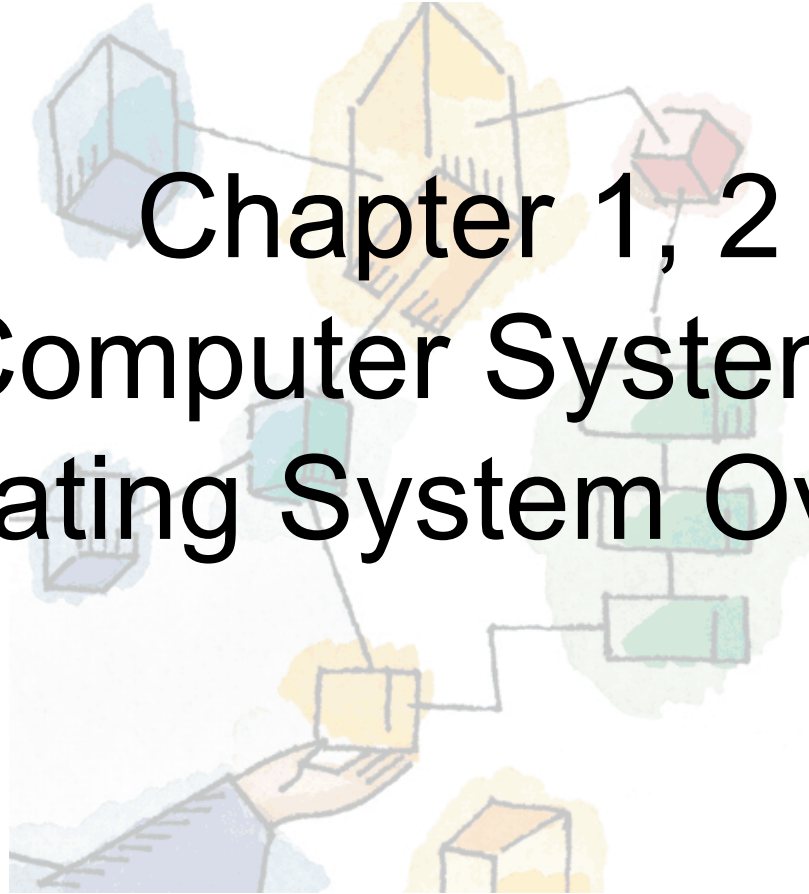
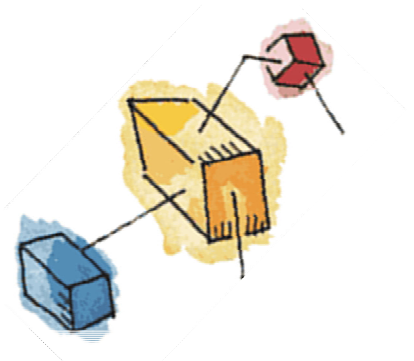


*Operating Systems:
Internals and Design Principles*
William Stallings

Chapter 1, 2
**Computer System &
Operating System Overview**





Roadmap

→ Basic Elements

- Instruction Execution
- Interrupts
- The Memory Hierarchy
- I/O Communication Techniques
- Operating System Objectives & Functions

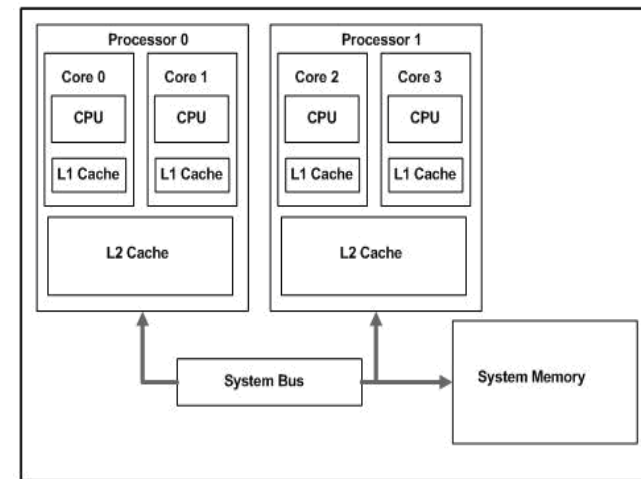
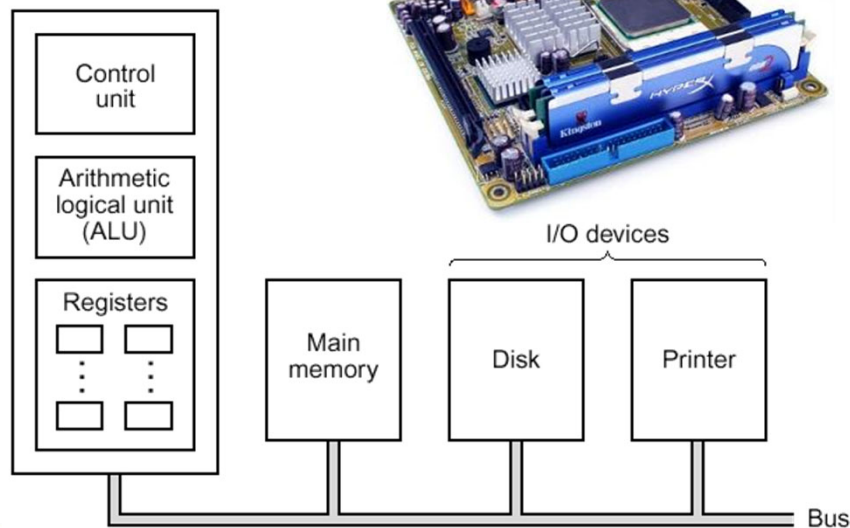


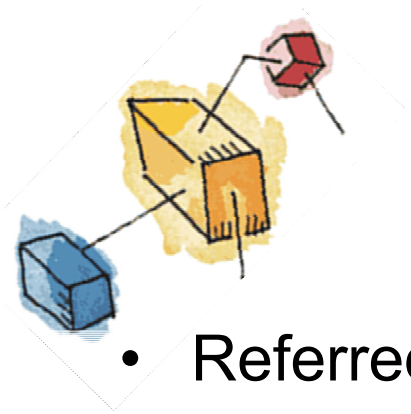
A Computer's Basic Elements

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

The OS *manages* computer system hardware on behalf of its users, so it is important to know them before studying OS.

Central processing unit (CPU)

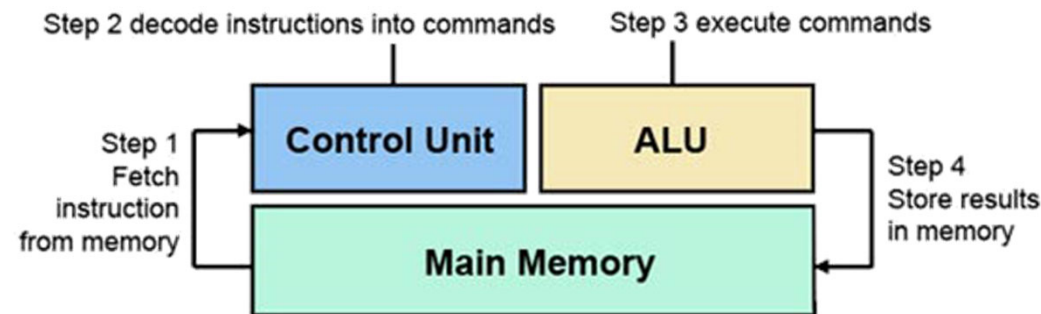




Processor

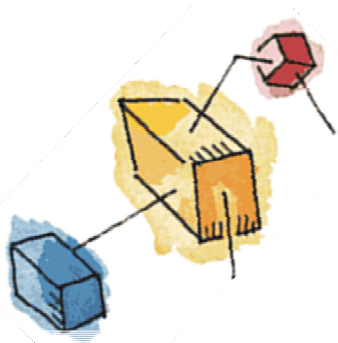
- Referred to as the *Central Processing Unit* (CPU)
- Two internal components
 - Control Unit
 - Arithmetic & Logic Unit (ALU)
- Controls operation, performs data processing
- Works repeatedly and continuously in cycles to execute instructions

Machine Cycle



<http://www.computerhope.com>

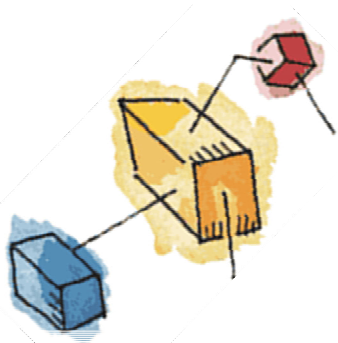




More about Processor

- Microprocessor
 - Invention that brought about desktop and handheld computing
 - Contains a processor on a single chip
- Multiprocessors
 - Each chip contains multiple processors (cores)
- Graphical Processing Units (GPU's)
 - Provide efficient computation on arrays of data using Single-Instruction Multiple Data ([SIMD](#)) techniques

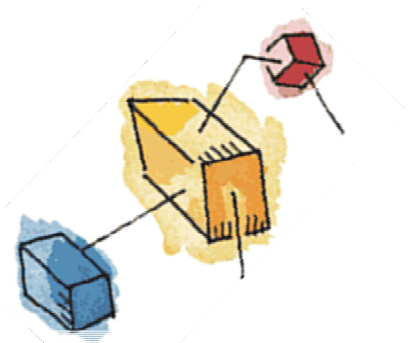




More about Processor

- Digital Signal Processors (DSPs)
 - Encoding/decoding speech and video (codecs)
 - Provide support for encryption and security
- System on a Chip (SoC)
 - Satisfy the requirements of handheld devices
 - CPUs, caches, DSPs, GPUs, I/O devices and main memory are on the same chip



