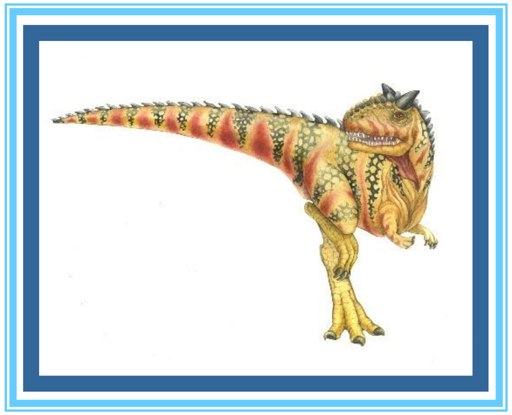
Operating Systems



Chapter 1
Introduction

Course Information

- Subject: Operating Systems
- Code:BSE-2405
- Credit Hours:3+1

Contact Information

Instructor: Engr. Shamila Nasreen

Qualification: PHD

Assistant Professor

Department of Software Engineering

MUST

Email: shamila.se@must.edu.pk

Office hours: Monday: 09:00AM-10:00 AM

Thursday: 8:30-10:00 AM

Grading Criteria

- **Total Marks(theory): 150(75%)**
  - Quizzes: 7.5%
  - Assignment:7.5%
  - Mid Term(25%) 45 marks
  - Terminal Exam:50%(75 marks)

Two quizzes:

- Before mid term:01
- After Mid Term:01

Assignment:1
- Before Mid term

Presentation: 1
- after Mid Term

Books

Text Book:

1. Avi Silberschatz, Peter Baer Galvin, Greg Gagne, **Operating System Concepts**, **9th Edition**, John wiley & Sons, Inc. ISBN 978-1-118-06333-0, 2012

Reference Book:

1. William Stallings, **Operating Systems: Internals and Design Principles**, **8th edition** Pearson Education Limited, 2014 ISBN: 1292061944, 9781292061948

2. Harvey M. Deitel, “Operating Systems Concepts”

Course Objectives

- To learn the fundamentals of Operating Systems.
- To learn the mechanisms of OS to handle processes and threads and their communication
- To learn the mechanisms involved in memory management in contemporary OS To gain knowledge on distributed operating system concepts that includes architecture,
- Mutual exclusion algorithms, deadlock detection algorithms and agreement protocols
- To know the components and management aspects of concurrency management
- To learn programmatically to implement simple OS mechanisms

Learning Outcomes

1. **Understand** the characteristics of different structures of the Operating Systems and identify the core functions of the Operating Systems.
2. **Analyze** and evaluate the algorithms of the core functions of the Operating Systems and explain the major performance issues with regard to the core functions.
3. **Demonstrate** the knowledge in applying system software and tools available in modern operating systems.

Student Alert

- Attendance is mandatory. Every class is important.
- All deadlines are hard. Under normal circumstances late work will not be accepted.
- 80% attendance is mandatory. Latecomers will be marked as absent.
- Students are required to take all the tests. No make-up tests will be given under normal circumstances.
- Any form of cheating on exams/assignments/quizzes is subject to serious penalty

Course Contents

Week 1	**What does an operating system do? Operating systems structure, Service provided by OS, Multiprocessor systems, Different computing environments, Evolution of operating systems, Hand-held systems and real-time operating systems.**
Week 2	Computer system operation, I/O handling, Storage structure and hierarchy, Coherency and consistency, Dual mode operation, I/O protection, Memory protection, Protection and Security ,
Week 3	Operating system structure, and Services, System calls, types of system calls, Operating system design and implementation, Different structures including MS DOS, UNIX, layered , Microkernel etc.
Week 4	Process concepts, Process structure in memory, process states and state models, PCB, Context switching, Process scheduling, Scheduler and its types, Various queues.

Course Contents

Week 5	**Operations on Processes, inter process communication, independent and cooperating processes, direct and indirect communication, buffering etc. Remote procedure calls**
Week 6	Multithreading processes and their models, Types of threads, Threading issues, Pthreads, Solaris 2 threads, Windows 2000 threads, Linux threads
Week 7	CPU Scheduling, scheduling criteria, preemption and non-preemption, Optimization criteria, FCFS Scheduling Algorithm, Shortest Job First Scheduling Algorithm, Shortest remaining First, Priority Scheduling
Week 8	Round Robin algorithm, HRRN, Multilevel queue and multilevel feedback queue algorithm
Week 9	Necessary conditions for deadlocks, Resource-allocation graph, Methods for handling deadlocks, Deadlock prevention
Week 10	Deadlock avoidance, Banker's and Safety Algorithms, Deadlock detection, Recovery methods and combined approach.

Course Contents

Week 11	**Multi-step processing of a user program, Loading and linking, Memory mapping, Swapping, Contiguous allocation methods, Fragmentation**
Week 12	Multithreading processes and their models, Types of threads, Threading issues, Pthreads, Solaris 2 threads, Windows 2000 threads, Linux threads
Week 13	Paging operation and structure, Paging method & hardware support, Segmentation, Page tables, Hardware implementation and segmentation with paging.
Week 14	Virtual Memory, sharing libraries using virtual memory, Demand paging and its performance, Copy on Write, Page replacement algorithms, Allocation of frames, Thrashing and other considerations, Memory mapped files, Allocation Kernel Memory. , Directory structure, General graph directory, Protection and consistency, semantics
Week 15	Disk Scheduling, Disk scheduling parameters, disk scheduling algorithms.
Week 16	Presentation

Module 1: Introduction

- What is an operating system?
- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Kernel Data Structures
- Computing Environments
- Open-Source Operating Systems

Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems

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.

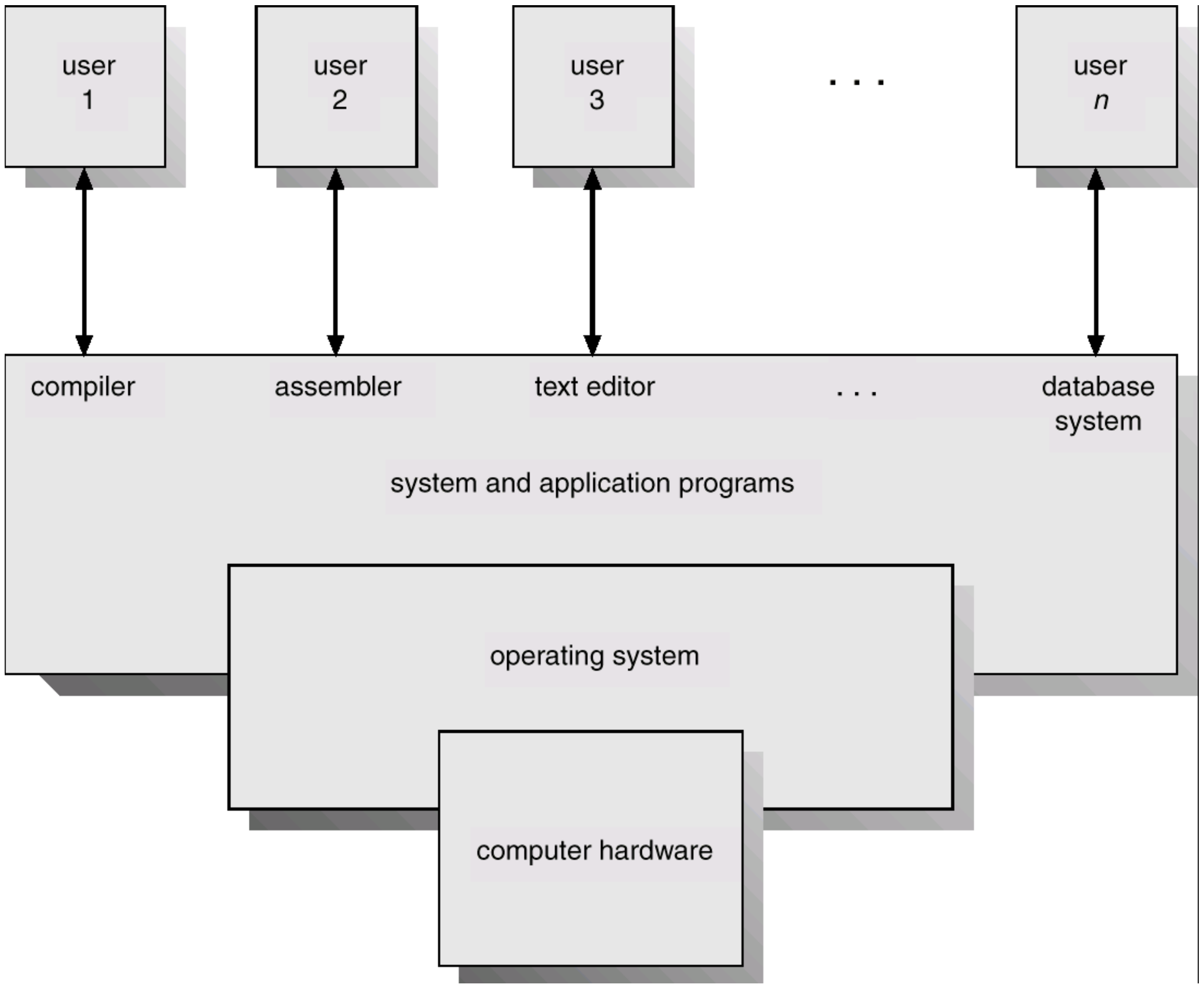
Operating System Goals

- Salient goals are:
  - execute user programs
  - solve user problems
  - ensure proper coordination and synchronization
  - provide convenience to the users
  - operation the computer system components efficiently
  - parallel and optimal use of computer resources
  - accommodate hardware upgrades and new services
  - resolve and fix problems and bugs

Computer System Components

1. Hardware – provides basic computing resources (CPU, memory, I/O devices).
2. Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.
3. Applications programs – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
4. Users (people, machines, other computers).

Abstract View of System Components



System View of OS

- **Resource Allocator**

A computer system has many resources that may be required to solve a problem

- CPU time
- Memory space
- File storage space
- I/O devices

acts as the manager of these resources to resolve conflicting requests for resources

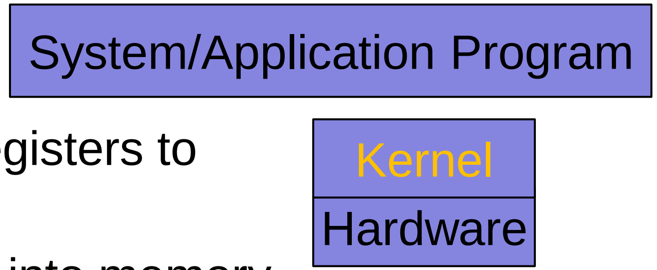
- **Control Program**
  - Executes user programs
  - Prevents errors and improper use
  - Responsible for the operation and control of I/O devices

Operating System Definition (Cont.)

- No universally accepted definition
- “Everything a vendor ships when you order an operating system” is a good approximation
  - But varies wildly
- “The one program running at all times on the computer” is the **kernel(OS)**.
- Everything else is either
  - a system program (ships with the operating system) , or
  - an application program.

