Advance Operating Systems

Lecture # 1
Introduction

Course Books

Textbook: ‘Operating System Concepts’ by Silberschatz, Galvin and Gagne, 8th Edition

Reference Book:

Operating System Concepts’ by William Stallings 6th Edition

Course Outline

Grading Policy

Assignments	10%
Quizzes	10%
Presentations/Viva	20%
Mid-Semester Exam	20%
Final Exam	40%

Assignments:

Assignments will be handwritten, submitted in proper folder.

No plagiarism and copying as per HEC Policy.

Quizzes:

Quizzes may be announced / unannounced.

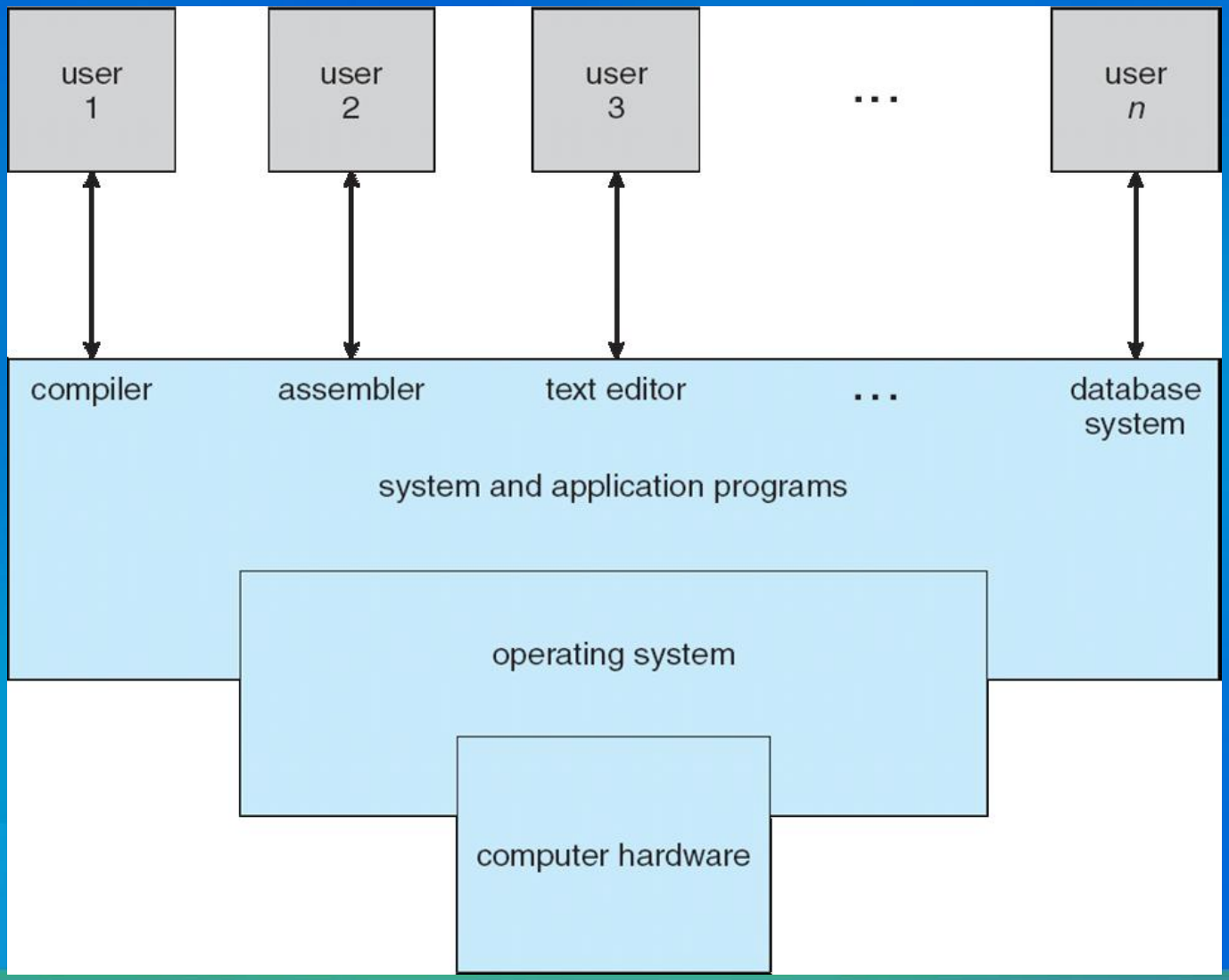
Computer System Overview

Chapter 1

Operating System

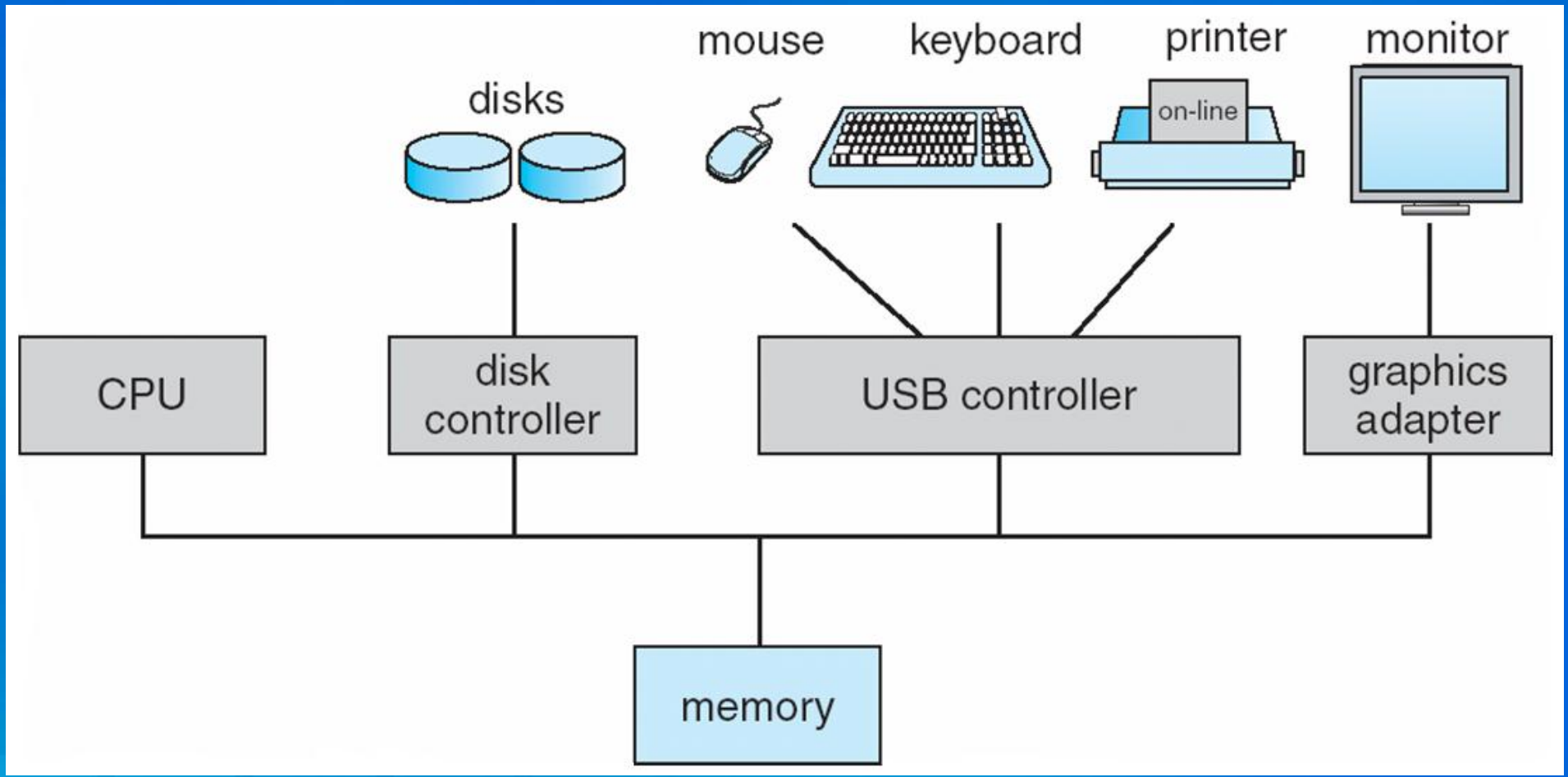
- Exploits the hardware resources of one or more processors
- Provides a set of services to system users
- Manages secondary memory and I/O devices
- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Computer System Stack