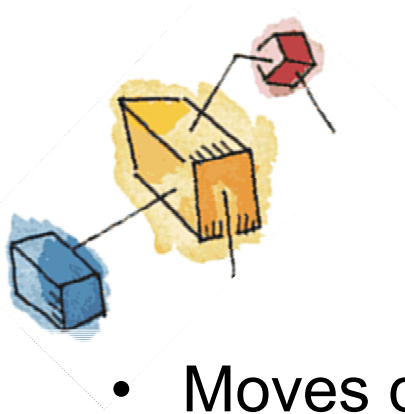


Main Memory

- Referred to as *primary memory* or simply [RAM](#)
- Consists of a set of locations defined by sequentially numbered **addresses**
- Stores data and programs
- Volatile
 - Contents of the memory is lost when the computer is shut down

A d d r e s s e s	0xFFFFFFFF	1000 0000
		
		
	0x00000008	0100 1001
	0x00000007	1100 1100
	0x00000006	0110 1110
	0x00000005	0110 1110
	0x00000004	0000 0000
	0x00000003	0110 1011
	0x00000002	0101 0001
	0x00000001	1100 1001
	0x00000000	0100 1111
Main Memory		

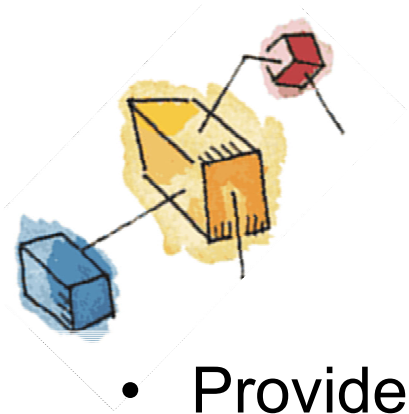




I/O Modules

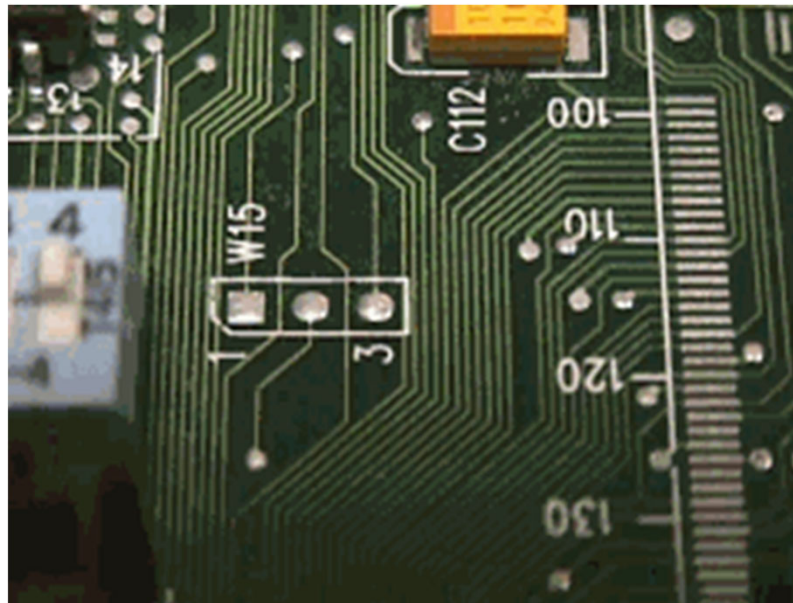
- Moves data between the computer and the external environment such as:
 - Storage / secondary memory devices (e.g. hard disks)
 - Communications equipment
 - Terminals





System Bus

- Provides for communication among processors, main memory, and I/O modules



Top-Level View

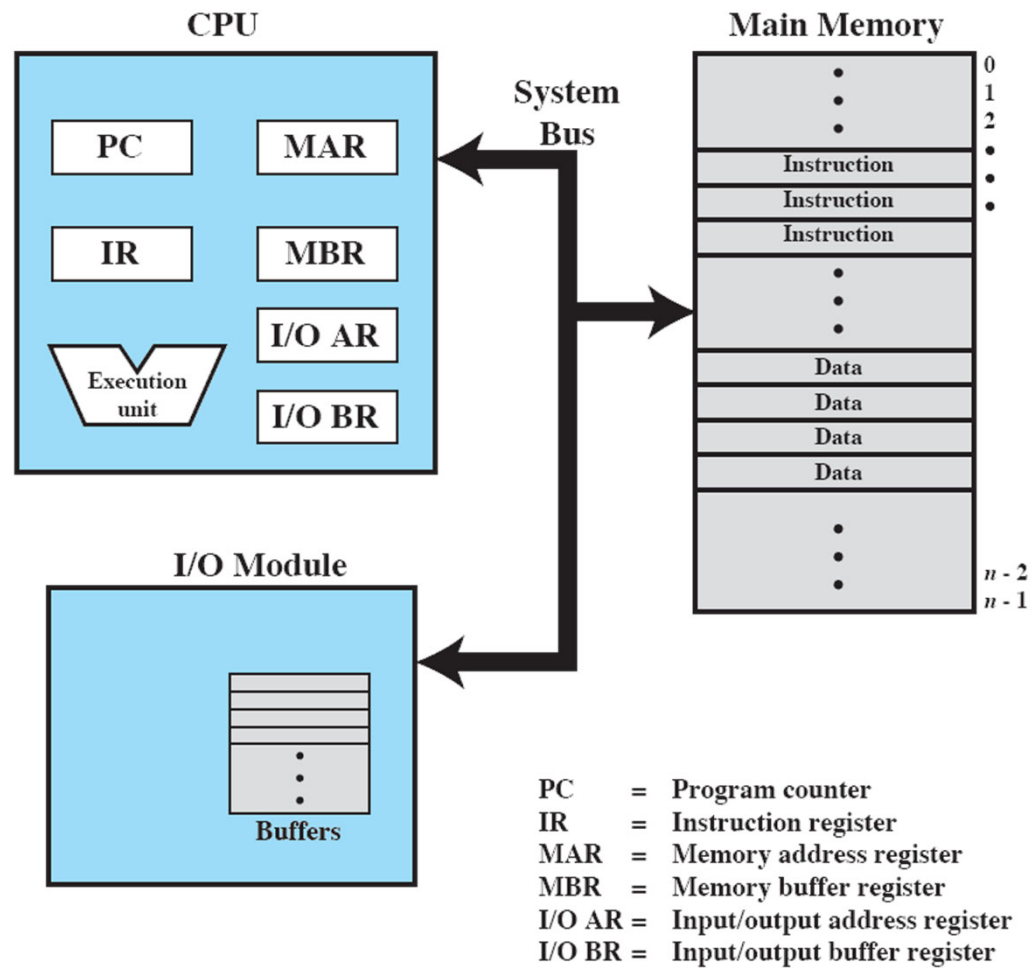
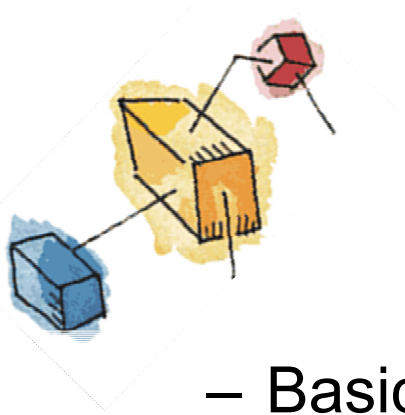


Figure 1.1 Computer Components: Top-Level View



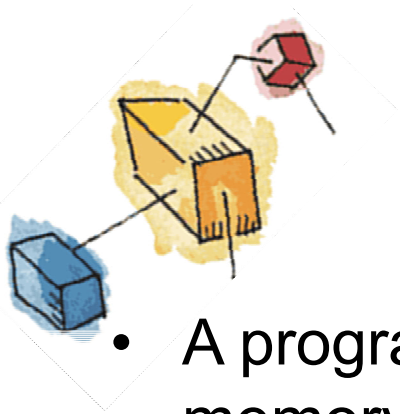
Roadmap

- Basic Elements

→ Instruction Execution

- Interrupts
- The Memory Hierarchy
- I/O Communication Techniques
- Operating System Objectives & Functions

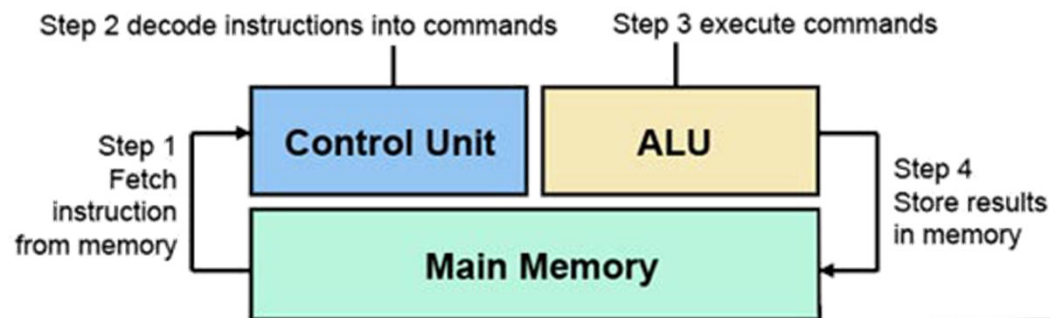




Instruction Execution

- A program consists of a set of instructions stored in memory
- 4 steps in a machine cycle
 - Processor reads (**fetches**) instruction from memory
 - Processor interprets (**decodes**) the current instruction
 - Processor **executes** the instruction
 - Processor **stores** the result back to memory

Machine Cycle



<http://www.computerhope.com>

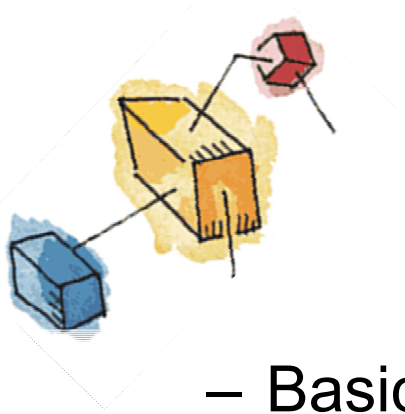




Processor Registers

- **Registers:** special memory locations **inside** the processor that can be accessed very fast
 - *Program Counter* (PC) holds address of the next instruction to be fetched
 - PC is incremented after each fetch
 - Fetched instruction is loaded into *Instruction Register* (IR)
 - Execution result stored in *Accumulator* (AC) temporarily
 - *Program Status Word* (PSW) contains execution status information