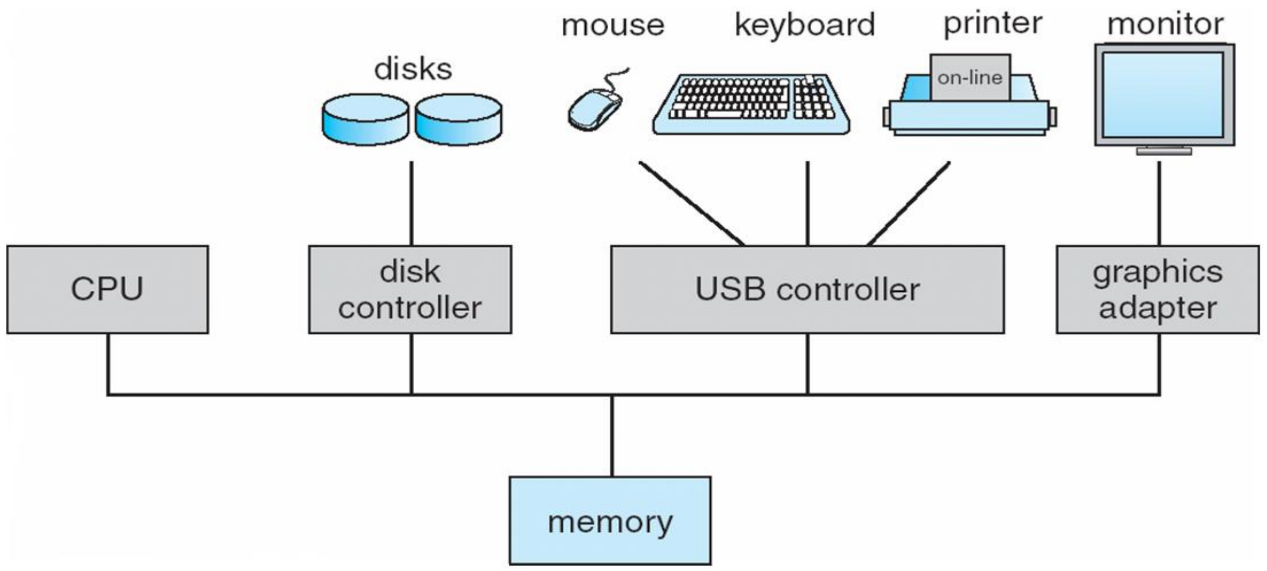
Computer Startup

- When a computer is powered up or rebooted-it needs to have an initial program to run: This initial program called **bootstrap program.**
- **it is stored in**
  - ▯ read-only memory (ROM)
  - ▯ electrically erasable programmable read-only memory (EEPROM)
  - ▯ known as **firmware**
  - – Initializes all aspects of system from CPU registers to device Controller to memory content
  - – the bootstrap program must locate and load into memory the operating system kernel.
  - – Load The operating system then starts executing the first process, such as "init," and waits for some event to occur.



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
  - To ensure orderly access to the shared memory, a memory controller is provided whose function is to synchronize access to the memory.
  - cpu moves data to/from memory to/from local buffers.
  - I/O is from device to device controller.

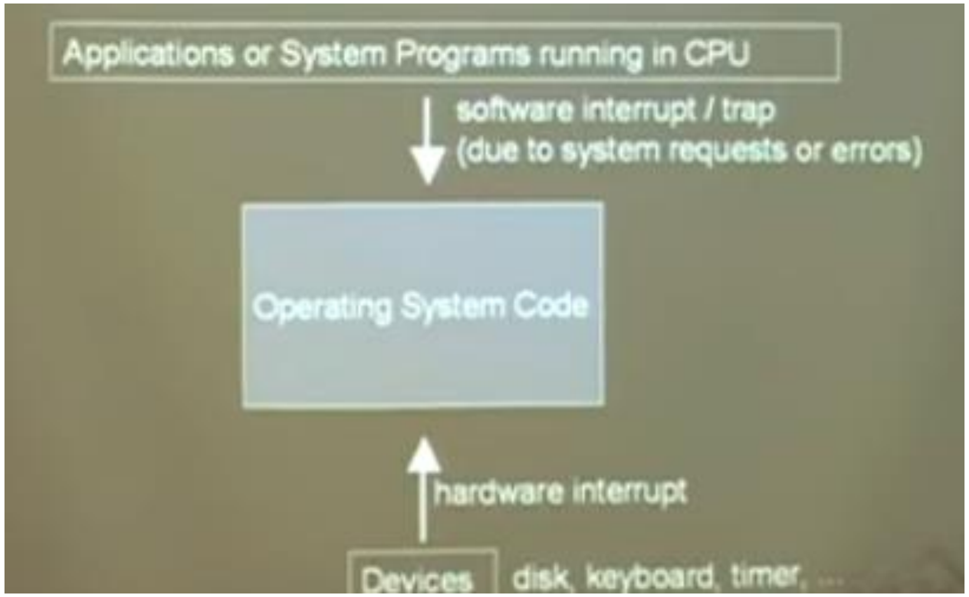


Computer-System Operation

- 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
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

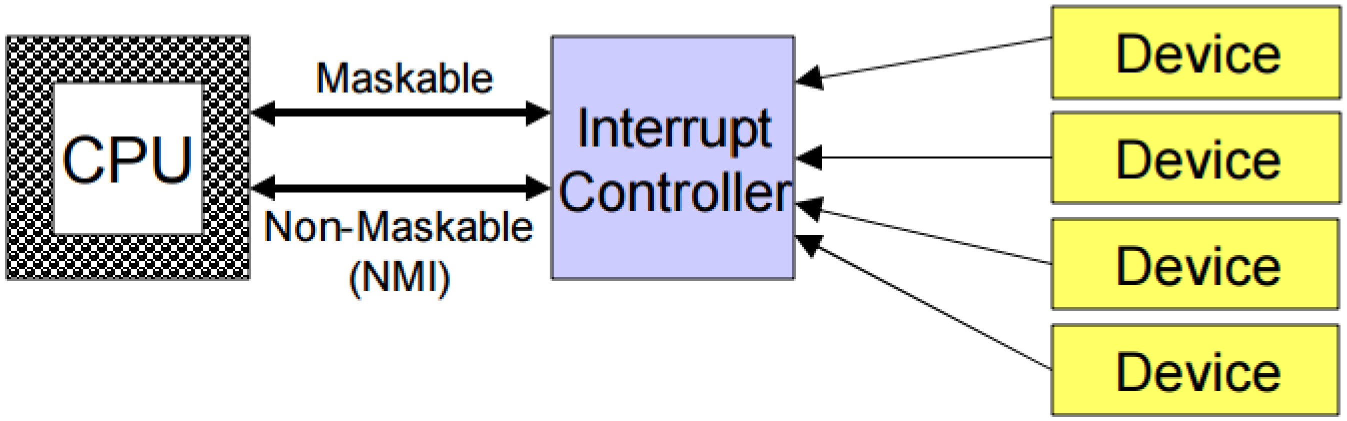

Applications or System Programs running in CPU
software interrupt / trap
(due to system requests or errors)
Operating System Code
hardware interrupt
Devices
disk, keyboard, timer, ...

Computer-System Operation

- 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
- **Possible Solution: Polling**
  - CPU periodically checks each device to see if it needs service °
  - takes CPU time even when no requests pending °
  - can be efficient if events arrive rapidly
  - "Polling is like picking up your phone every few seconds to see if you have a call. …"

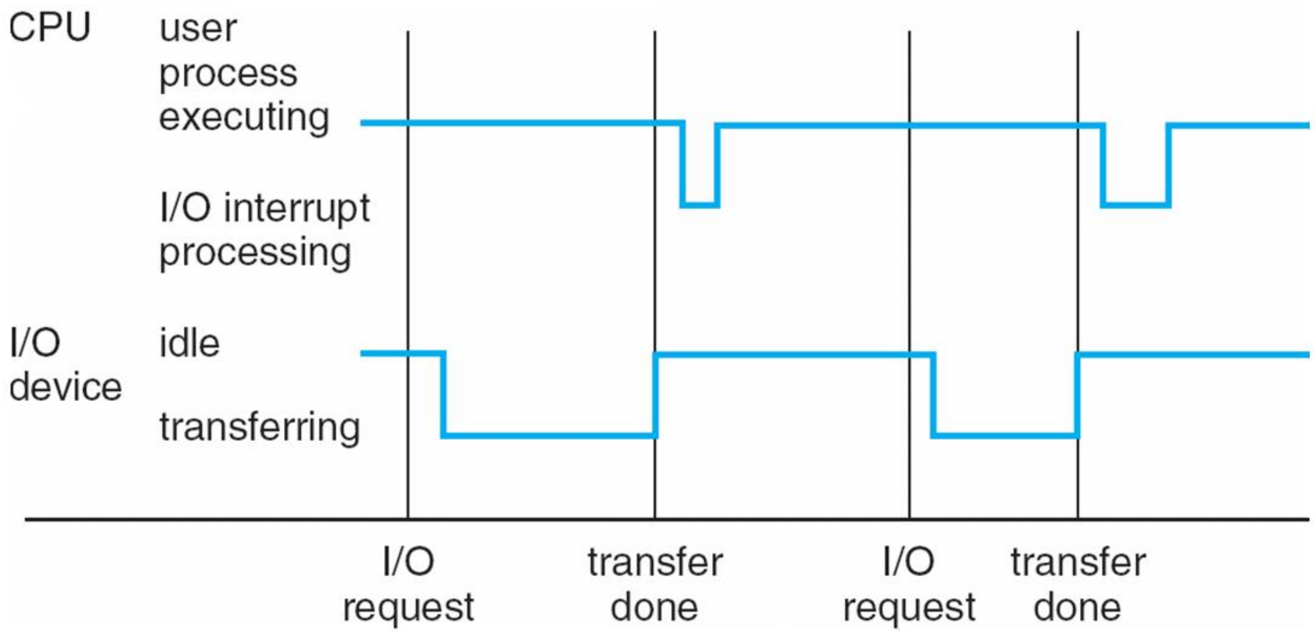
Computer-System Operation

- **Alternative: 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
- No overhead when no requests pending

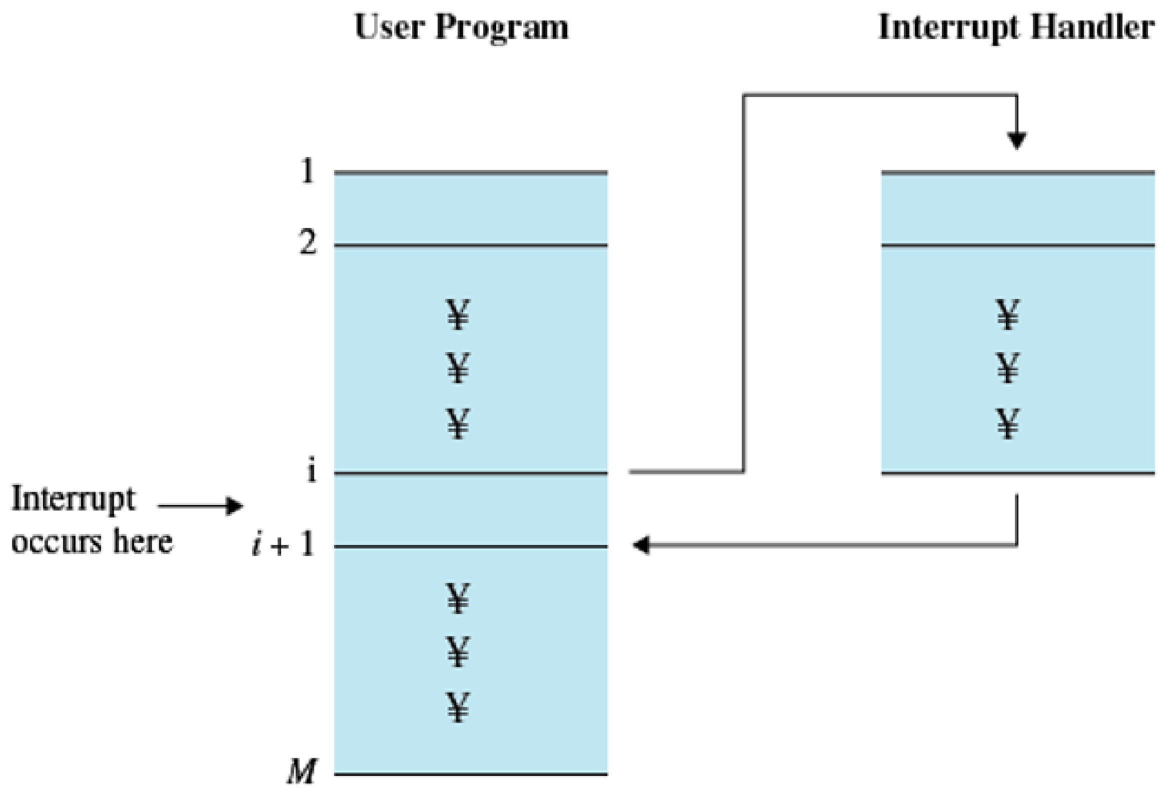


Interrupts are like waiting for the phone to ring.”