Basic Elements

- Processor
- Main Memory
  - volatile
  - referred to as real memory or primary memory
- I/O modules
  - secondary memory devices
  - communications equipment
  - terminals
- System bus
  - communication among processors, memory, and I/O modules


mouse
keyboard
printer
monitor
disks
on-line
CPU
disk controller
USB controller
graphics adapter
memory

Top-Level Components

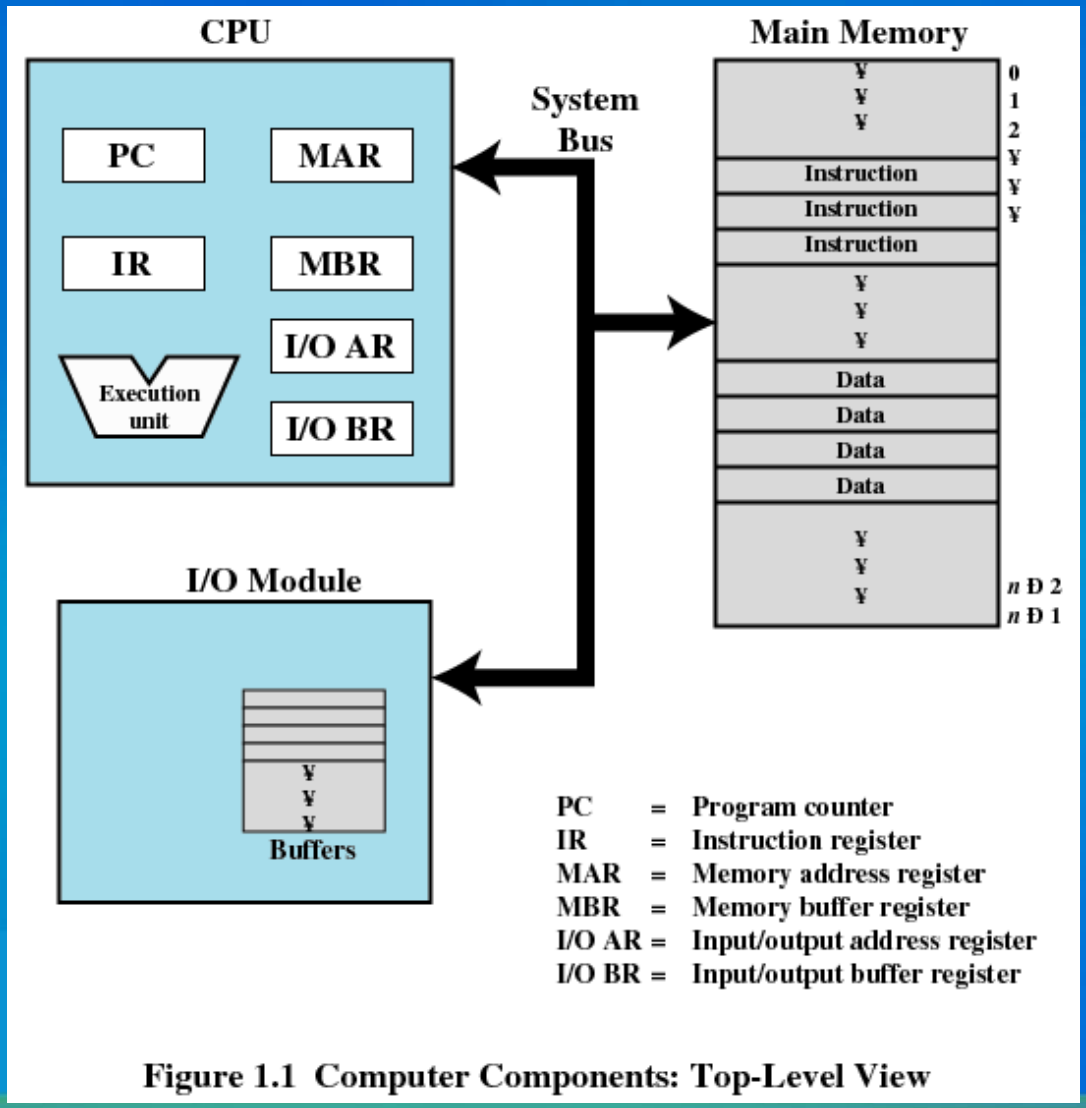


Figure 1.1 Computer Components: Top-Level View

Processor Registers

- User-visible registers
  - Enable programmer to minimize main-memory references by optimizing register use
  - Data, Address (Index, Segment pointer, Stack pointer)
- Control and status registers
  - Used by processor to control operation of the processor
  - Used by privileged operating-system routines to control the execution of programs
  - Examples ? ? ?

Instruction Execution

- Two steps
  - Processor reads instructions from memory
    - Fetches
  - Processor executes each instruction