Roadmap

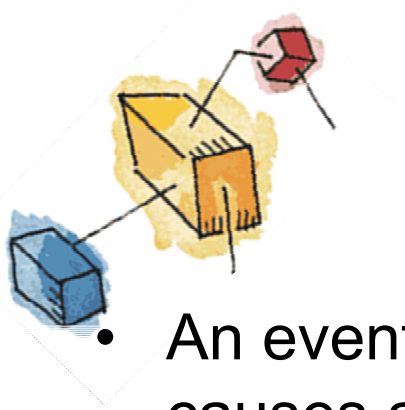
- Basic Elements
- Instruction Execution

→ Interrupts

- The Memory Hierarchy
- I/O Communication Techniques
- Operating System Objectives & Functions

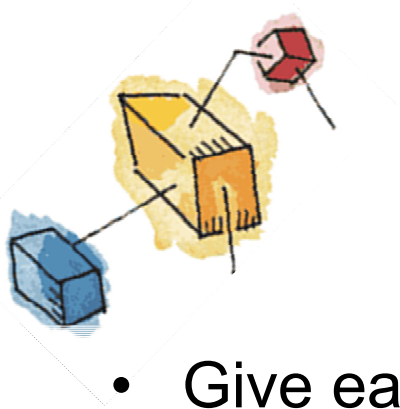


Interrupts



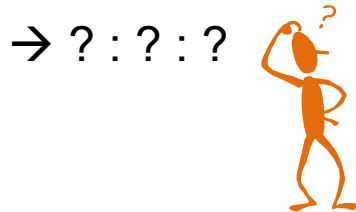
- An event external to the currently executing process that causes a change in (interrupts) the normal flow of instruction execution;
 - Usually generated by hardware devices external to the CPU
 - Typically indicate that some device needs service
- Why interrupts?
 - A computer is much more than the CPU
 - Keyboard, mouse, screen, disk drives, printer, sound card, camera, etc.
 - These devices occasionally need CPU service
 - But we can't predict when
 - We want to keep the CPU busy between events
 - Need a way for CPU to find out that devices need attention

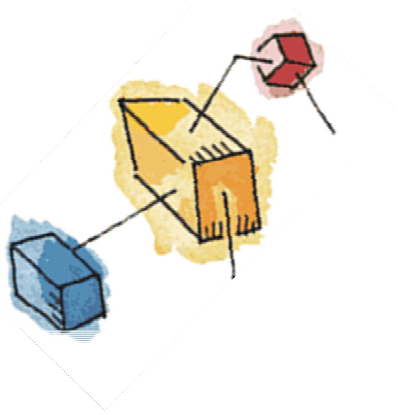




Interrupts

- Give each device a wire (interrupt line) that it can use to signal the processor
- When interrupt signaled, processor executes a routine called an *interrupt handler* to deal with the interrupt
- Provided to improve processor utilization
 - Most I/O devices are slower than the processor
 - Processor pauses to wait for device → 🖱️ wasteful use of the processor
 - Example: a 1GHz CPU could run 2,000 MIPS (0.5ns per instruction) vs. fast RAM chip with an access time of 50ns vs. 7200rpm HDD takes 5ms for a half-track rotation





Transfer of Control via Interrupts

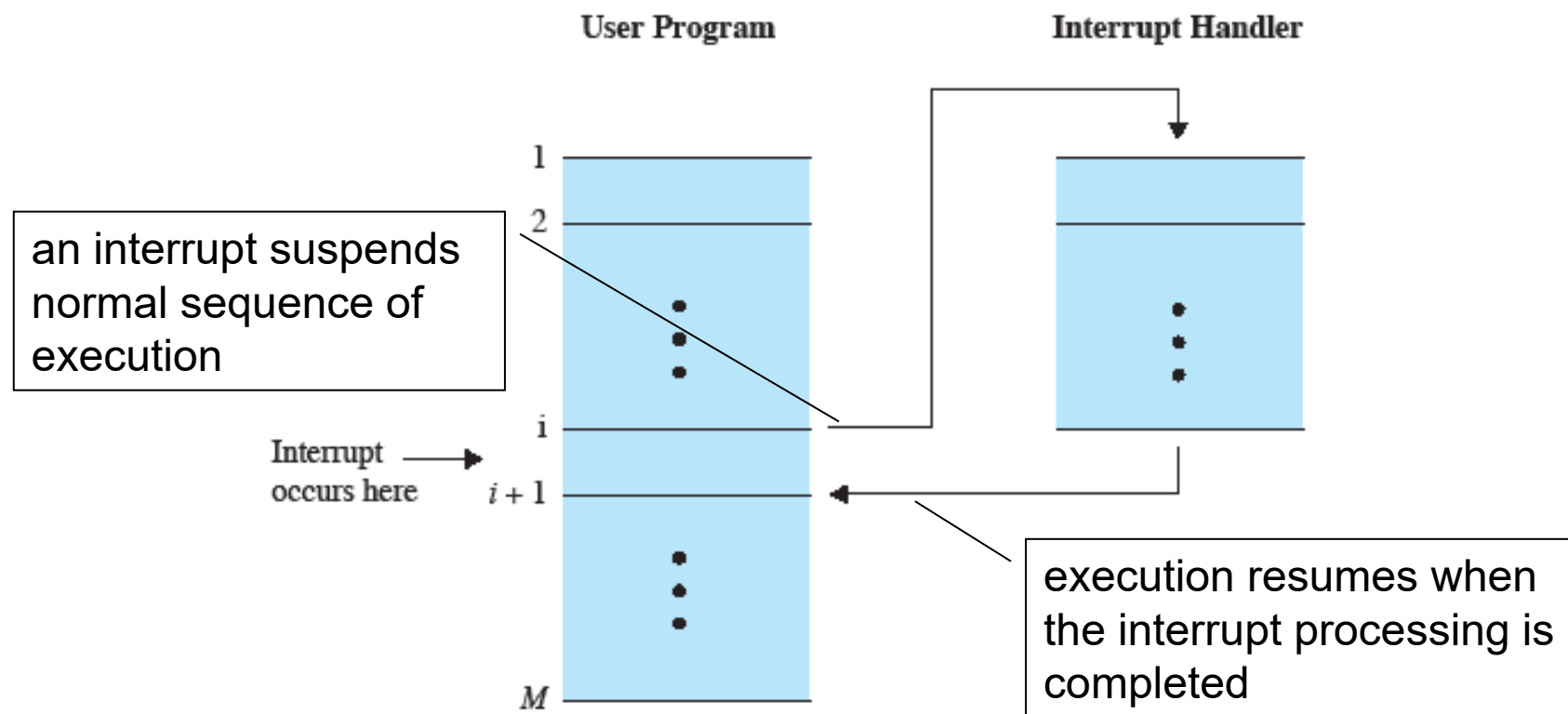
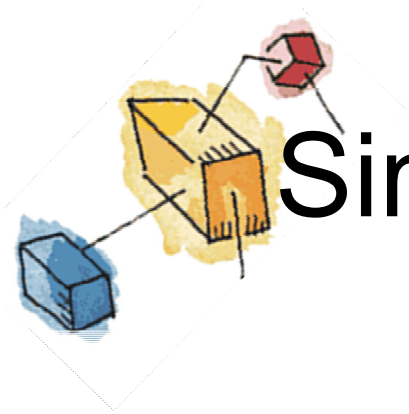