Interrupt Timeline



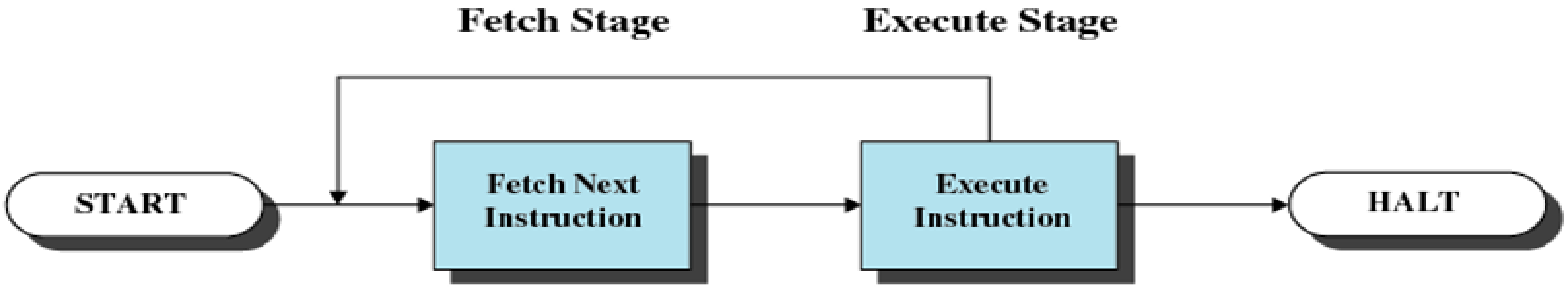
Computer-System Operation: Interrupt Handling



Transfer of Control Via interrupt

Instruction Execution

- A program to be executed by a processor consists of a set of instructions stored in memory.
- Instruction processing consists of two steps:
  - The processor reads (**fetches**) instructions from memory one at a time
  - **executes** each instruction.
- The processing required for a single instruction is called an **instruction cycle**



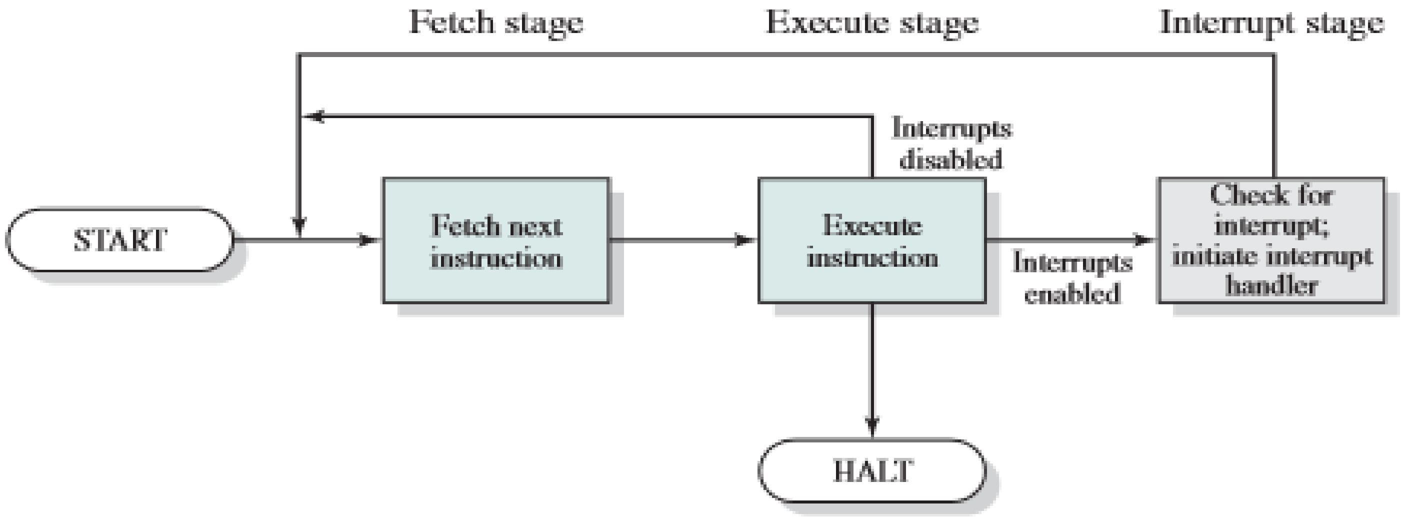
Instruction Fetch & Execute

- The processor fetches the instruction from memory.
- Program counter (PC) holds address of the instruction to be fetched next .
- Program counter is incremented after each fetch
- The fetched instruction is loaded into the instruction register (IR).
- The instruction contains bits that specify the action the processor is to take.

Instruction Fetch & Execute(Cont'd)

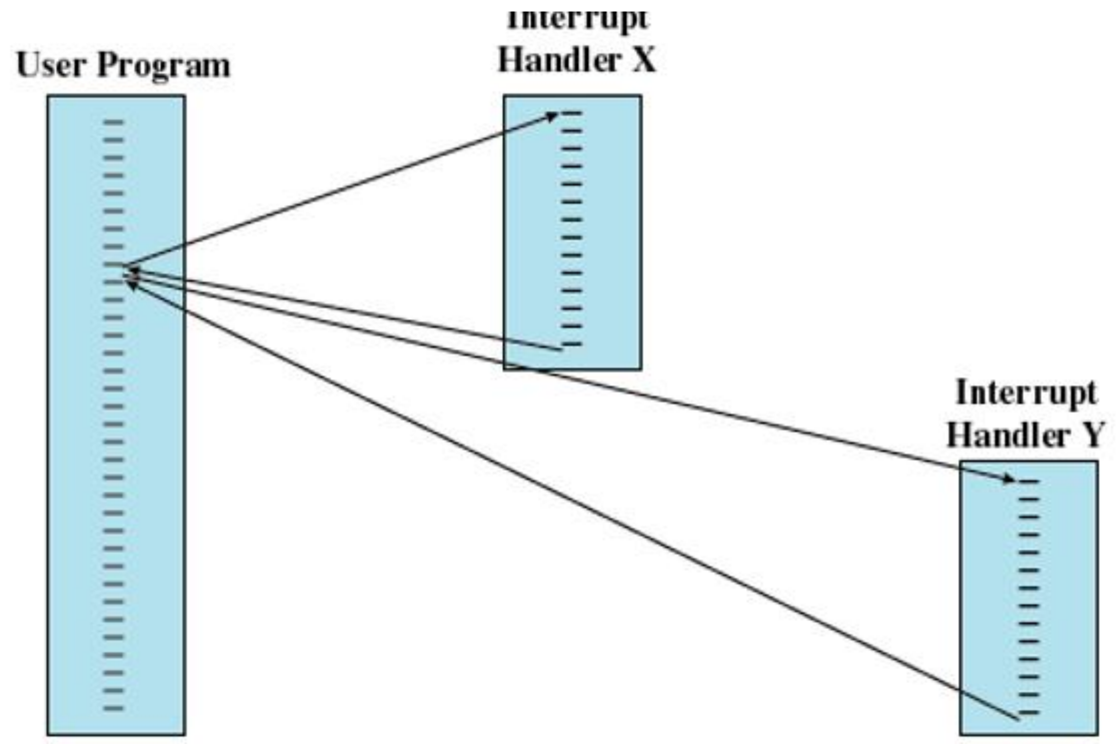
- **Categories**
  - **Processor-memory:** Data may be transferred from processor to memory or from memory to processor.
  - **Processor-I/O:** Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module.
  - **Data processing:** The processor may perform some arithmetic or logic operation on data.
  - **Control:** An instruction may specify that the sequence of execution be altered.

Interrupt Cycle



Multiple Interrupts

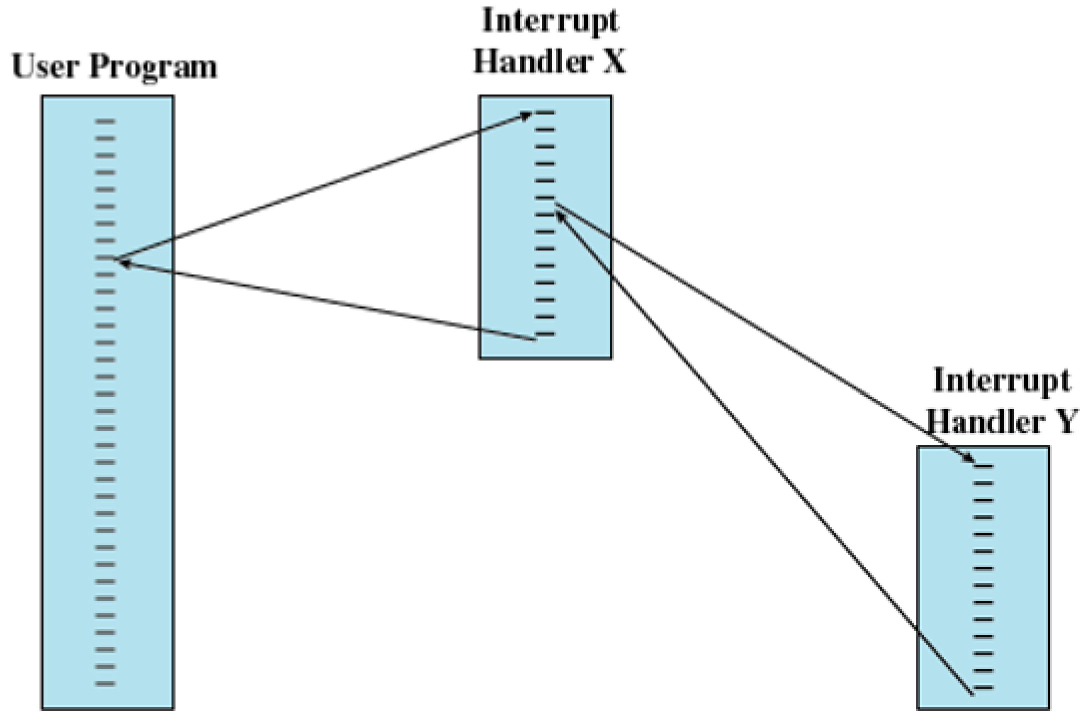
- Disable interrupts while an interrupt is being processed