Instruction Cycle

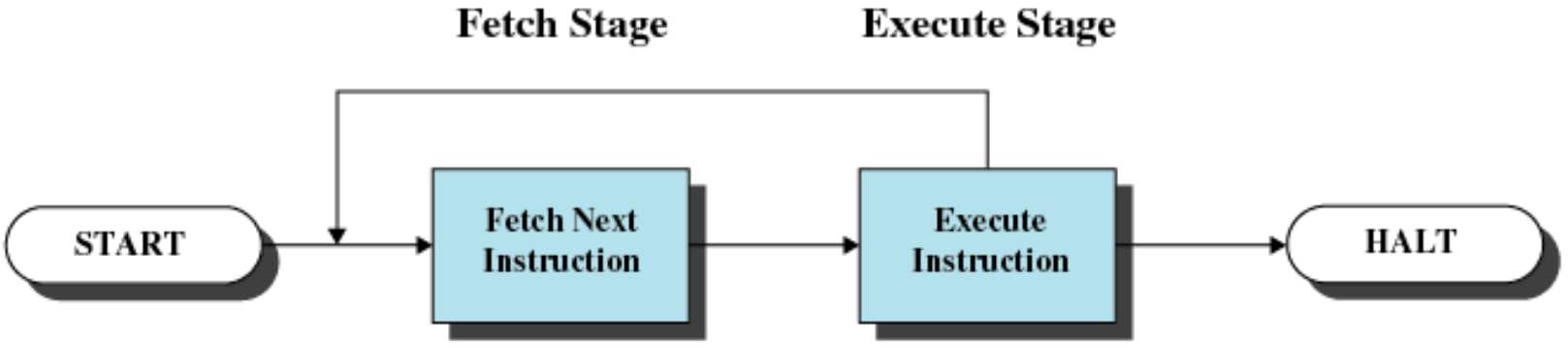


Figure 1.2 Basic Instruction Cycle

Instruction Fetch and 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

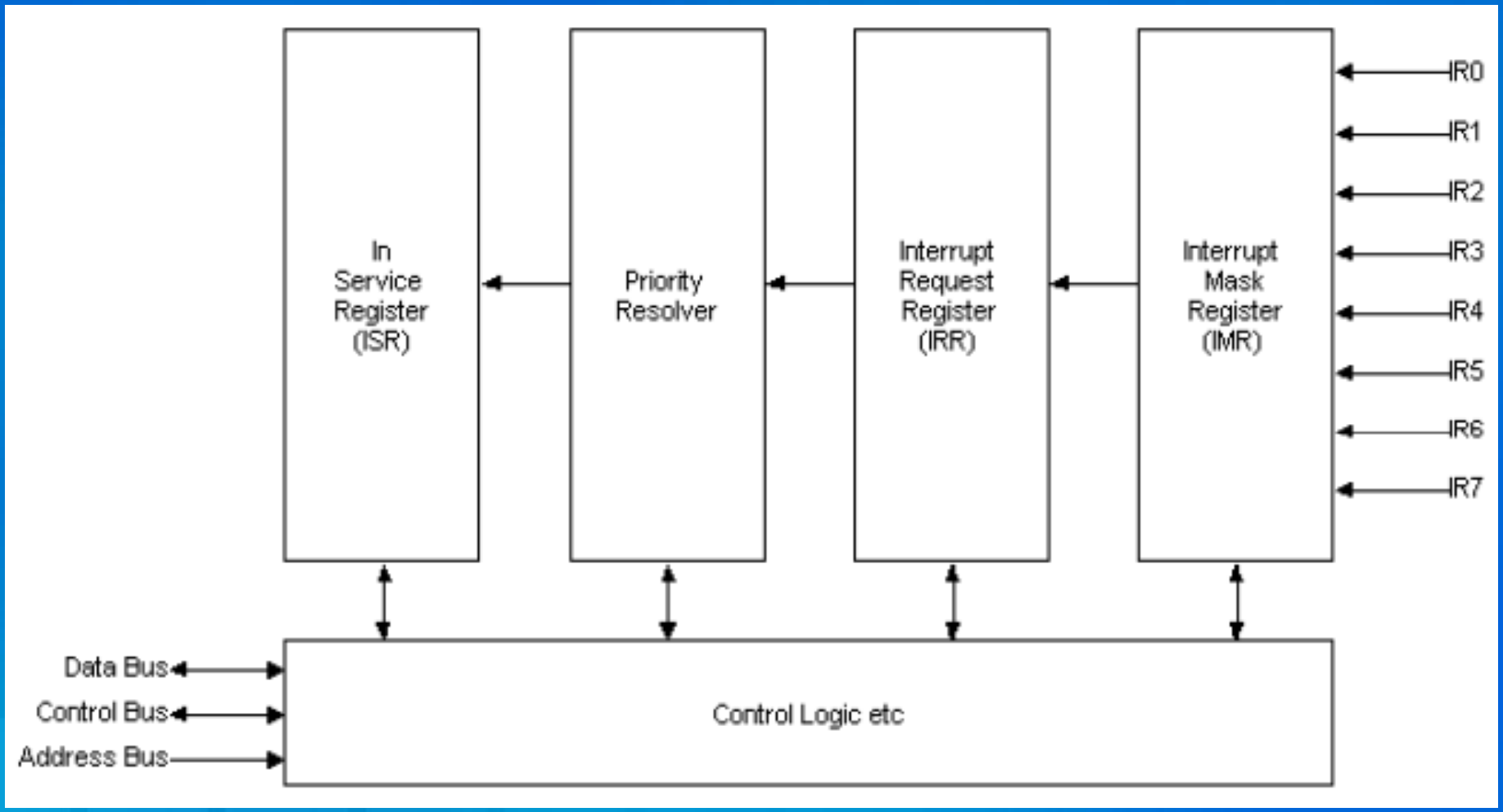
Instruction Register

- Fetched instruction is placed in the instruction register
- Categories
  - Processor-memory
    - Transfer data between processor and memory
  - Processor-I/O
    - Data transferred to or from a peripheral device
  - Data processing
    - Arithmetic or logic operation on data
  - Control
    - Alter sequence of execution

Interrupts

- Interrupt the normal sequencing of the processor
- Most I/O devices are slower than the processor
  - Processor must pause to wait for device
- 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* is a software-generated interrupt caused either by an error or a user request

PIC (Programmable Interrupt Controller)

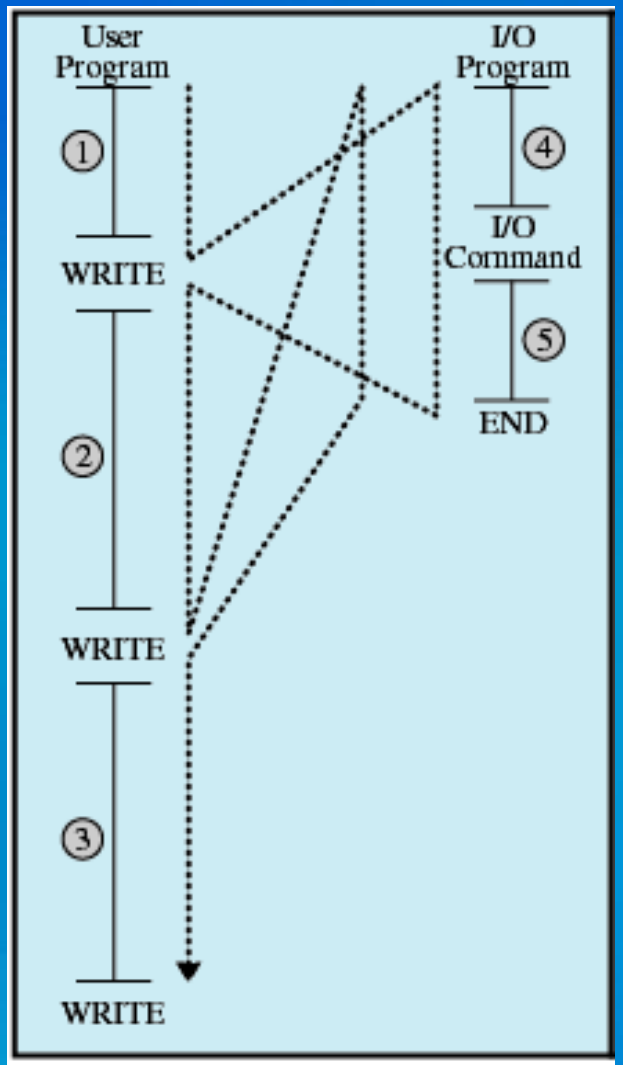


Classes of Interrupts

Table 1.1 Classes of Interrupts

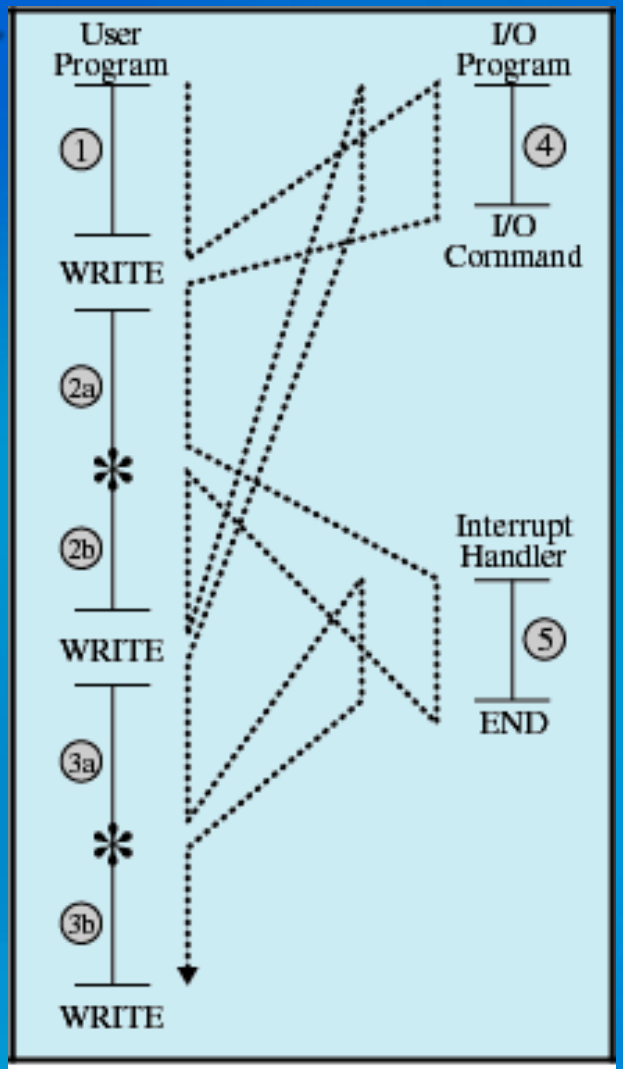
Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows 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.