Figure 1.6 Transfer of Control via Interrupts





Simple Interrupt Processing

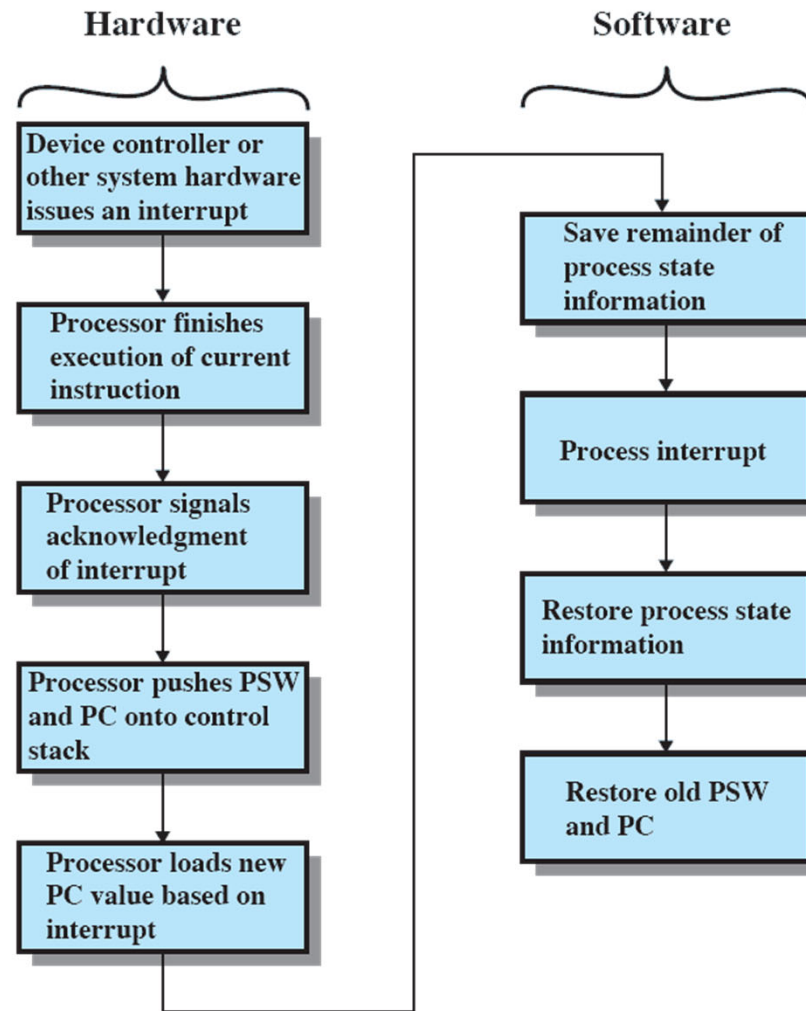
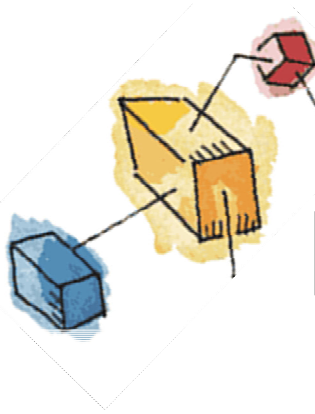
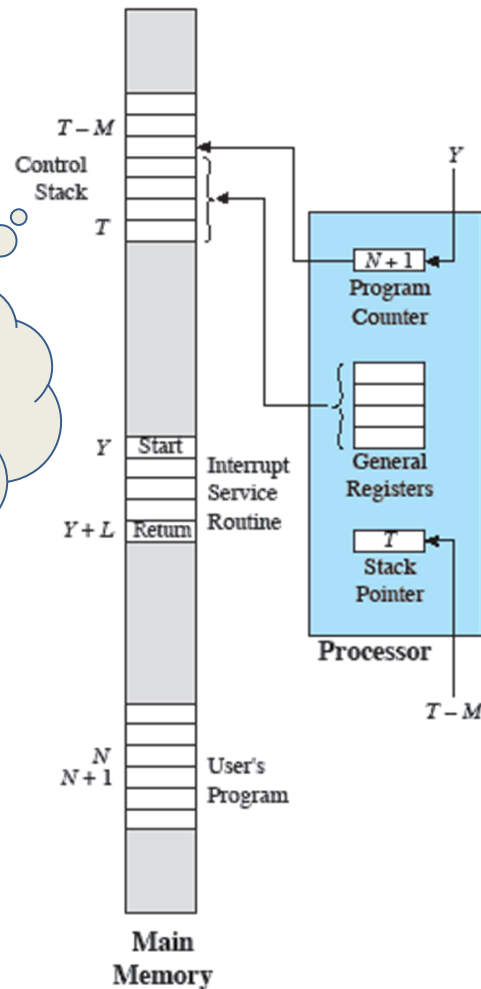


Figure 1.10 Simple Interrupt Processing

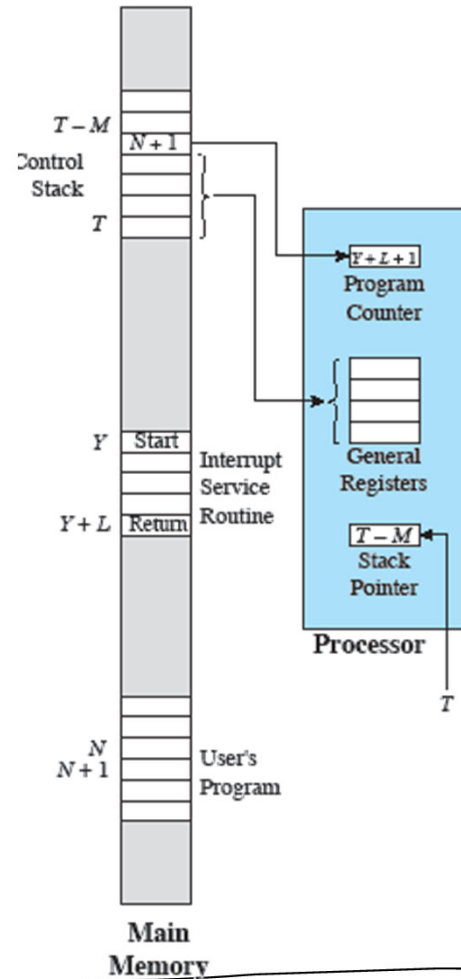


Changes in Memory and Registers for an Interrupt

stack: a certain memory area allocated as scratch space for a program to run

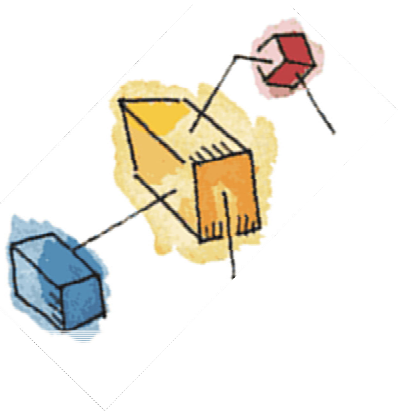


(a) Interrupt occurs after instruction at location N



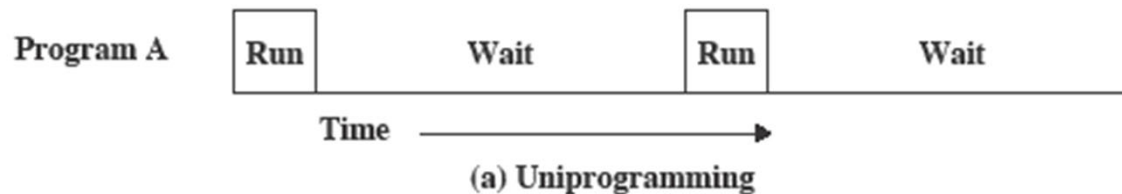
(b) Return from interrupt

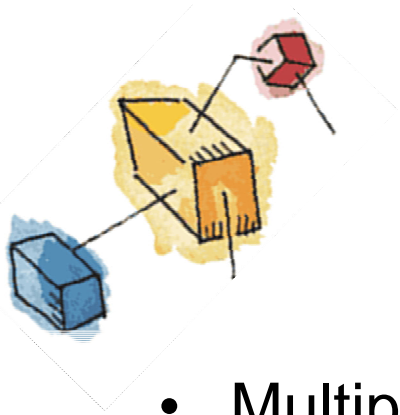




Uniprogramming vs Multiprogramming

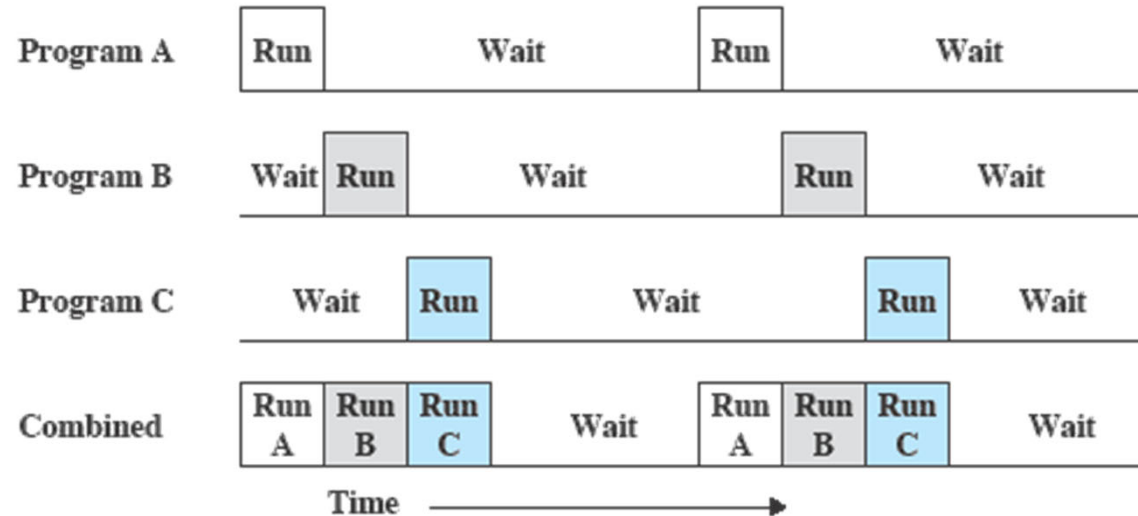
- Uniprogramming: only one program is running at a given time.
 - The processor spends a certain amount of time executing, until it reaches an I/O instruction; it must then wait for the I/O instruction to complete before proceeding





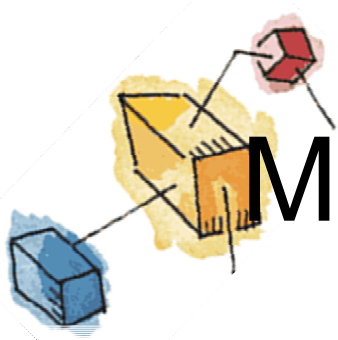
Uniprogramming vs Multiprogramming

- Multiprogramming (also known as multitasking): processor has more than one program to execute.
 - When one job needs to wait for I/O, the processor can switch to the other job.



(c) Multiprogramming with three programs





Multiprogramming Example

- Consider a computer with 250 Mbytes of available memory, a disk, a terminal, and a printer.
- Three programs, JOB1, JOB2 and JOB3 are submitted for execution at the same time.
- Assume minimal processor requirements for JOB2 and JOB3 and continuous disk and printer use by JOB3.