(a) Sequential interrupt processing

Multiple Interrupts

- Define priorities for interrupt



(b) Nested interrupt processing

Multiple Interrupts

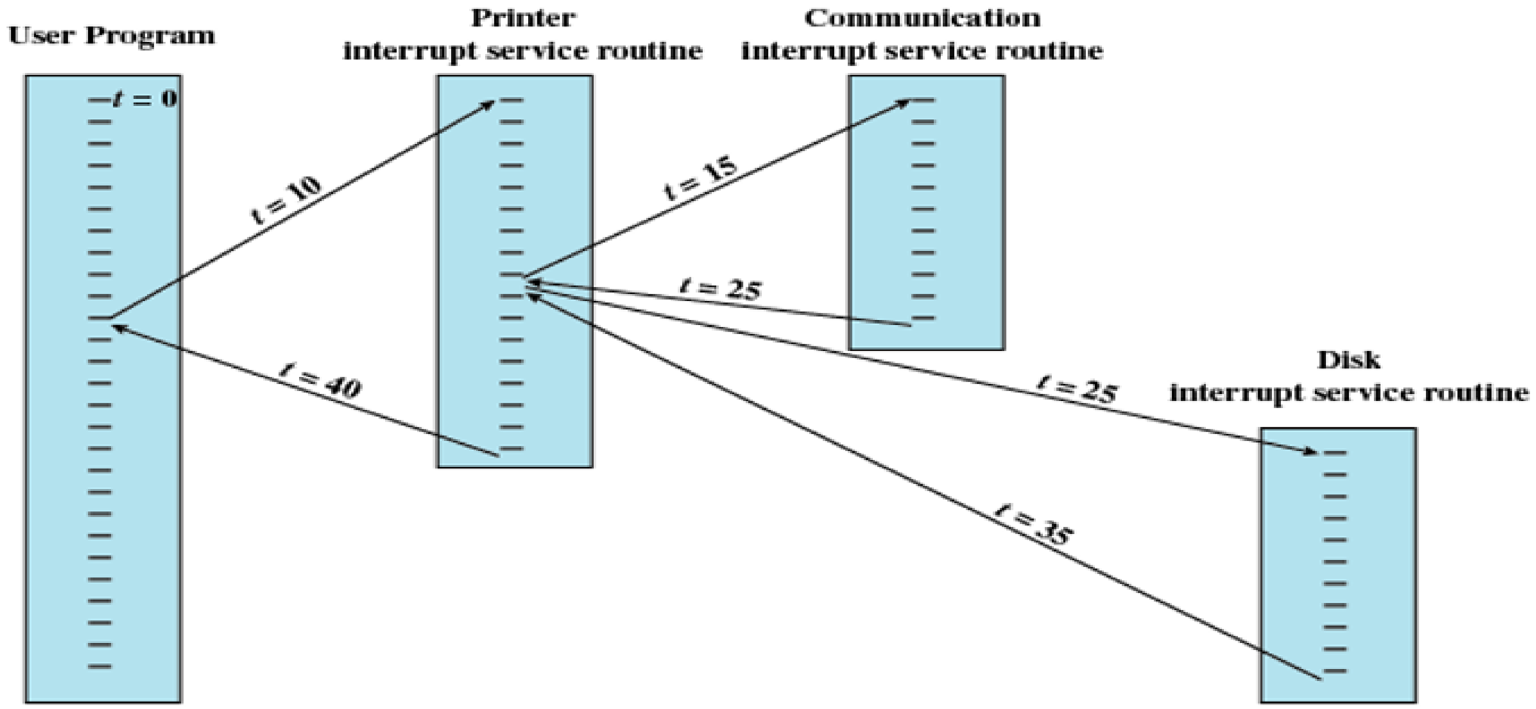
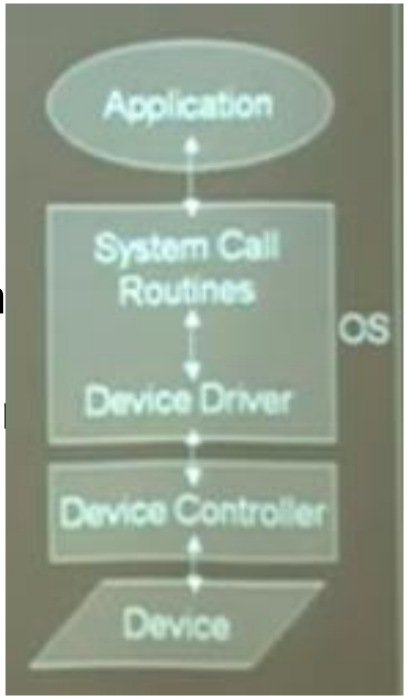

User Program
Printer interrupt service routine
Communication interrupt service routine
Disk interrupt service routine
t = 0
t = 10
t = 15
t = 25
t = 25
t = 35
t = 40

Figure 1.13 Example Time Sequence of Multiple Interrupts

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 waitin for I/O completion
  - **System call** – request to the OS to allow user to wait fo I/O completion
  - **Device-status table** contains entry for each I/O device indicating its type, address, and state
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt



I/O Operations

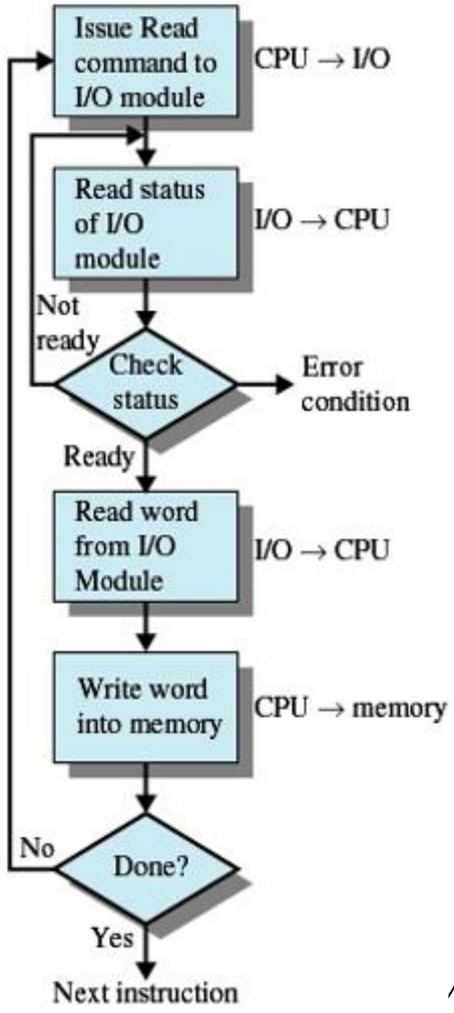
- Three techniques are possible for I/O operations:
- programmed I/O
- interrupt-driven I/O
- and direct memory access (DMA).

Programmed I/O

- I/O module performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur to processor
- Processor checks status periodically until operation is complete.

Processor is responsible for extracting data from main memory for output and storing into main memory

- Disadvantage??

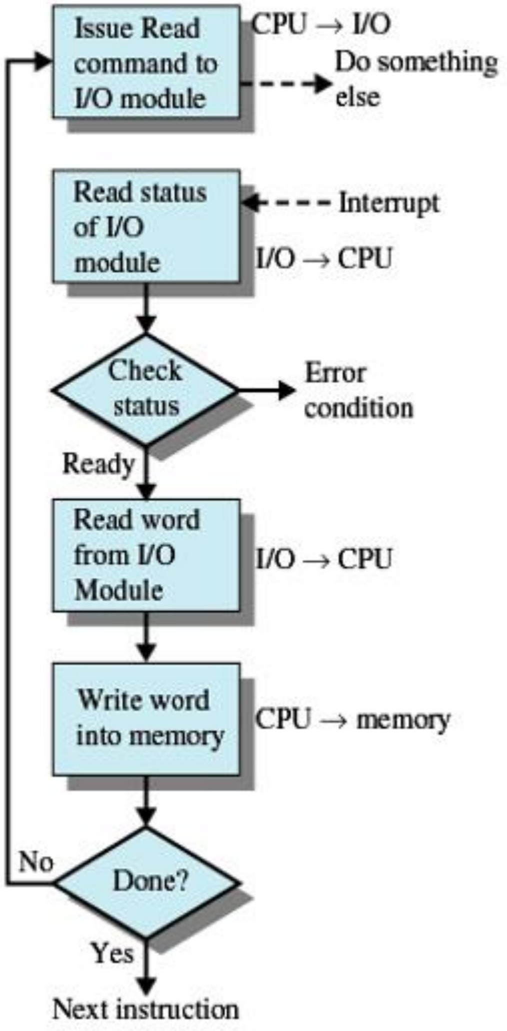


(a) Programmed I/O

Interrupt Driven I/O

- Processor is interrupted when I/O module ready to

  exchange data
- Processor saves context of program executing and begins executing interrupt handler
- No needless waiting
- Consumes a lot of processor time because every word read written passes through the processor