Program Flow of Control Without Interrupts



(a) No interrupts

Program Flow of Control With Interrupts, Short I/O Wait



(b) Interrupts; short I/O wait

Interrupt Handler

- Program to service a particular I/O device
- Generally part of the operating system

Interrupts

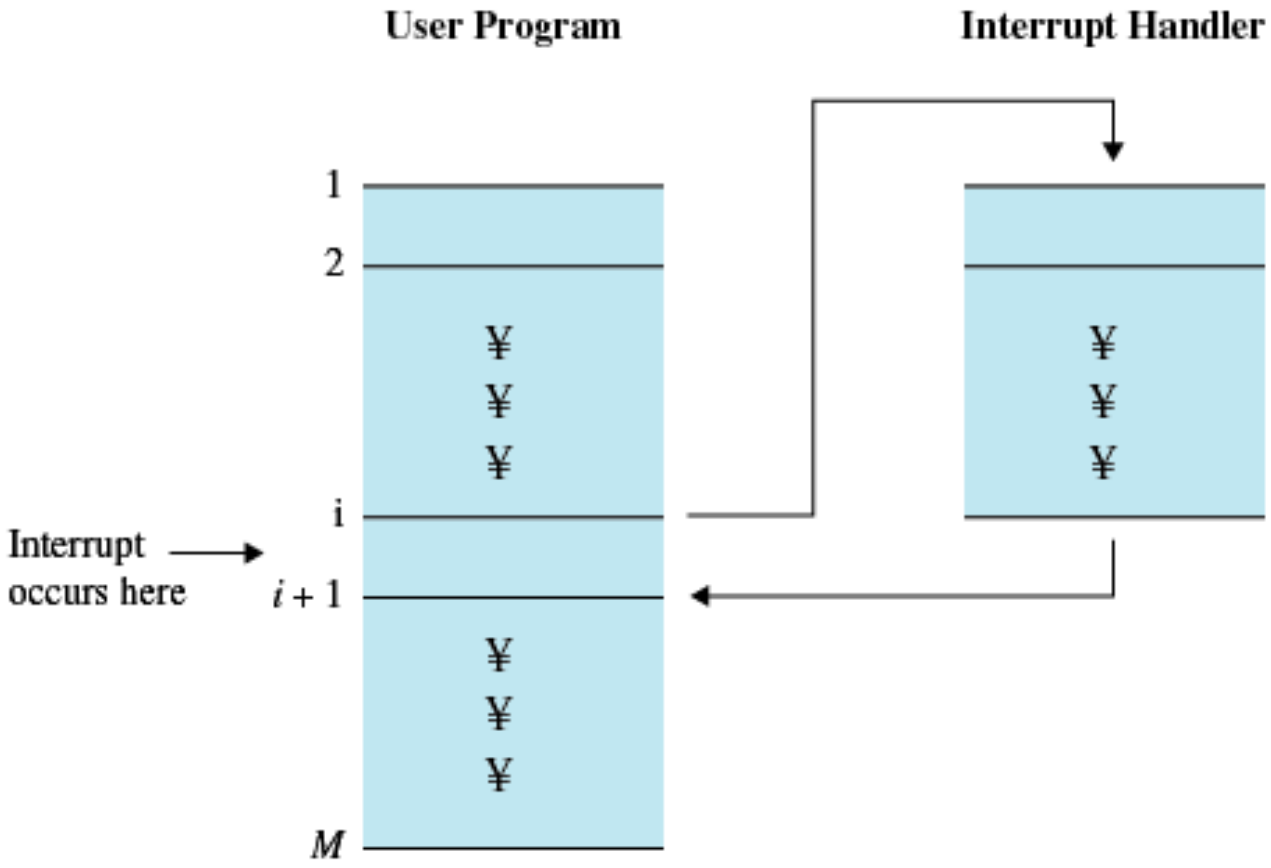


Figure 1.6 Transfer of Control via Interrupts

Interrupt Cycle

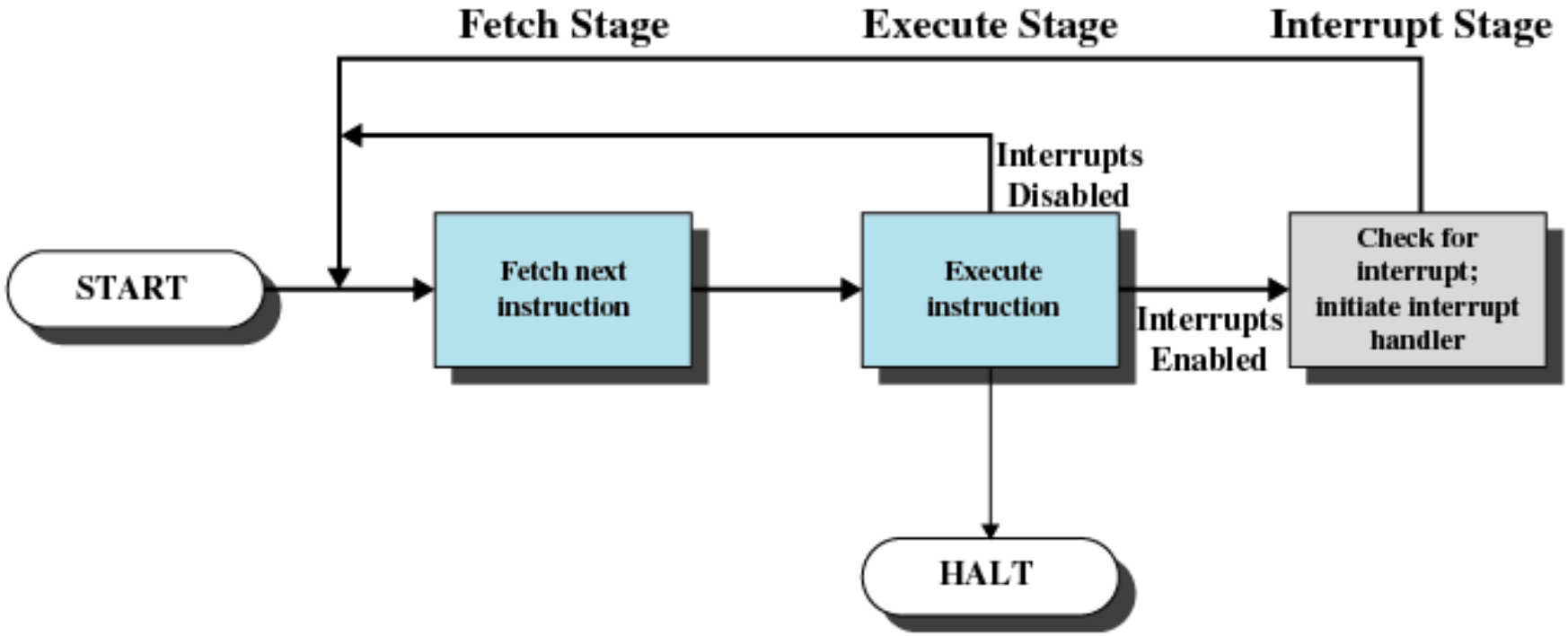


Figure 1.7 Instruction Cycle with Interrupts

Interrupt Cycle

- Processor checks for interrupts
- If no interrupts fetch the next instruction for the current program
- If an interrupt is pending, suspend execution of the current program, and execute the interrupt-handler routine