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes





Multiprogramming Example

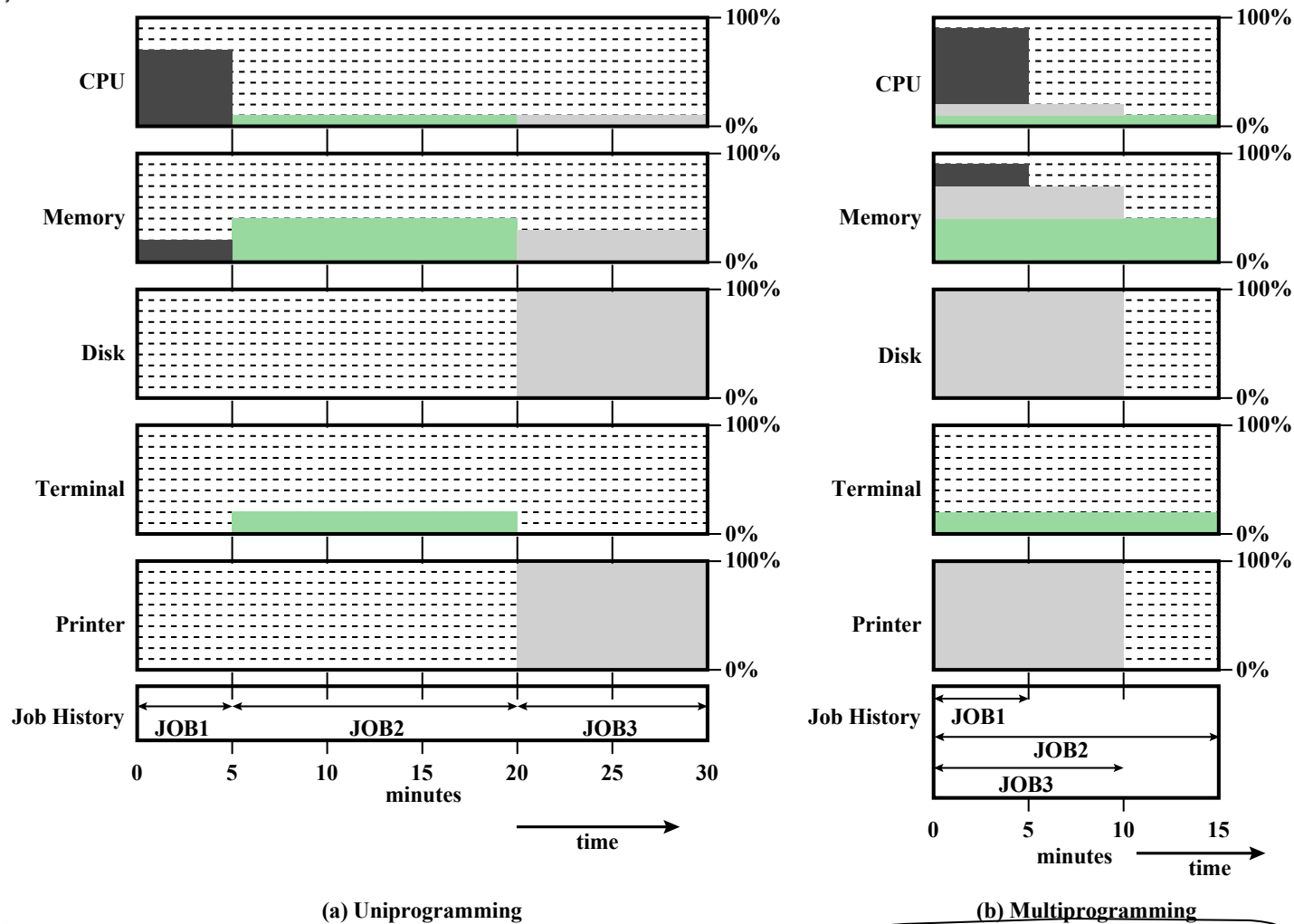
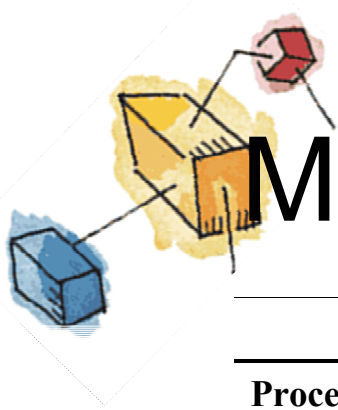


Figure 2.6 Utilization Histograms



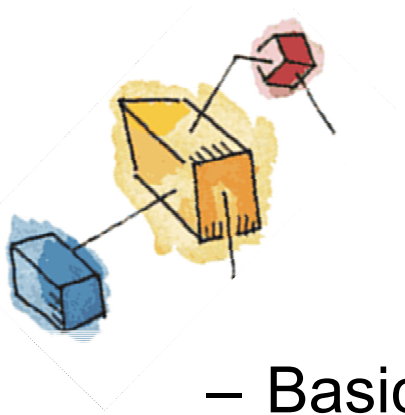


Multiprogramming Example

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

- With uniprogramming, there is gross underutilization of all resources.
- With multiprogramming, the improvement of average resource utilization, throughput and response times is evident.





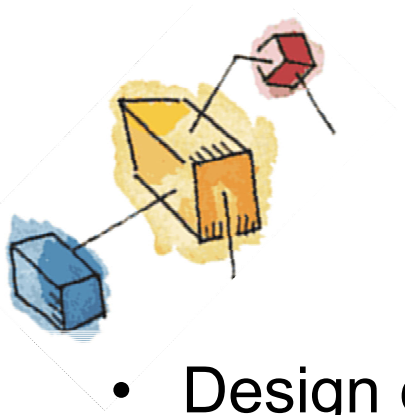
Roadmap

- Basic Elements
- Instruction Execution
- Interrupts

→ The Memory Hierarchy

- I/O Communication Techniques
- Operating System Objectives & Functions

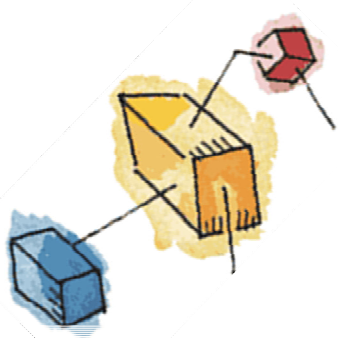




Memory Hierarchy

- Design constraints on a computer's memory
 - How much?
 - How fast?
 - How expensive?
- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access speed





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

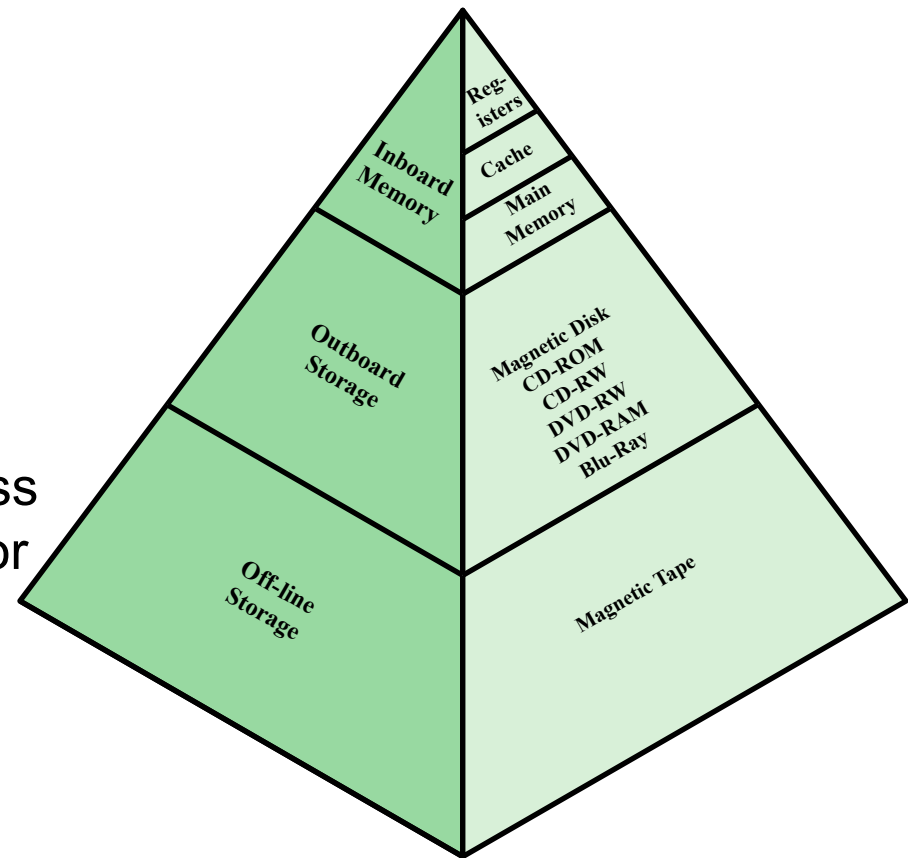
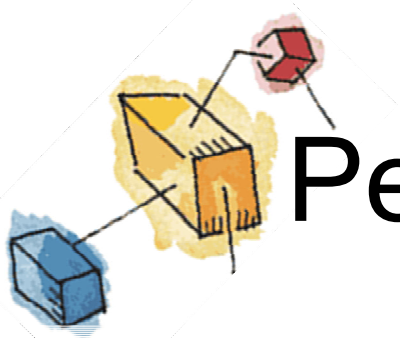


Figure 1.14 The Memory Hierarchy



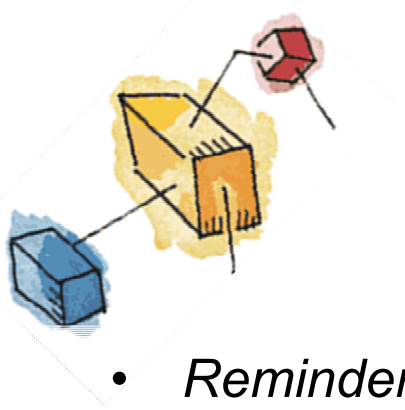


Performance Comparison

- To compare various levels

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape





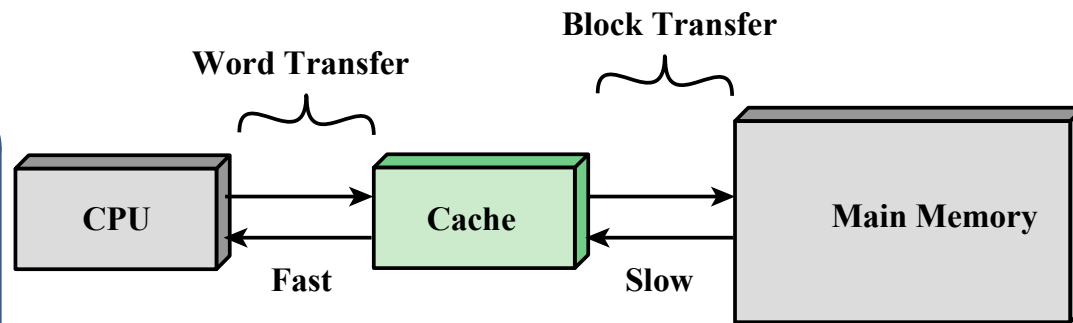
Cache Memory

- *Reminder:* processor must access memory at least once per instruction cycle → processor execution is limited by memory cycle time, but processor speed is much faster than memory access speed!
- Solution: copy information in use from slower to faster (but smaller) storage (cache) temporarily.
- Cache is checked first to determine if information is there.
 - If it is, information used directly from the cache
 - If not, data copied to cache and used there