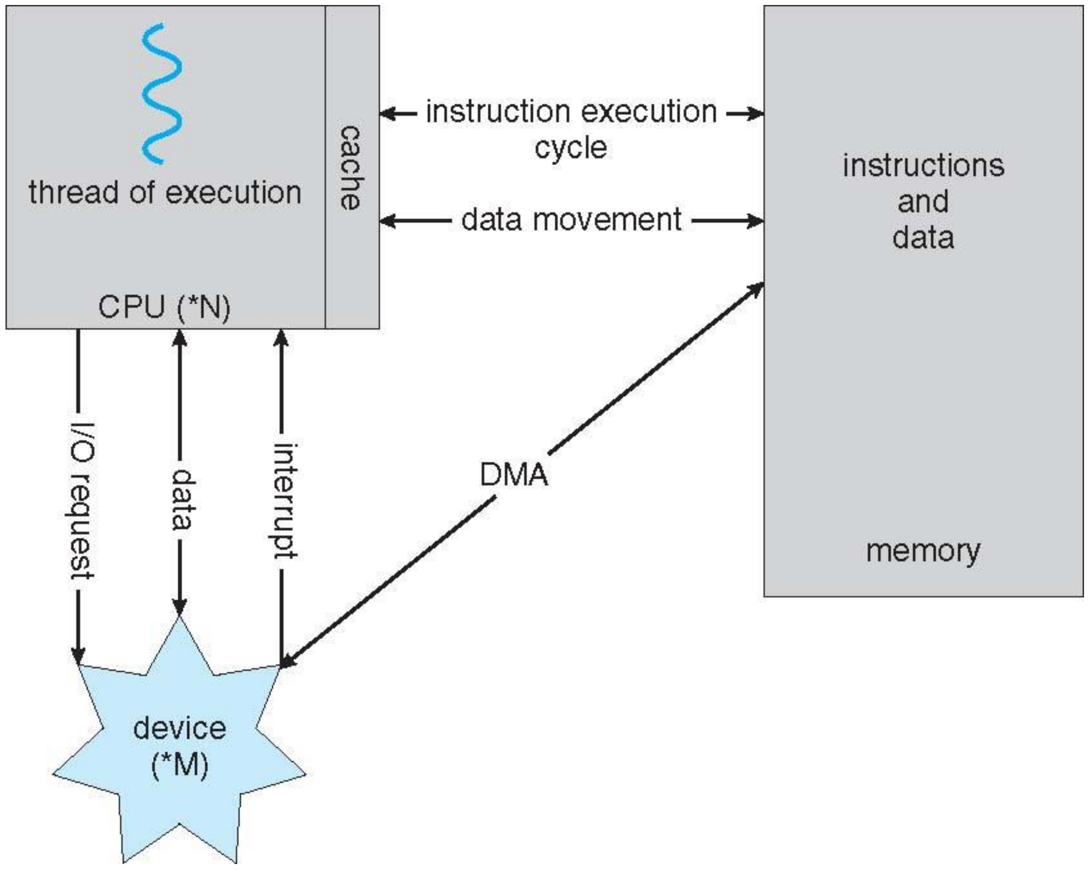


(b) Interrupt-driven I/O

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

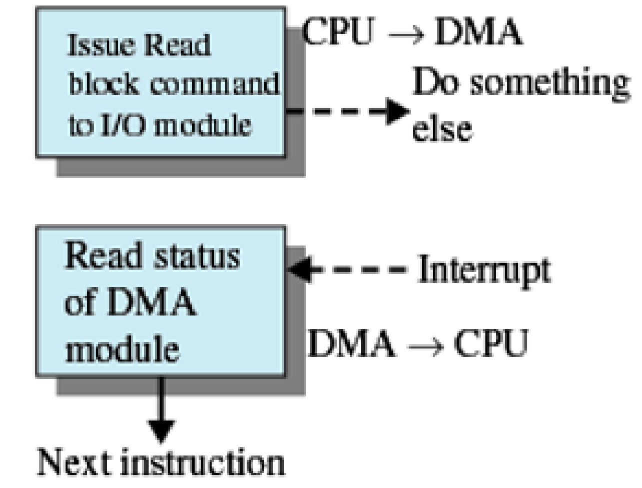
How a Modern Computer Works



A von Neumann architecture

Direct Memory Access(DMA)

- Transfers a block of data directly to or from memory
- When the processor wishes to read or write a block of data, it issues a command to the DMA module, by sending to the DMA module the following information:
  - Whether a read or write is requested
  - The address of the I/O device involved
  - The starting location in memory to read data from or write data to
  - The number of words to be read or written


Issue Read block command to I/O module
CPU → DMA
Do something else
Read status of DMA module
Interrupt
DMA → CPU
Next instruction

(c) Direct memory access

Direct Memory Access(DMA)

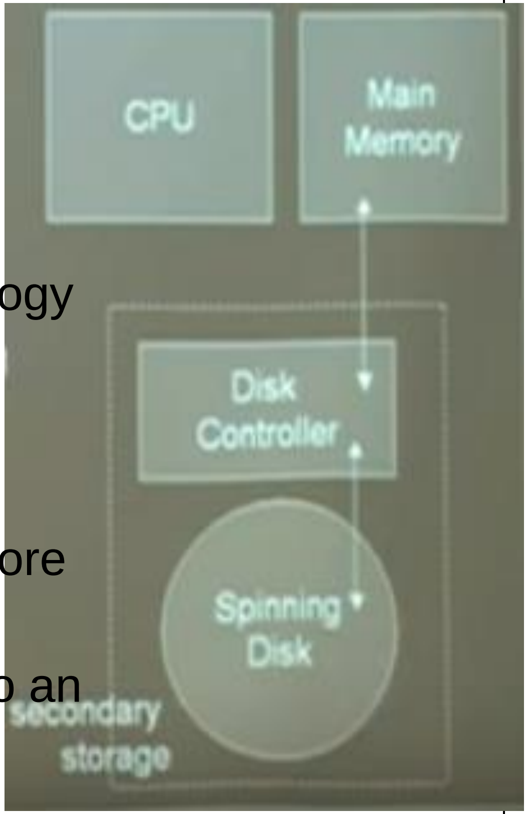
- The DMA module transfers the entire block of data, one word at a time, directly to or from memory without going through the processor.
- An interrupt is sent when the transfer is complete
- the processor is involved only at the beginning and end of the transfer.
- Processor continues with other work

Storage Structure

- **Main memory**

– only large storage media that the CPU can access directly

  - **Random access**
  - Typically **volatile**
  - It commonly is implemented in a semiconductor technology called dynamic random-access memory (DRAM),
  - Forms an array of memory words.
  - Each word has its own address.
  - Interaction is achieved through a sequence of load or store instructions to specific memory addresses.
  - The load instruction moves a word from main memory to an internal register within the CPU
  - whereas the store instruction moves the content of a register to main memory.
  - the CPU automatically loads instructions from main memory for execution.



Storage Structure

Main Memory(Contd..)

- Ideally, we want the programs and data to reside in main memory permanently.
- This arrangement usually is not possible for the following two reasons:

  1. Main memory is usually too small to store all needed programs and. data permanently.

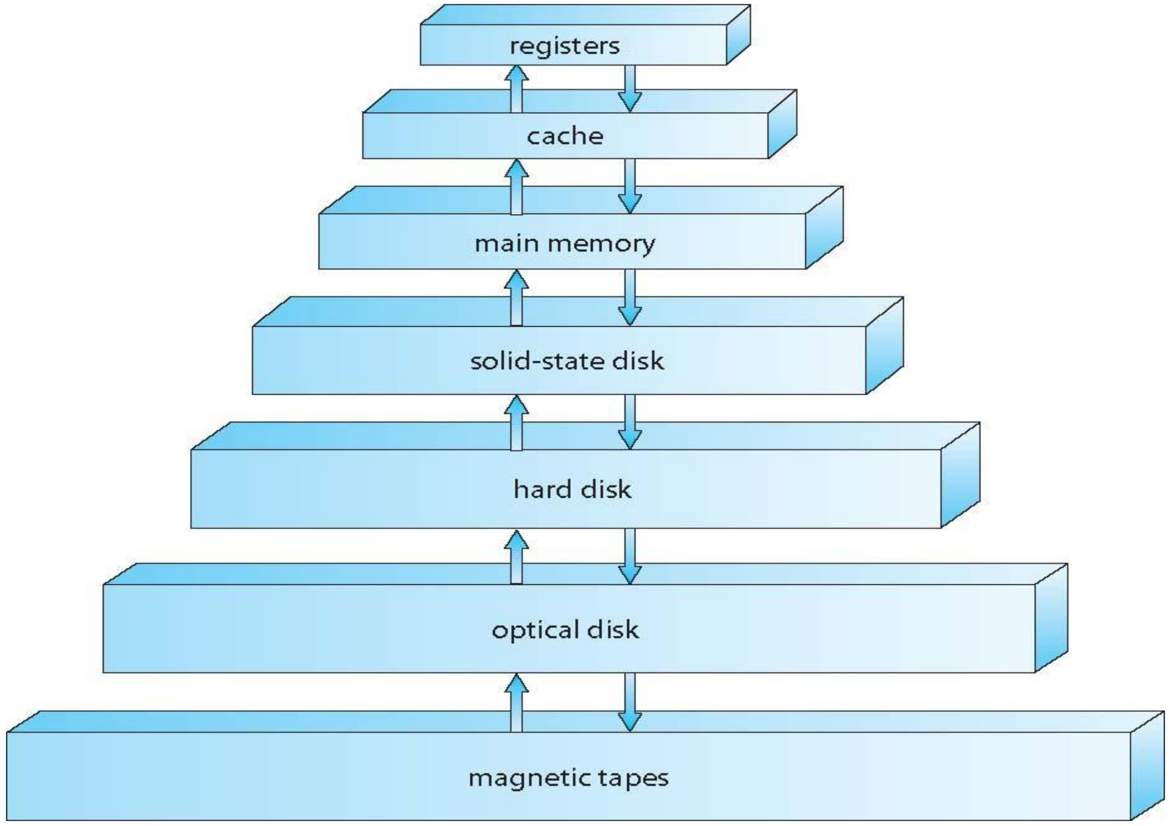
  2. Main memory is a volatile storage device that loses its contents when power is turned off or otherwise lost.

Storage Structure

- **Secondary storage** – extension of main memory that provides large **nonvolatile** storage capacity
- The main requirement for secondary storage is that it be able to hold large quantities of data permanently.
- Most programs (web browsers, compilers, word processors, spread sheets and so on) are stored on a disk until they are loaded into memory.
- The most common secondary-storage device is a magnetic disk.
  - **Hard disks(HDD)** – 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
- **Solid-state disks(SSD)** – faster than hard disks, nonvolatile
  - Various technologies
  - uses NAND-based flash memory to store data electronically.
  - Becoming more popular

Storage Hierarchy

Storage systems organized in hierarchy

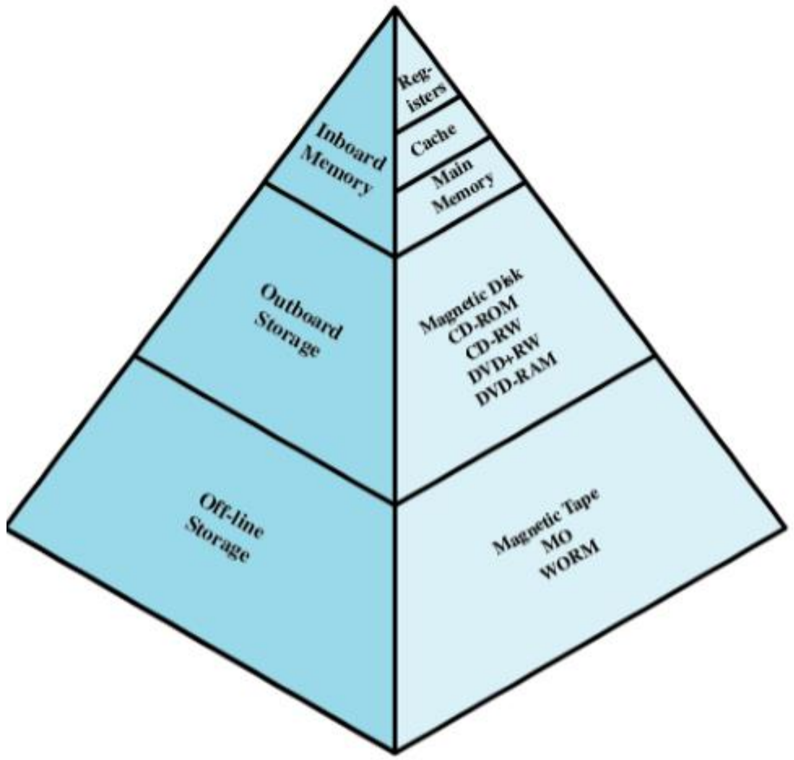


Memory Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access frequency

Going Down the Hierarchy

- Decreasing cost per bit
- Increasing capacity
- Increasing access time
- Decreasing frequency of access of the memory by the processor
  - Locality of reference



Cache Memory

- Invisible to operating system
- Increase the speed of memory
- Processor speed is faster than memory speed
- Exploit the principle of locality
- 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