Simple Interrupt Processing

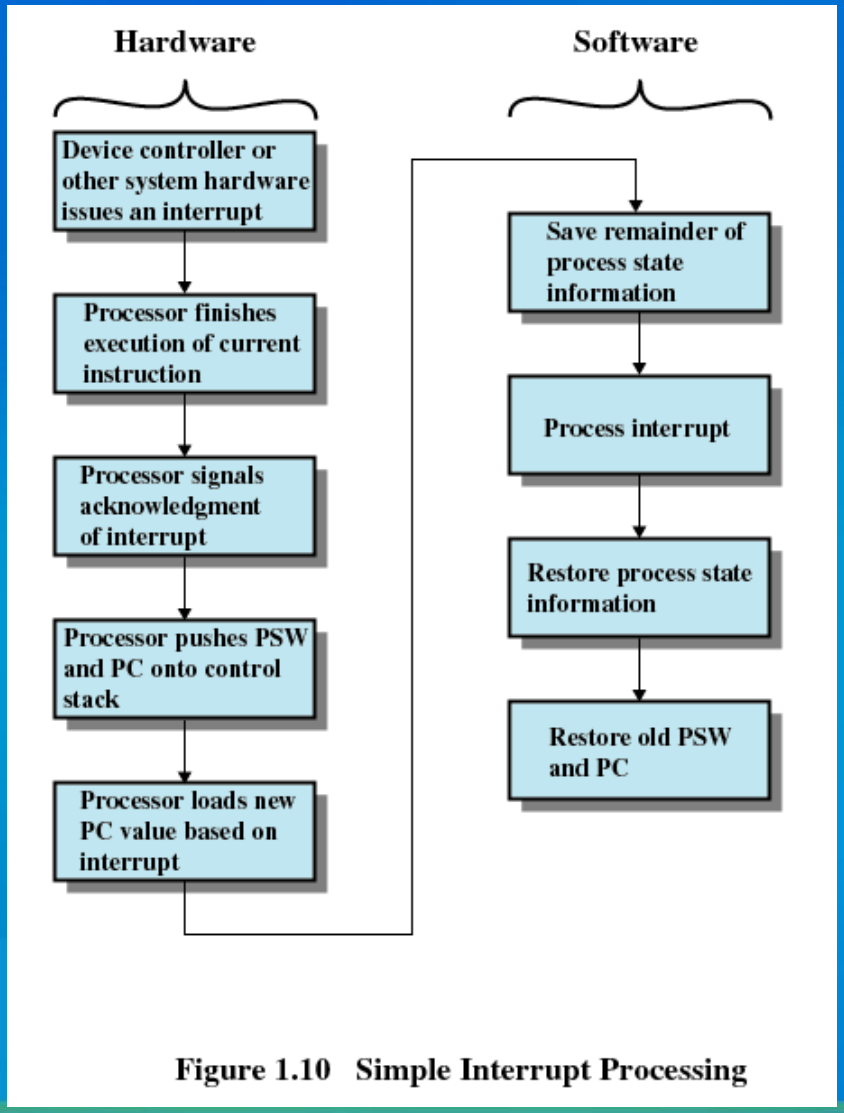
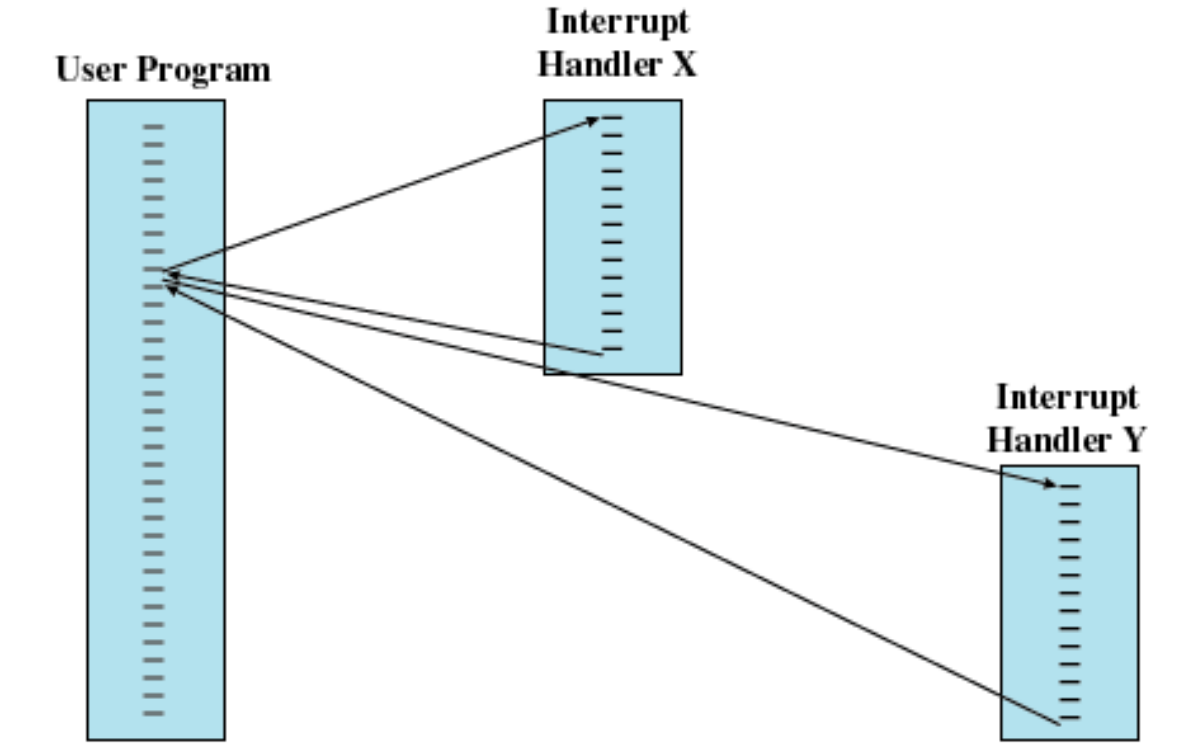


Figure 1.10 Simple Interrupt Processing

Multiple Interrupts

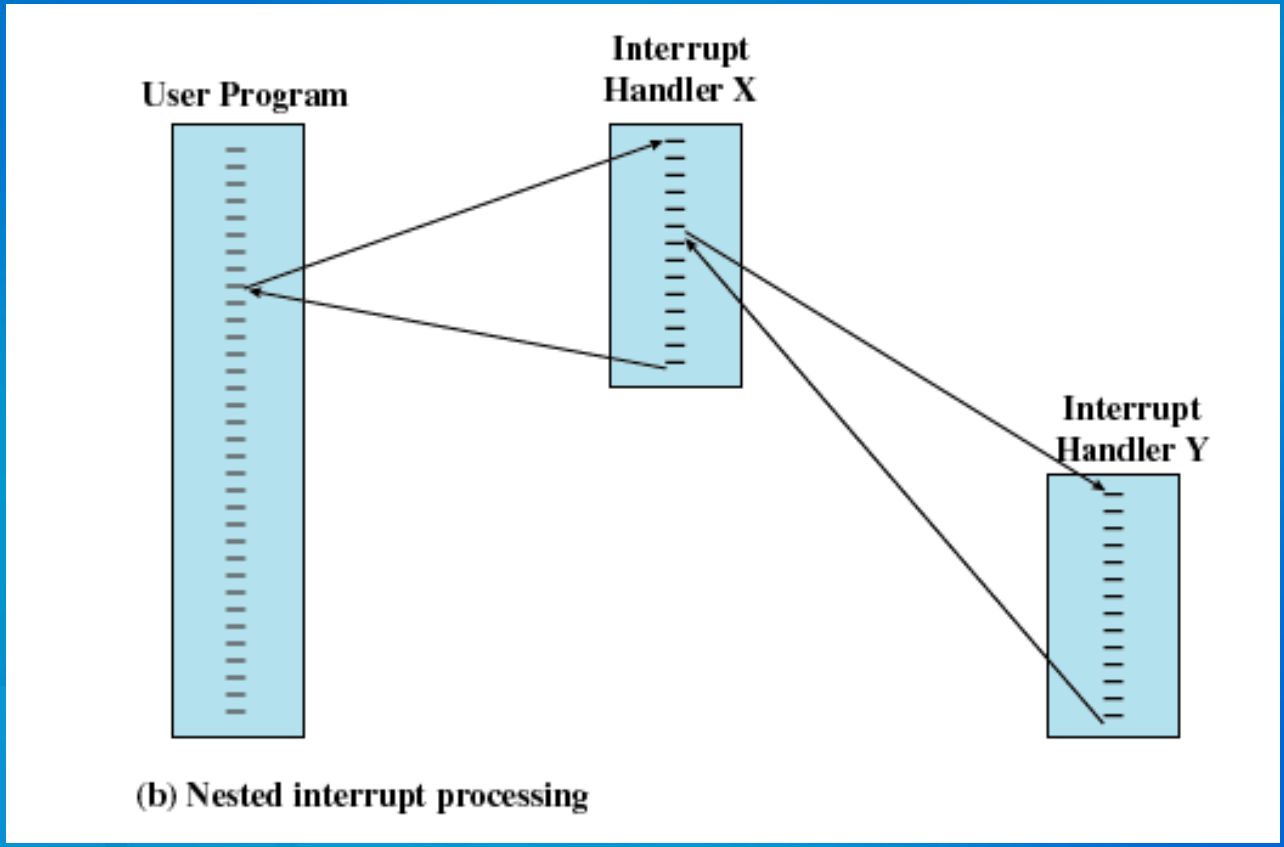
- Disable interrupts while an interrupt is being processed