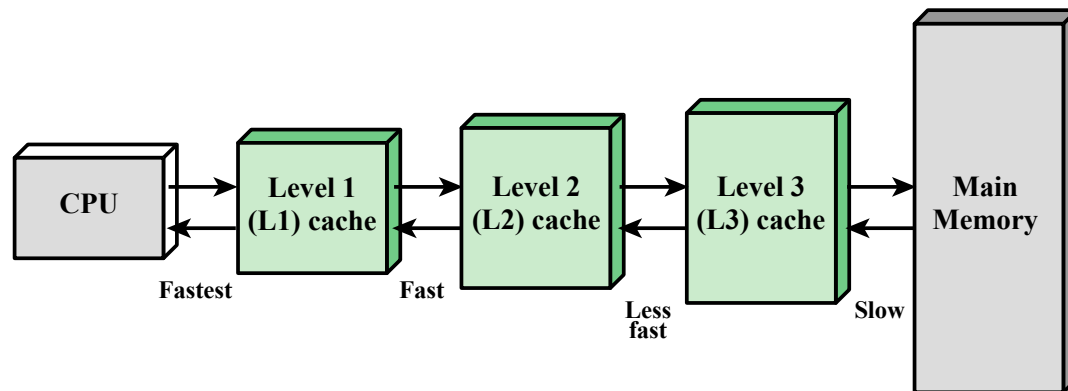


Cache Memory

Principle of locality: Data which is required soon is often close to the current data. If data is accessed, then it's neighbors might also be accessed in the near future.

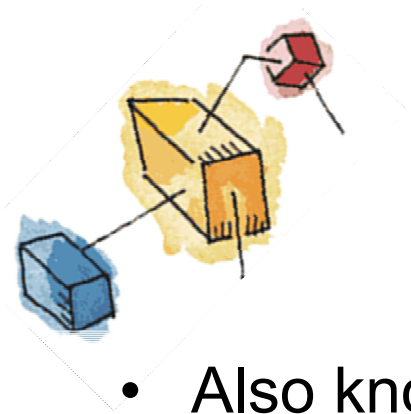


(a) Single cache



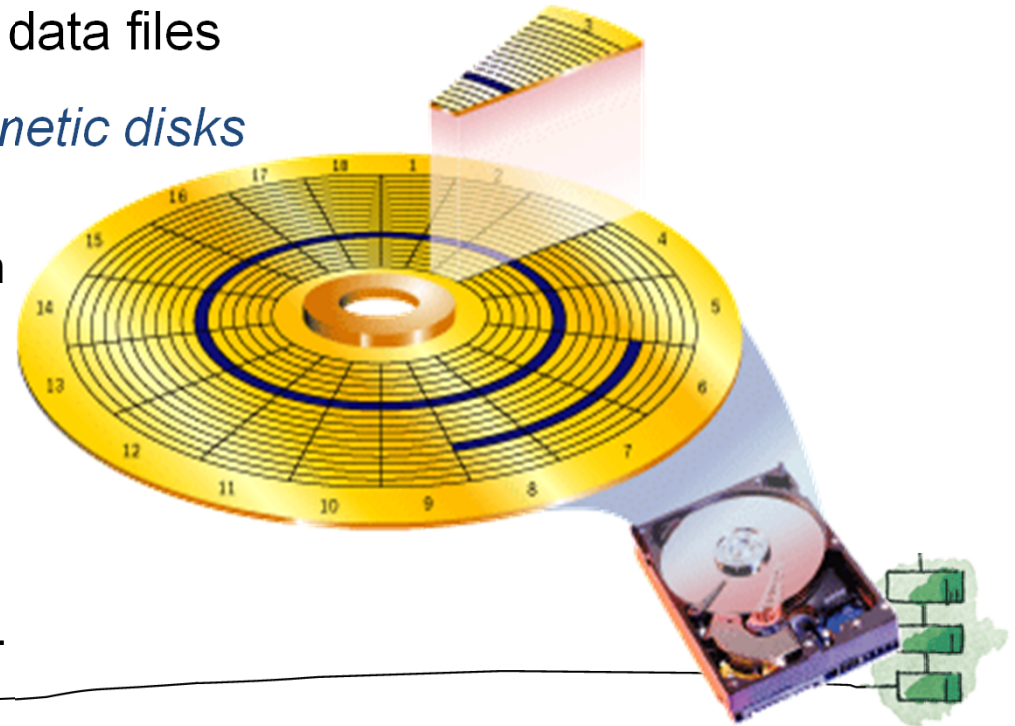
(b) Three-level cache organization

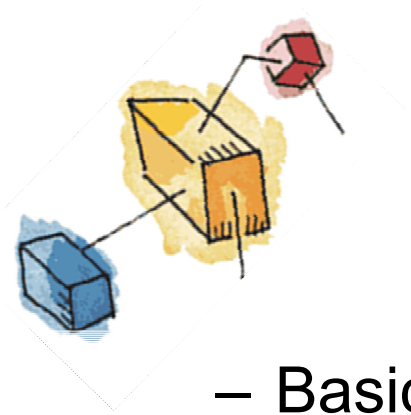
Figure 1.16 Cache and Main Memory



Secondary Memory

- Also known as storage devices
 - Extension of main memory that provides large nonvolatile storage capacity
 - Used to store program and data files
 - Most commonly used: *magnetic disks*
 - 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.





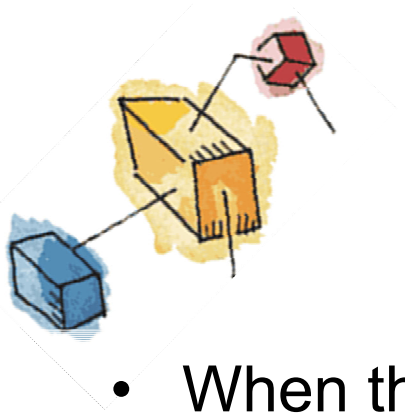
Roadmap

- Basic Elements
- Instruction Execution
- Interrupts
- The Memory Hierarchy

➔ I/O Communication Techniques

- Operating System Objectives & Functions

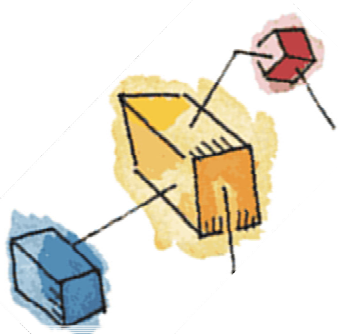




I/O Techniques

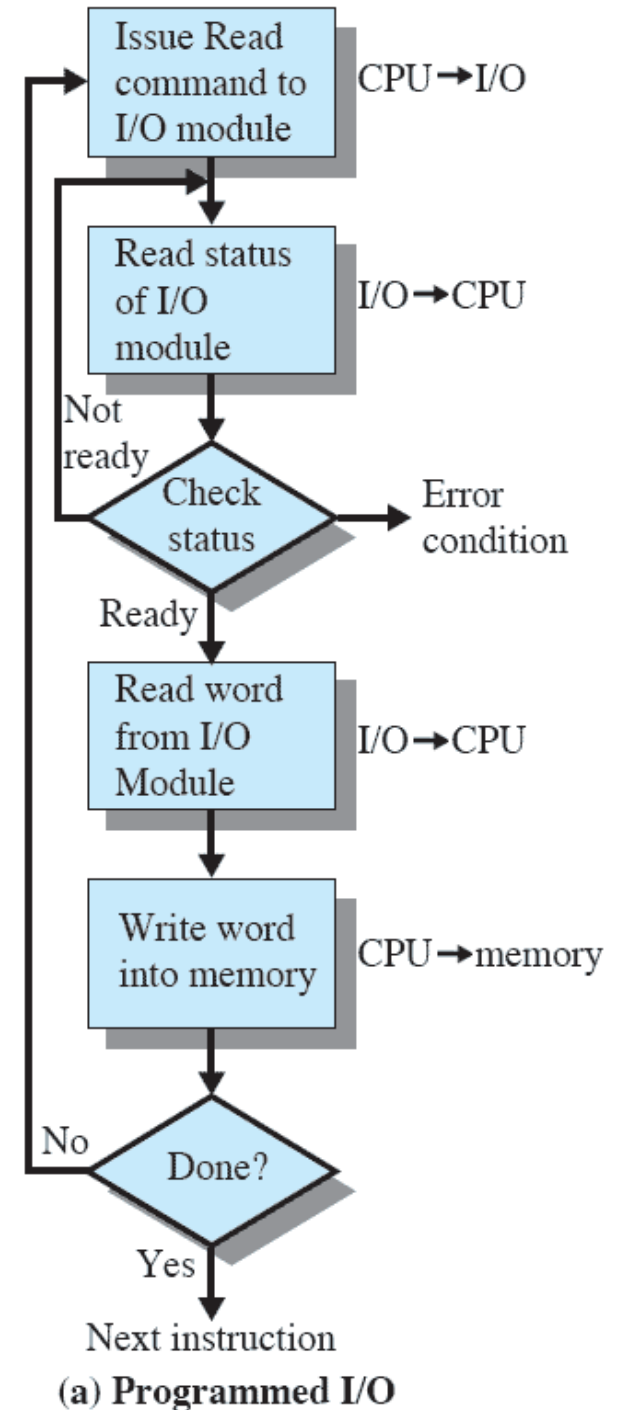
- When the processor encounters an instruction relating to I/O,
 - it executes that instruction by issuing a command to the appropriate I/O module.
- Three techniques are possible for I/O operations:
 - Programmed I/O
 - Interrupt-driven I/O
 - Direct memory access (DMA)