Cache Memory

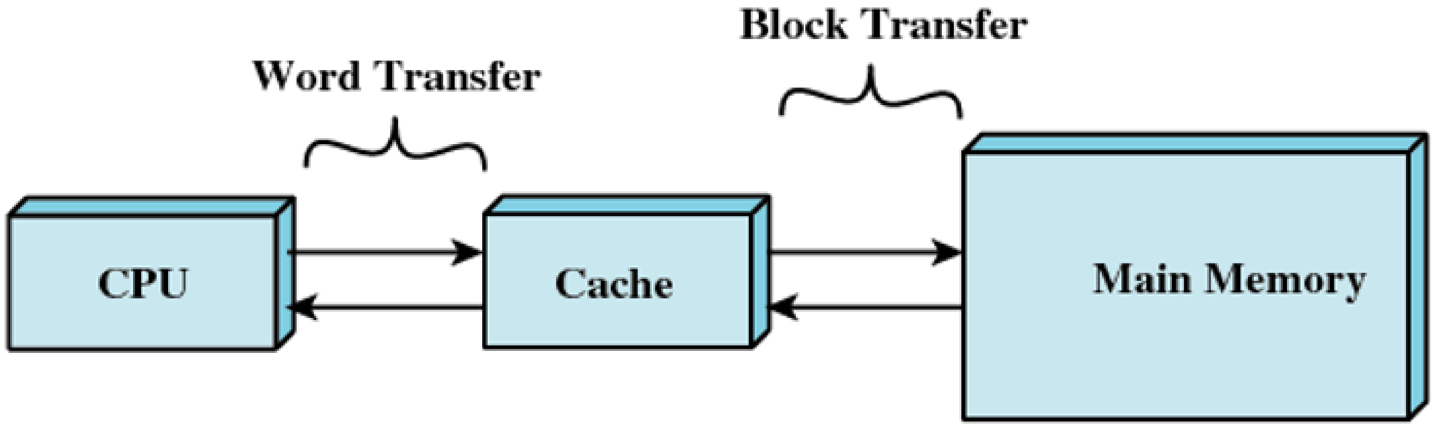


Figure 1.16 Cache and Main Memory

Cache/Main Memory

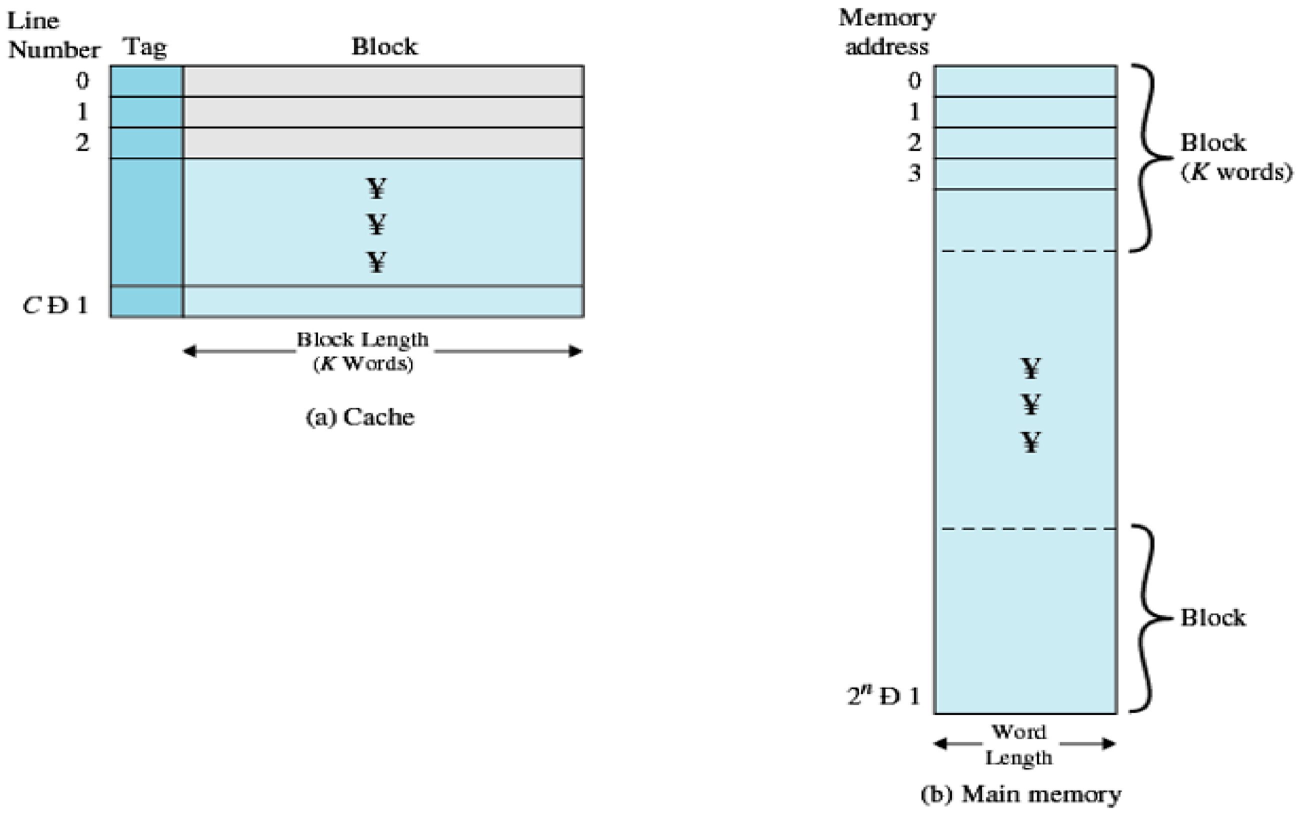


Figure 1.17 Cache/Main-Memory Structure

Cache Design

- **Cache size :**Small caches have a significant impact on performance
- **Block Size:**
  - The unit of data exchanged between cache and main memory.
  - As the block size increases from very small to larger sizes, the hit ratio will at first increase because of the principle of locality:
  - the high probability that data in the vicinity of a referenced word are likely to be referenced in the near future.

Cache Design

- **Mapping Function:**
  - When a new block of data is read into the cache, the mapping function determines which cache location the block will occupy.
  - The more flexible the mapping function, the more scope we have to design a replacement algorithm to maximize the hit ratio.
- **Replacement Algorithm:**
  - The replacement algorithm chooses, within the constraints of the mapping function, which block to replace when a new block is to be loaded into the cache and the cache already has all slots filled with other blocks.
  - Replace the block that is least likely to be needed again in the near future(LRU Algorithm)

Cache Design

- **Write Policy:**
  - If the contents of a block in the cache are altered, then it is necessary to write it back to main memory before replacing it.
  - When the memory write operation takes place.
  - Can occur every time block is updated (Write Through)
  - Can occur only when block is replaced (Write Back)
    - Minimizes memory write operations
    - Leaves main memory in an obsolete state

- Reading Assignment 1:
- What is cache coherence?
- What are different levels of Cache? And what they mean?

Cache Design

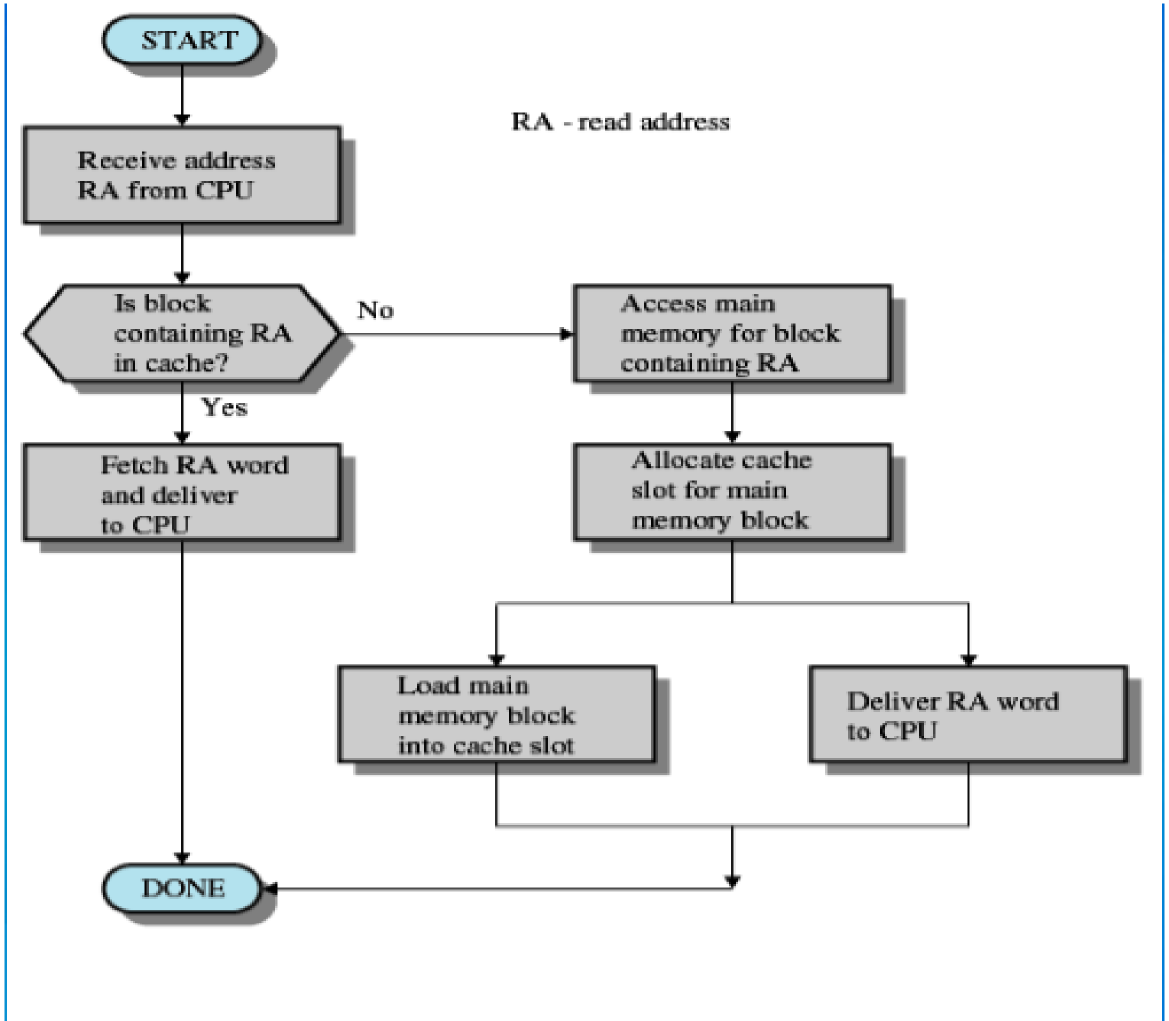

START
RA - read address
Receive address RA from CPU
Is block containing RA in cache?
No
Access main memory for block containing RA
Yes
Fetch RA word and deliver to CPU
Allocate cache slot for main memory block
Load main memory block into cache slot
Deliver RA word to CPU
DONE