(a) Sequential interrupt processing

Multiple Interrupts

- Define priorities for interrupts



(b) Nested interrupt processing

Multiple Interrupts

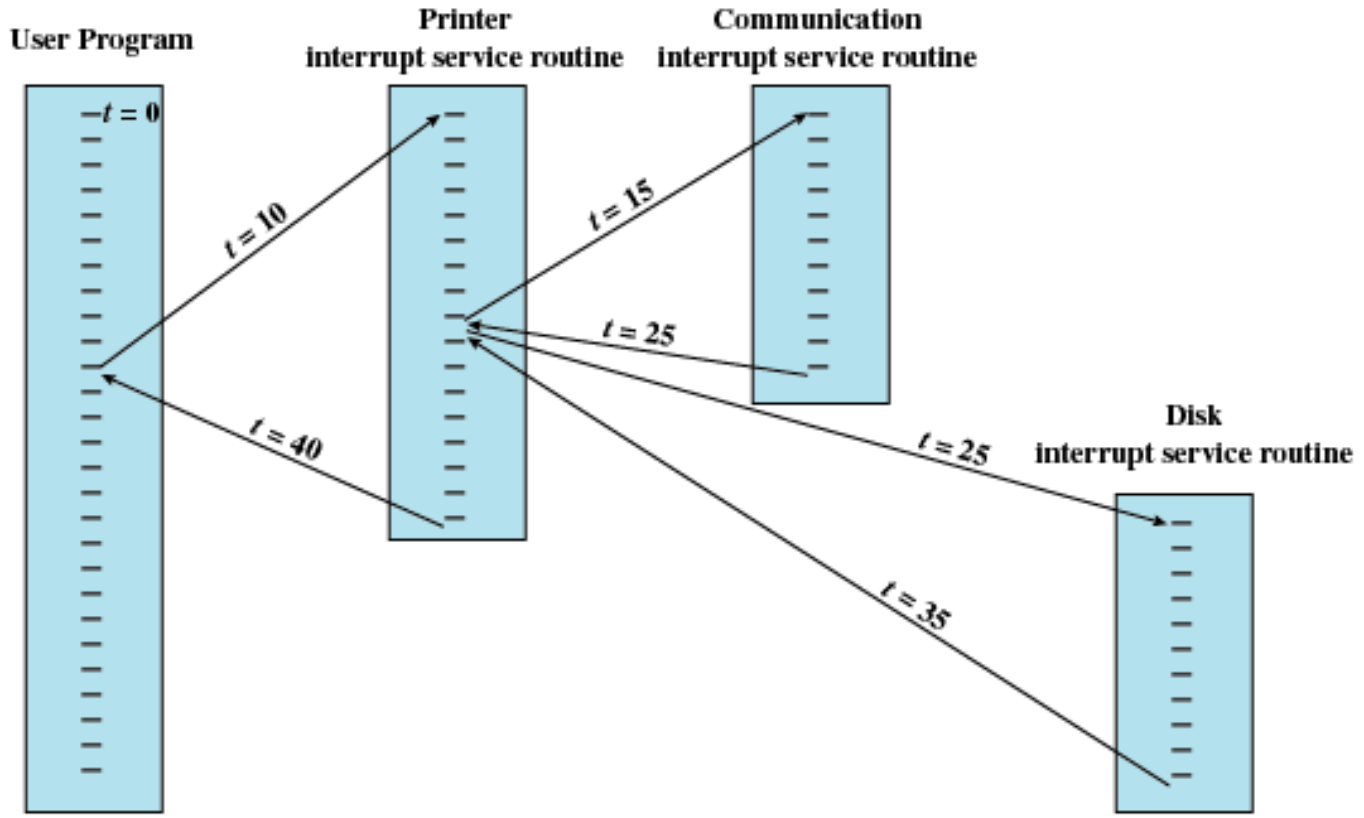


Figure 1.13 Example Time Sequence of Multiple Interrupts

Multiprogramming

- Processor has more than one program to execute
- The sequence the programs are executed depend on their relative priority and whether they are waiting for I/O

Memory Hierarchy

- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access frequency

Memory Hierarchy

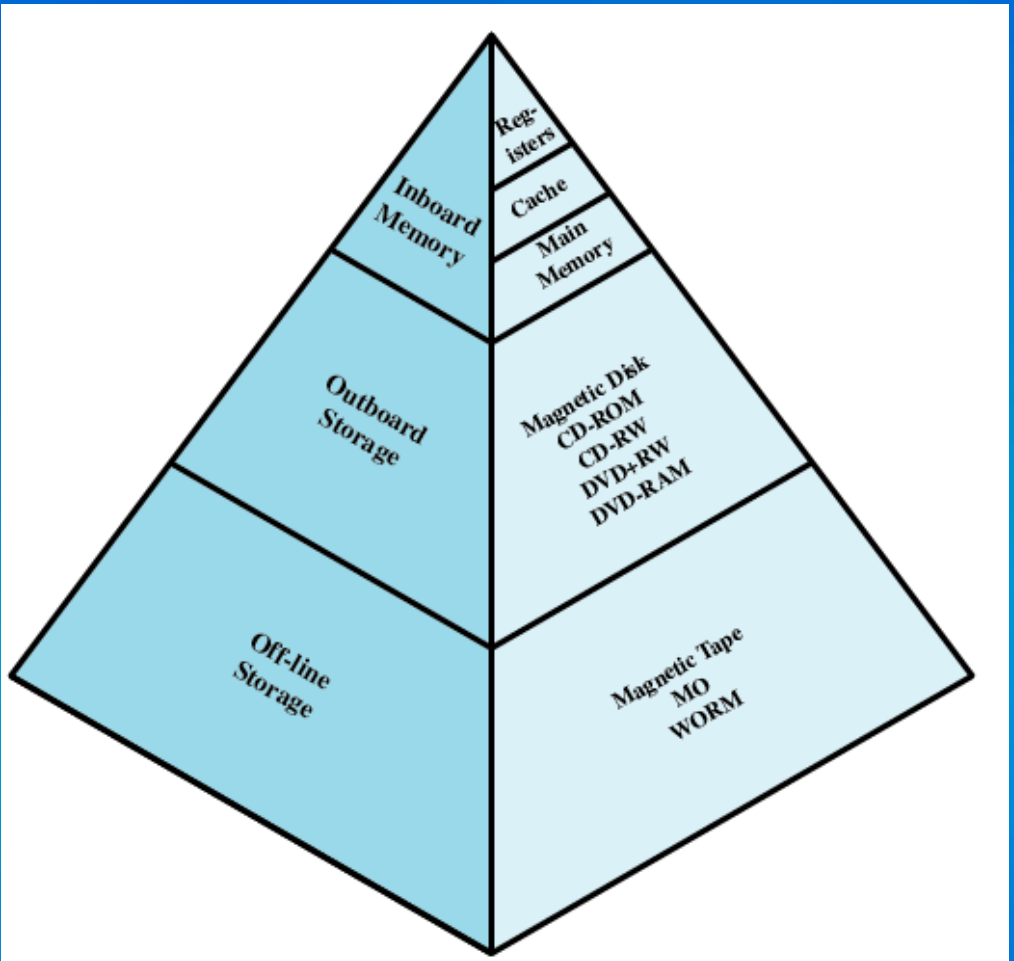


Figure 1.14 The Memory Hierarchy

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

Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

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

Secondary Memory

- Nonvolatile
- Auxiliary memory
- Used to store program and data files

Disk Cache

- A portion of main memory used as a buffer to temporarily to hold data for the disk
- Some data written out may be referenced again. The data are retrieved rapidly from the software cache instead of slowly from disk