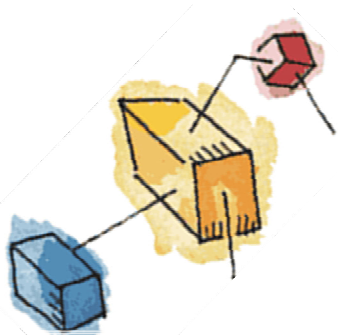




Programmed I/O

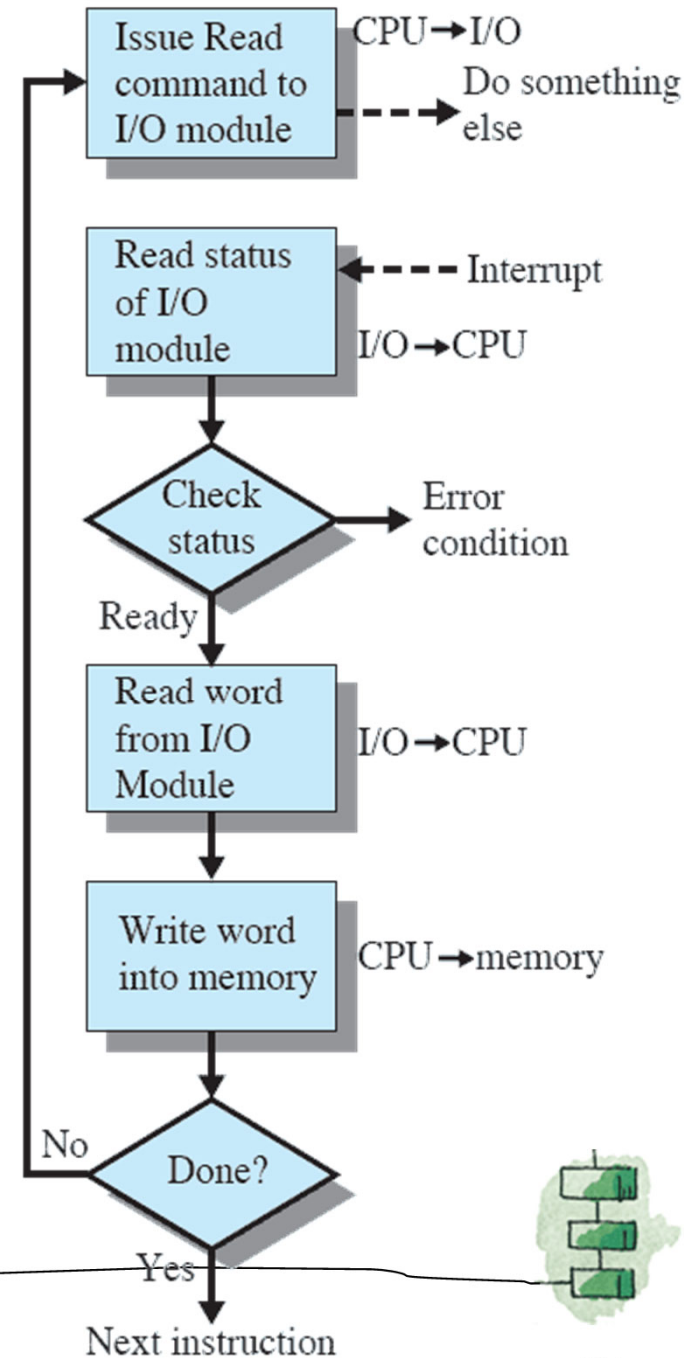
- CPU issues an I/O command to the I/O module (e.g., a read command to read in a word of data from an external device into memory).
- I/O module performs the requested action
 - then sets the appropriate bits in the I/O status register (but takes no further action to alert CPU).
- CPU periodically checks the status of I/O module until it determines the instruction is completed.
- 🗑️ The performance is poor.



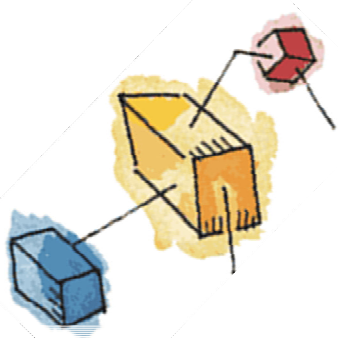


Interrupt-Driven I/O

- CPU issues an I/O command to the I/O module and then goes on to do some other useful work.
- I/O module will then interrupt CPU to request service when it is ready to exchange data with CPU.
- CPU executes the data transfer and then resumes its former processing.
- 👍 More efficient than programmed I/O (no needless waiting)
- 🗣 Requires active intervention of CPU to perform data transfer.

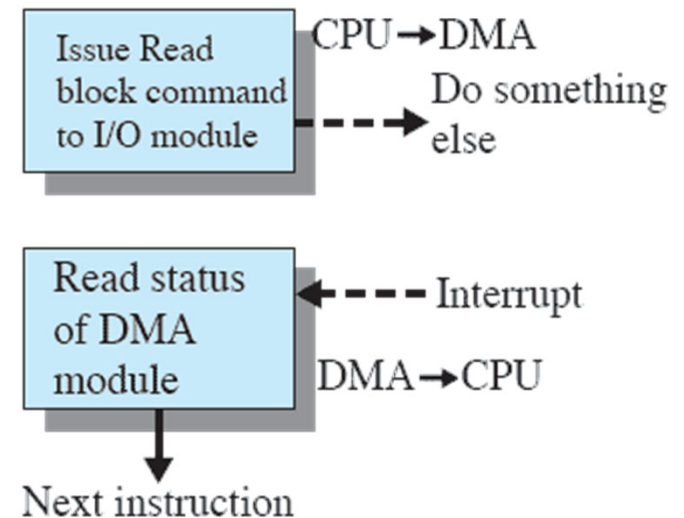


(b) Interrupt-driven I/O



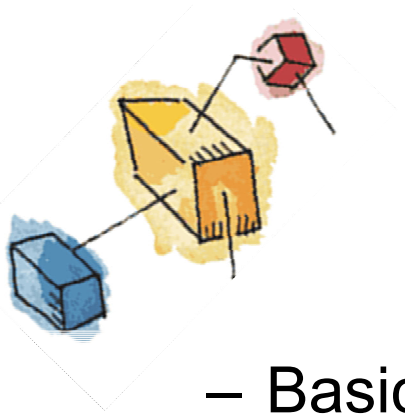
Direct Memory Access

- Performed by a separate module.
- When CPU wishes to read/write, it issues a command to DMA module containing:
 - Whether a read or write is requested
 - The address of the I/O device involved
 - The starting location in memory to read/write
 - The number of words to be read/written
- DMA transfers the entire block of data directly to and from memory without going through CPU, sends interrupt when finished.
- 👍 More efficient than interrupt-driven or programmed I/O.



(c) Direct memory access



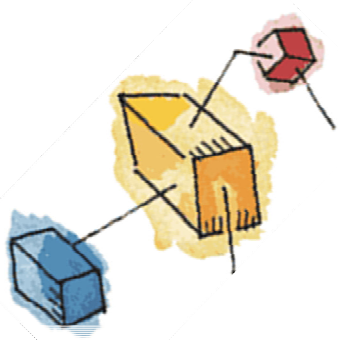


Roadmap

- Basic Elements
- Instruction Execution
- Interrupts
- The Memory Hierarchy
- I/O Communication Techniques

➔ Operating System Objectives & Functions

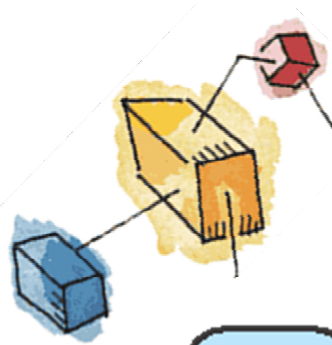




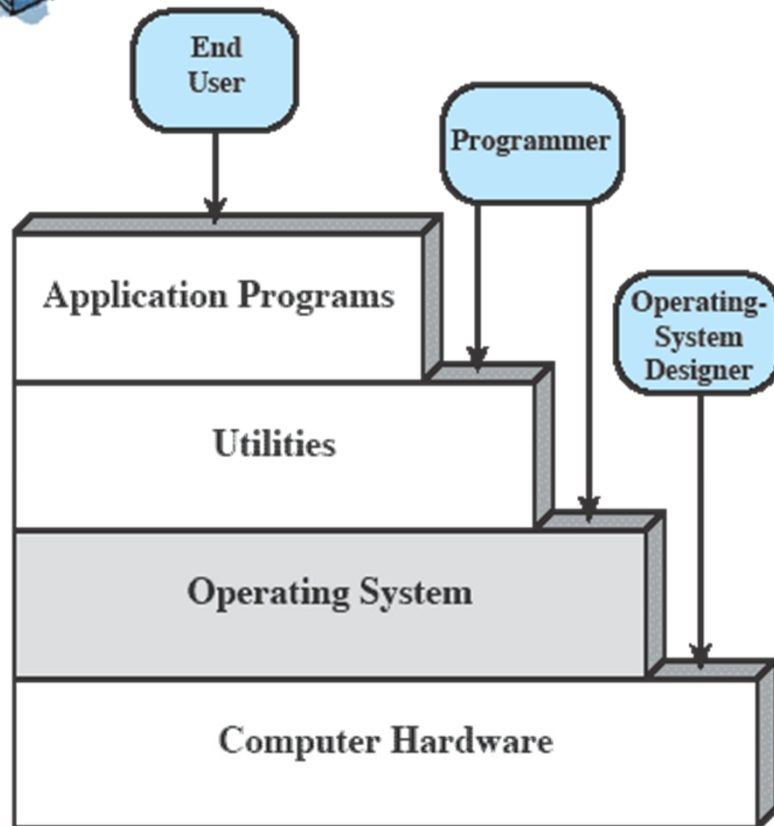
Operating System (OS)