Figure 1.18 Cache Read Operation

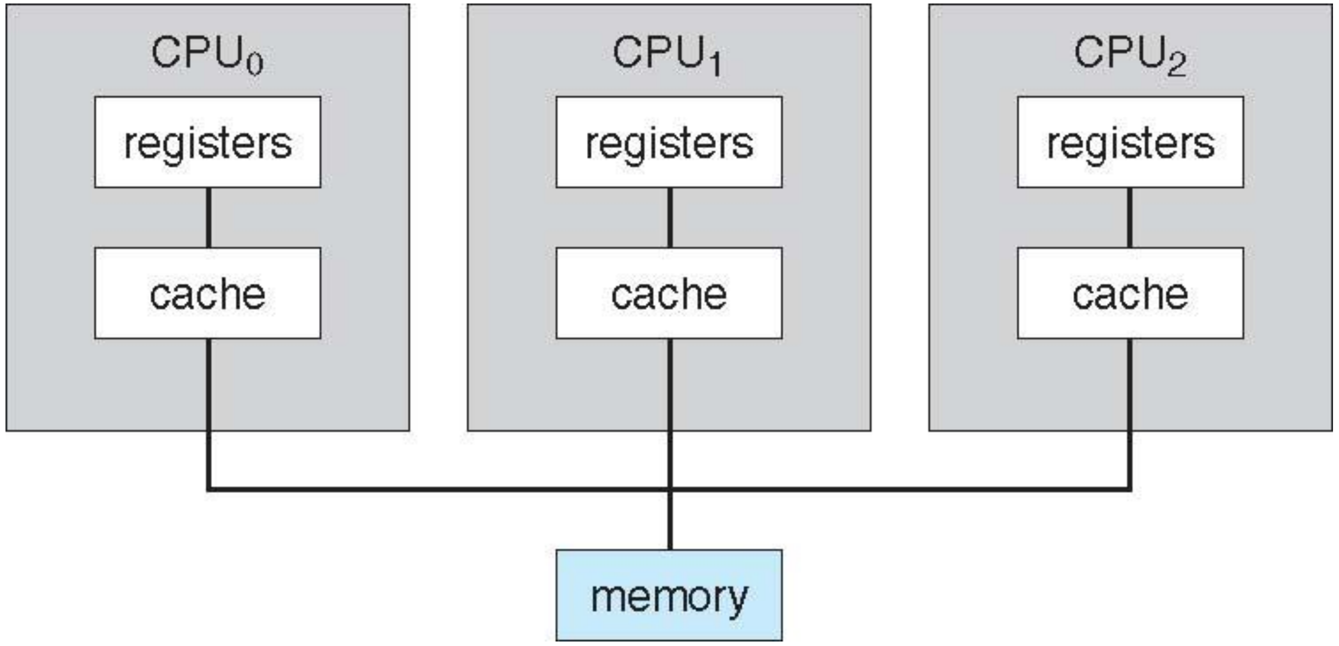
Computer-System Architecture

- Most systems use a single general-purpose processor
  - 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**(cheaper than using multiple computers)
    3. **Increased reliability** – graceful degradation or fault tolerance
  - Two types:
    1. **Asymmetric Multiprocessing** – each processor is assigned a specie task.(Master-Slave)
    2. **Symmetric Multiprocessing** – each processor performs all tasks

Parallel Systems (Cont.)

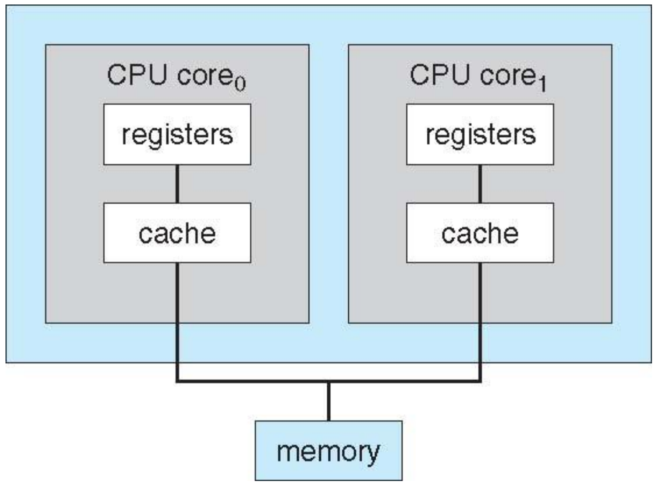
- *Symmetric multiprocessing (SMP)*
  - Each processor runs an identical copy of the operating system.
  - Many processes can run at once without performance deterioration.
  - Most modern operating systems support SMP
- *Asymmetric multiprocessing*
  - Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.
  - More common in extremely large systems

Symmetric Multiprocessing Architecture



A Dual-Core Design

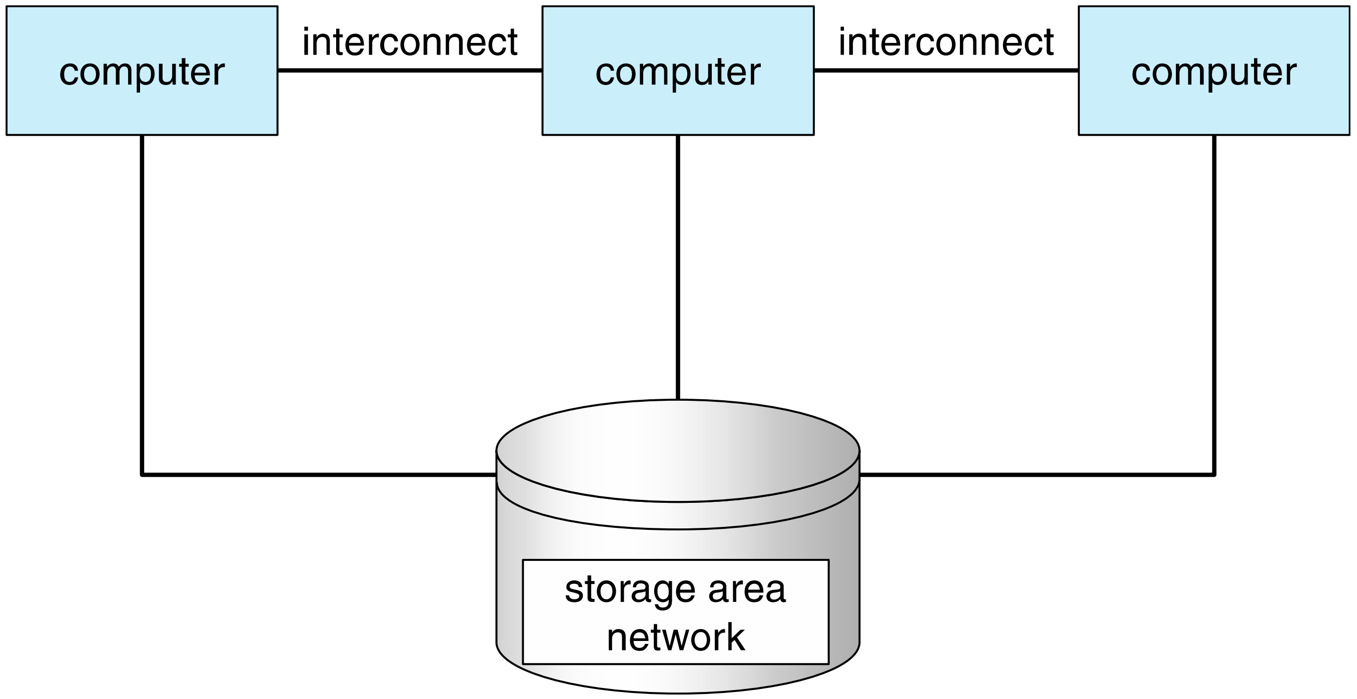
- Multi-chip and **multicore**



Clustered Systems

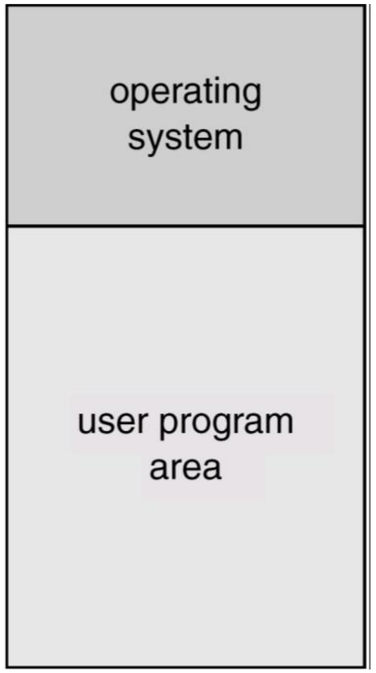
- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)**
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** 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



Memory Layout for a Simple Batch System

- A batch system is a type of operating system or processing environment in which tasks or jobs are processed in groups (batches), without user interaction during execution.

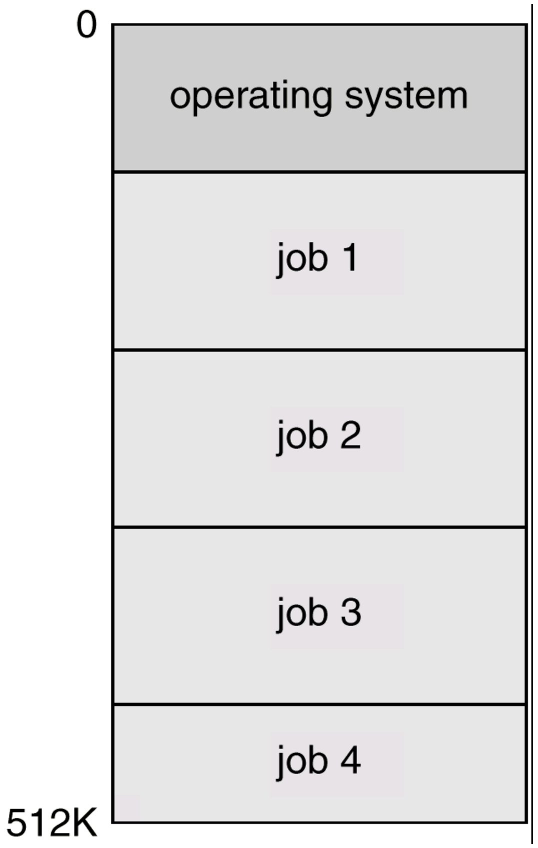


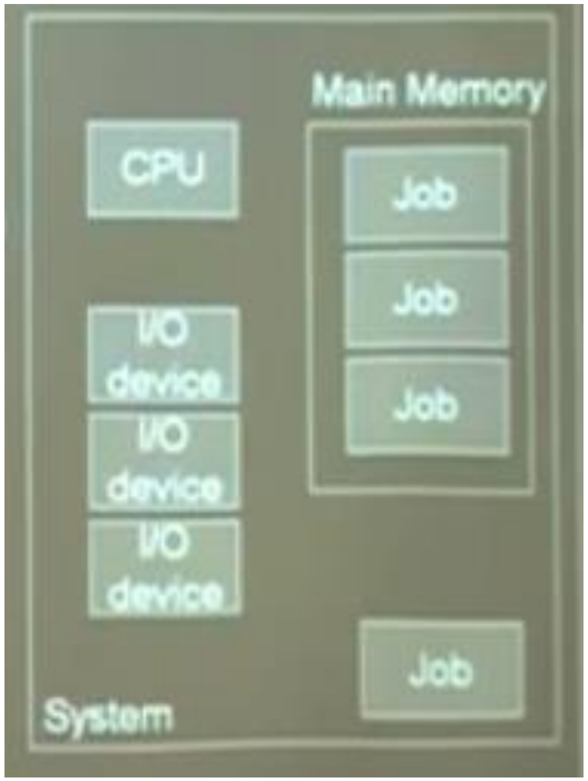
OS Features Needed for Multiprogramming

- **Multiprogramming** (**Batch system**) needed for efficiency
- Multiprogramming is a fundamental concept in operating systems that allows multiple programs or processes to run concurrently on a single system.
  - 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