Cache Memory

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

Cache Memory

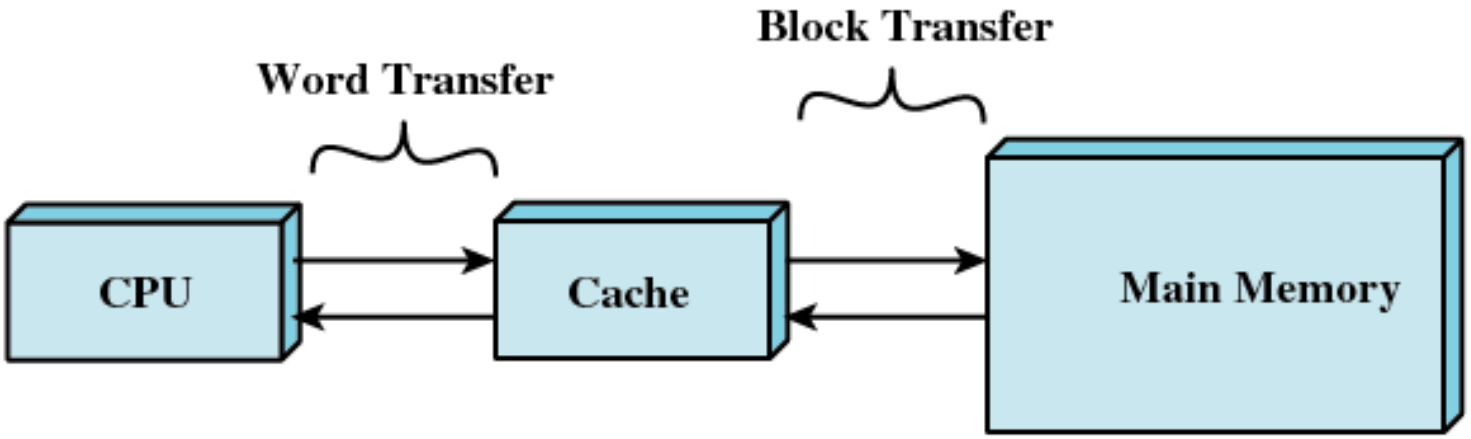


Figure 1.16 Cache and Main Memory

Cache Memory

- Contains a copy of a portion of main memory
- Processor first checks cache
- If not found in cache, the block of memory containing the needed information is moved to the cache and delivered to the processor