- A program that controls the execution of application programs.
- An interface between applications and hardware.
- Main objectives of an OS:
 - Convenience
 - Making a computer more convenient to use
 - Efficiency
 - Allowing computer resources to be used efficiently
 - Ability to evolve
 - Permitting effective development, testing and introduction of new system functions without interfering with service.
 - hardware upgrade / new hardware
 - new services
 - fixes





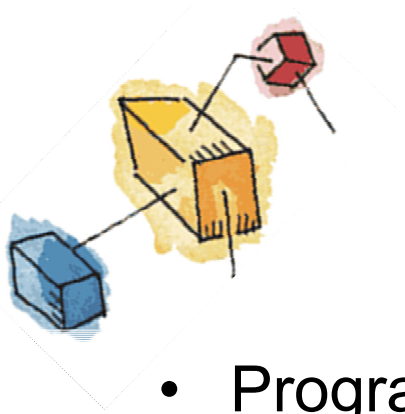
OS as a User/Computer Interface



- End user
 - views a computer system in terms of a set of applications
 - is not concerned with the details of computer hardware.
- Programmer
 - makes use of a set of system programs (utilities) in developing an application
 - a running application invokes the utilities to perform certain functions.
- OS
 - the most important collection of system programs (frequently used functions)
 - masks the details of the hardware from the programmer and provides the programmer with a convenient interface for using the system.

Figure 2.1 Layers and Views of a Computer System

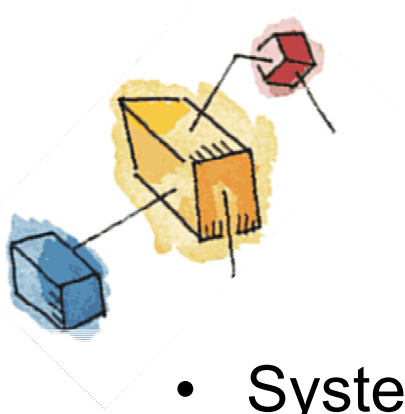




OS Services

- Program development
 - OS provides a variety of facilities and services, such as editors and debuggers, to assist the programmer in creating programs.
- Program execution
 - OS handles steps need to be performed to execute a program.
- Access I/O devices
 - OS provides a uniform interface so that programmers can access I/O devices using simple reads and writes.
- Controlled access to files
 - In the case of a system with multiple users, OS provides protection mechanisms to control access to the files.





OS Services

- System access
 - For shared or public systems, OS controls access to the system as a whole and to specific system resources in order to provide protection of resources and data from unauthorized users and resolve conflicts for resource contention.
- Error detection and response
 - A variety of errors can occur while a computer system is running. OS must provide a response that clears the error condition with the least impact on running applications.
- Accounting
 - A good OS will collect usage statistics for various resources and monitor performance parameters such as response time.





OS as Resource Manager

- A computer is a set of resources for the movement, storage, and processing of data.

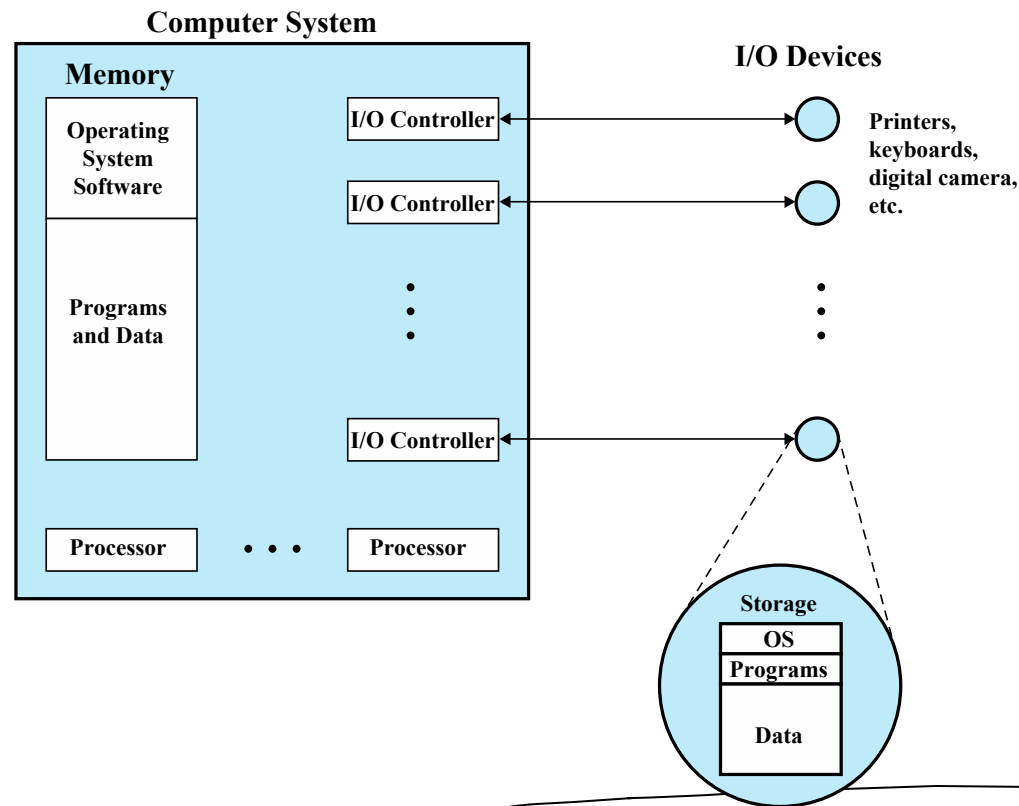
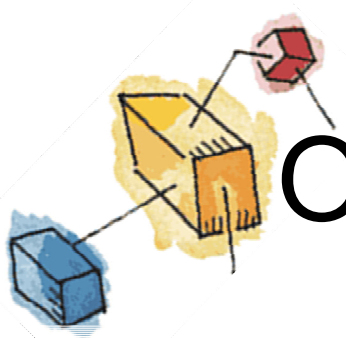


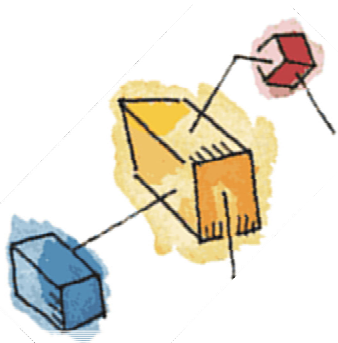
Figure 2.2 The Operating System as Resource Manager



OS as Resource Manager

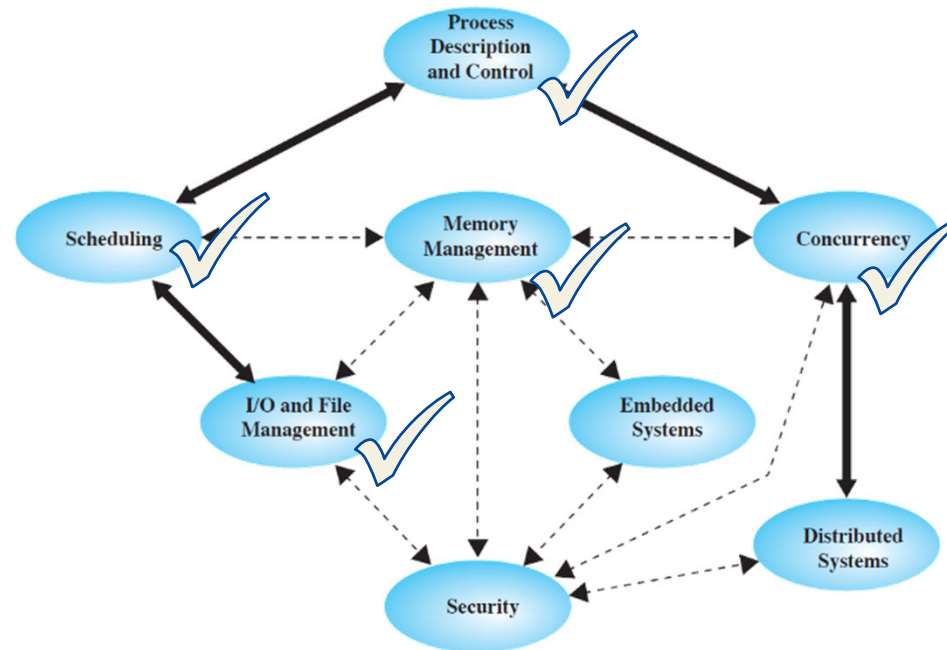
- The OS is responsible for managing these resources.
 - determines how much processor time is to be devoted to the execution of a particular user program
 - controls the allocation of main memory
 - decides when an I/O device can be used by a program in execution
 - controls access to and use of files.



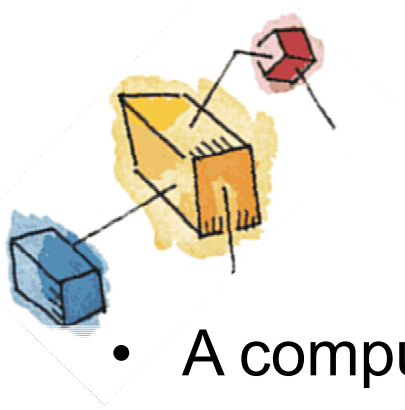


Major Topics of an OS

- Operating Systems are among the most complex pieces of software ever developed.
- Major topics to be covered in this course include:
 - Processes
 - Concurrency
 - Scheduling
 - Memory
 - Input/Output and Files



Summary



- A computer platform consists of a collection of hardware resources.
- Computer applications are developed to perform some tasks.
- It is inefficient for applications to be written directly for a given hardware platform.
- The OS was developed to provide a convenient, feature-rich, secure, and consistent interface for applications to use.
- We can think of the OS as providing a uniform, abstract representation of resources that can be requested and accessed by applications.