Multiprogrammed Batch Systems

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





Operating System Structure

- **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

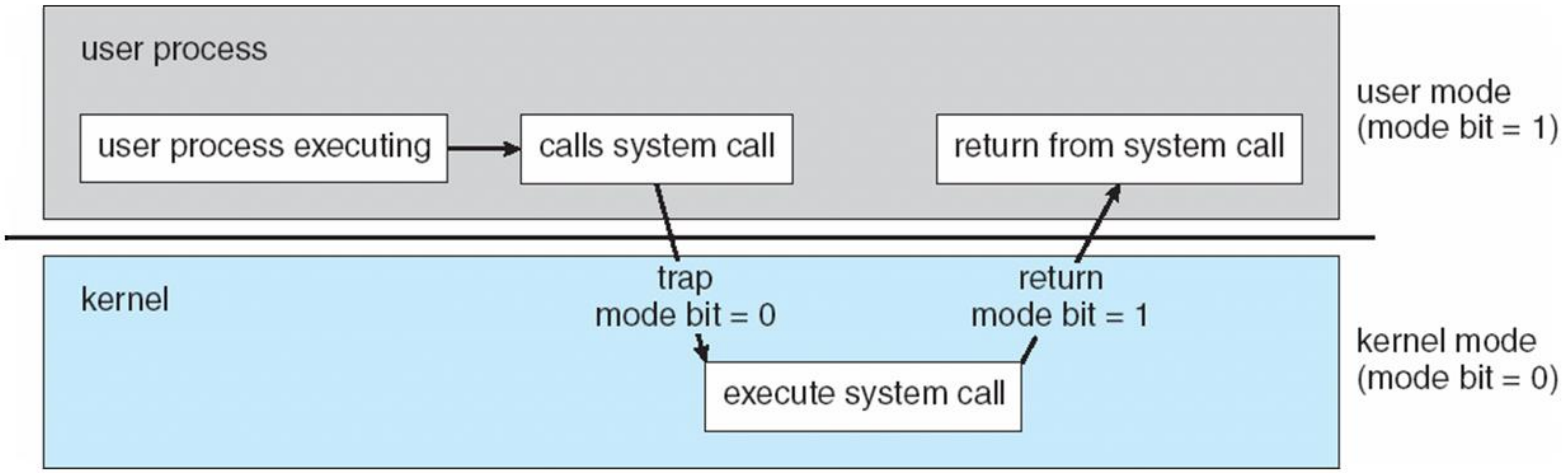
Operating-System Operations

- **Interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap):**
    - Software error (e.g., division by zero)
    - Request for operating system service
    - Other process problems include infinite loop, handled by timer interrupt
    - processes modifying each other or the operating system→ handled by dual mode

Operating-System Operations (cont.)

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - 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

Transition from User to Kernel Mode

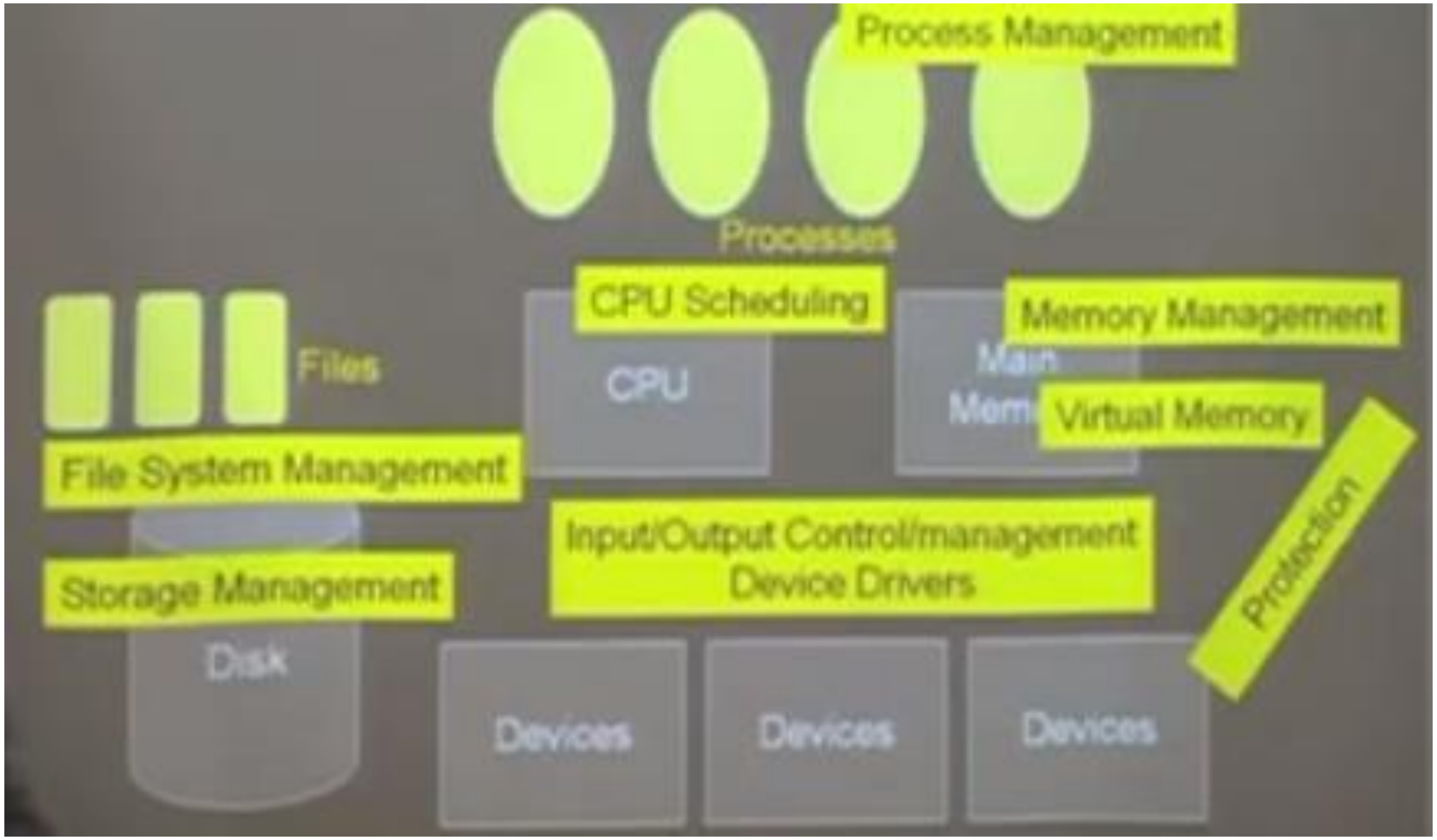


Preventing a process hogging resource

- Timer to prevent infinite loop / process hogging resources
  - Timer is set to interrupt the computer after some time period
    - Can be fixed or variable time period
  - CPU executes a program(process)
  - Timer device send an interrupt after specified time
  - CPU starts executing time handler: OS gain controls
  - Can schedule same or different process

  - Keep a counter that is decremented by the physical clock.
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

Major systems and components of Operating System



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 CPUs among the processes / threads

Process Management Activities

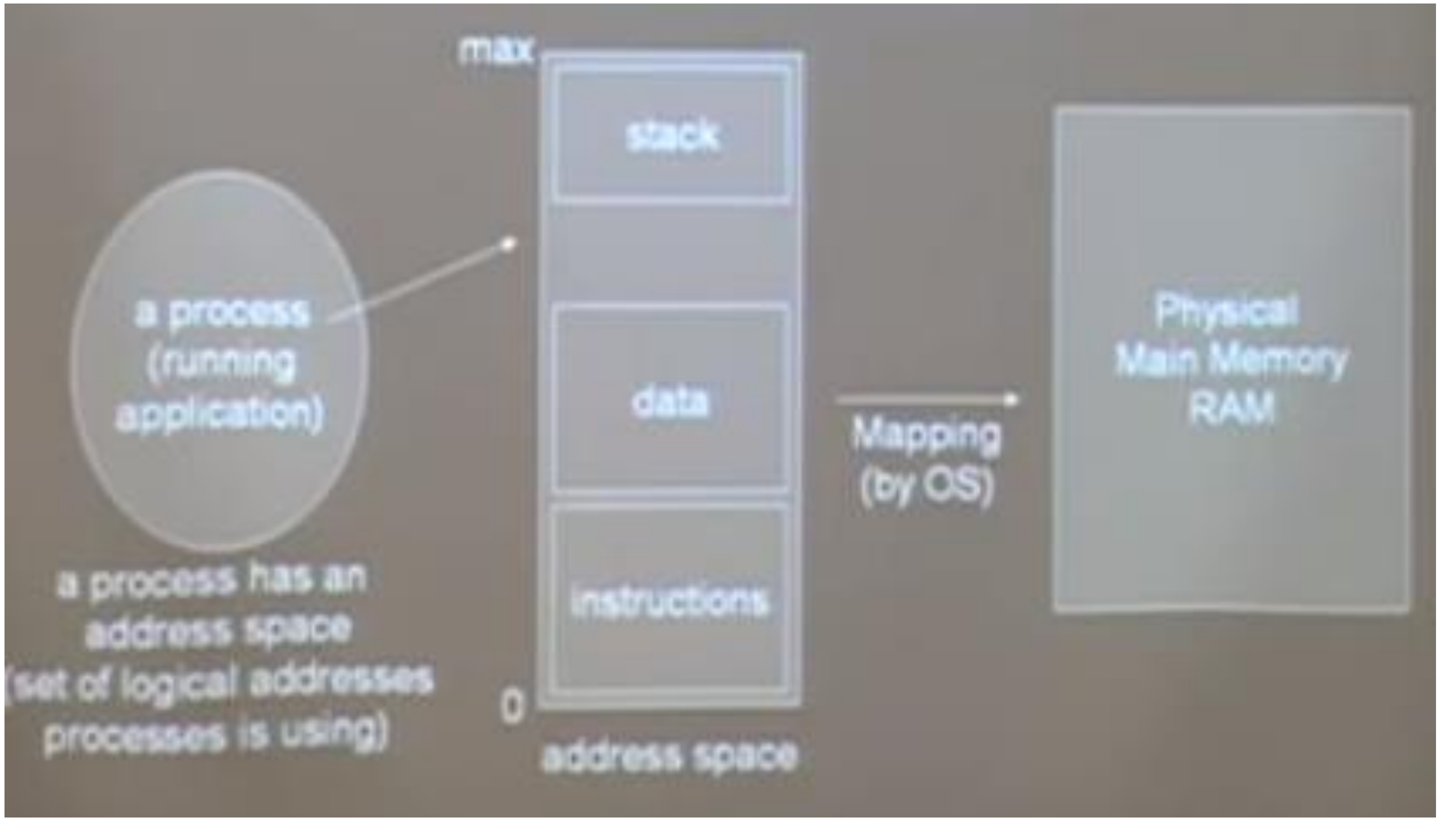
The operating system is responsible for the following activities in connection with process management:

- 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
- Providing mechanisms for deadlock handling

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and 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

Memory Management



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 activities include
    - 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)

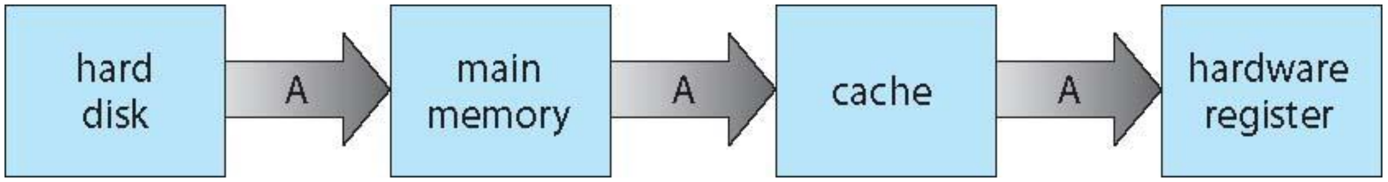
Performance of Various Levels of Storage

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

Movement between levels of storage hierarchy can be explicit or implicit

Migration of data “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

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 (the 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