Cache/Main Memory System

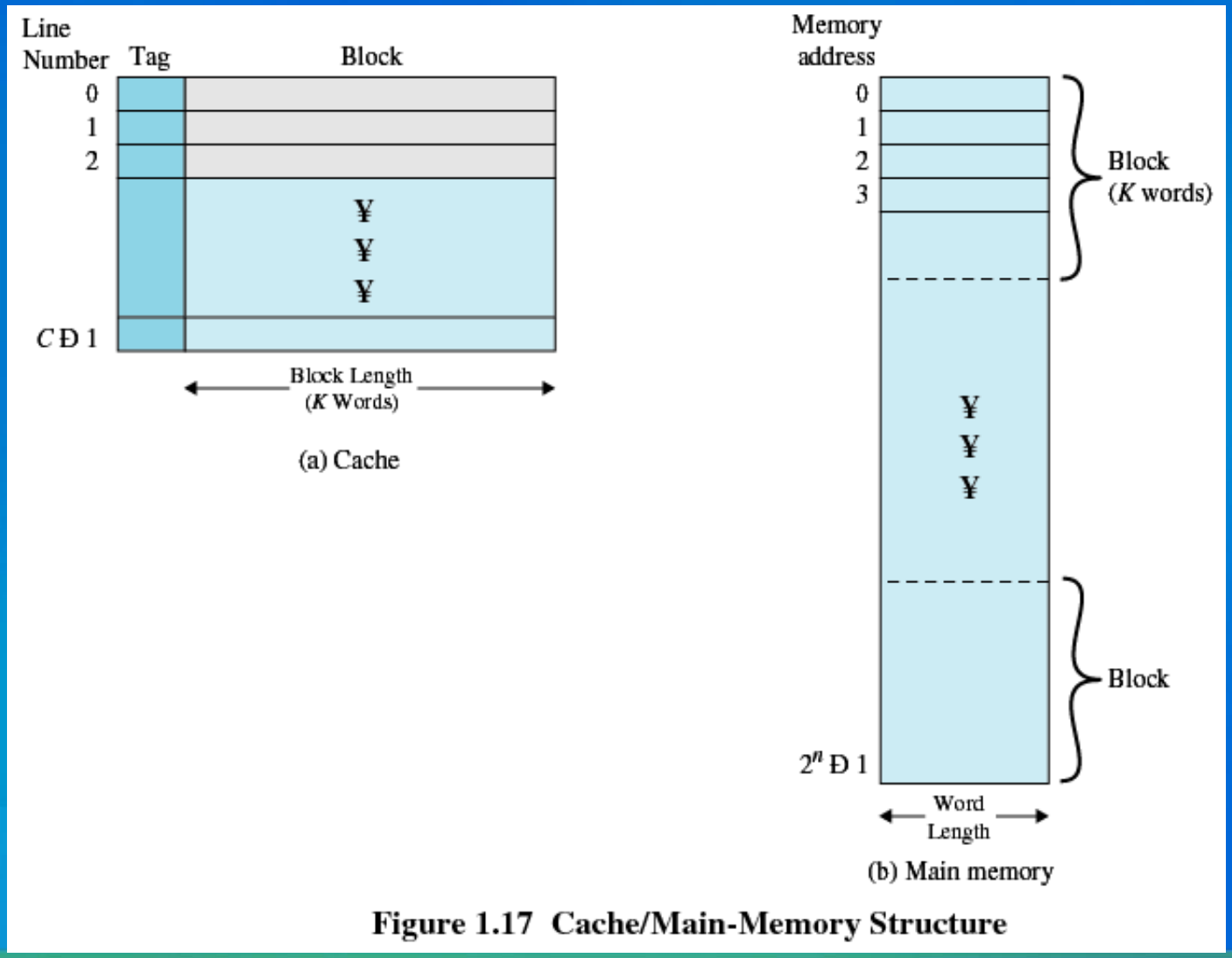


Figure 1.17 Cache/Main-Memory Structure

Cache Read Operation

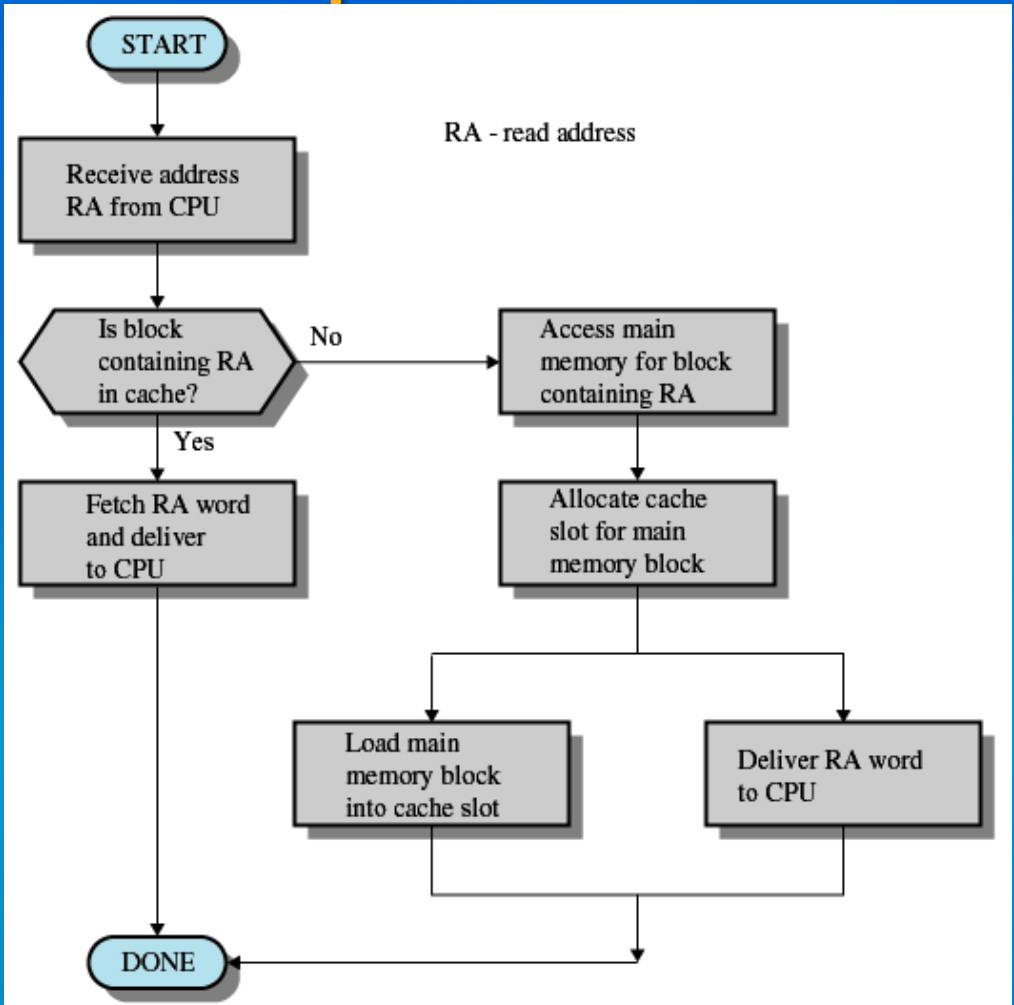


Figure 1.18 Cache Read Operation

Cache Design

- Cache size
  - Small caches have a significant impact on performance
- Block size
  - The unit of data exchanged between cache and main memory
  - Larger block size more hits until probability of using newly fetched data becomes less than the probability of reusing data that have to be moved out of cache

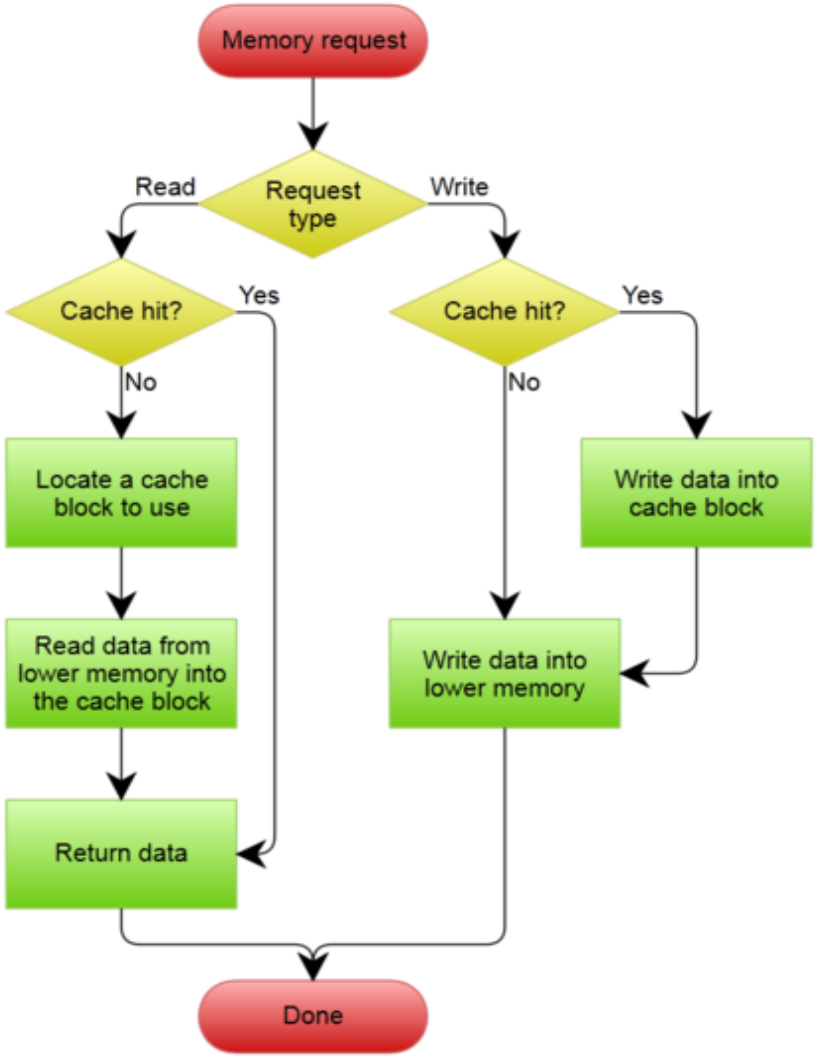
Cache Design

- Mapping function
  - Determines which cache location the block will occupy
- Replacement algorithm
  - Determines which block to replace
  - Least-Recently-Used (LRU) algorithm

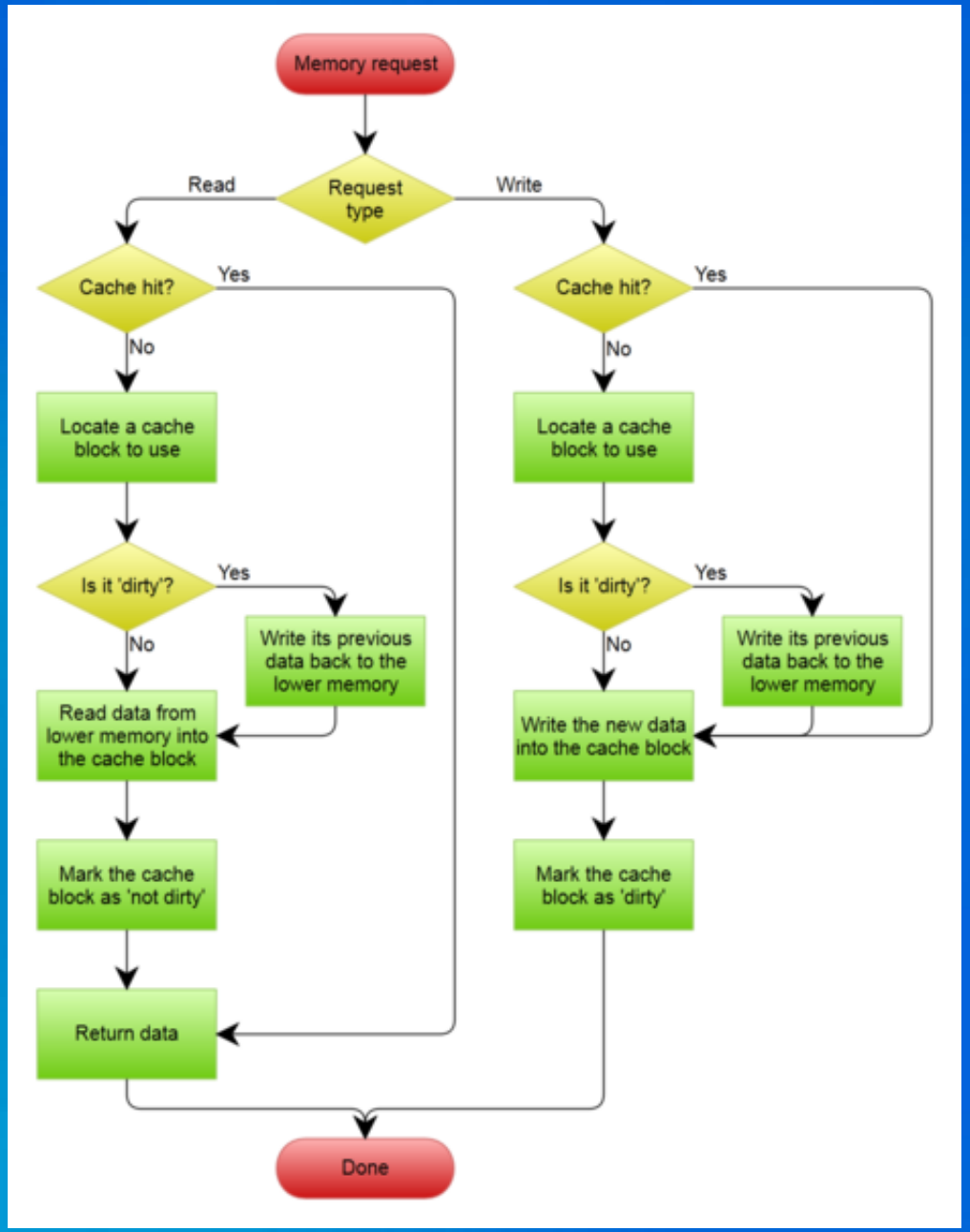
Cache Design

- Write policy
  - 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

Write Polices



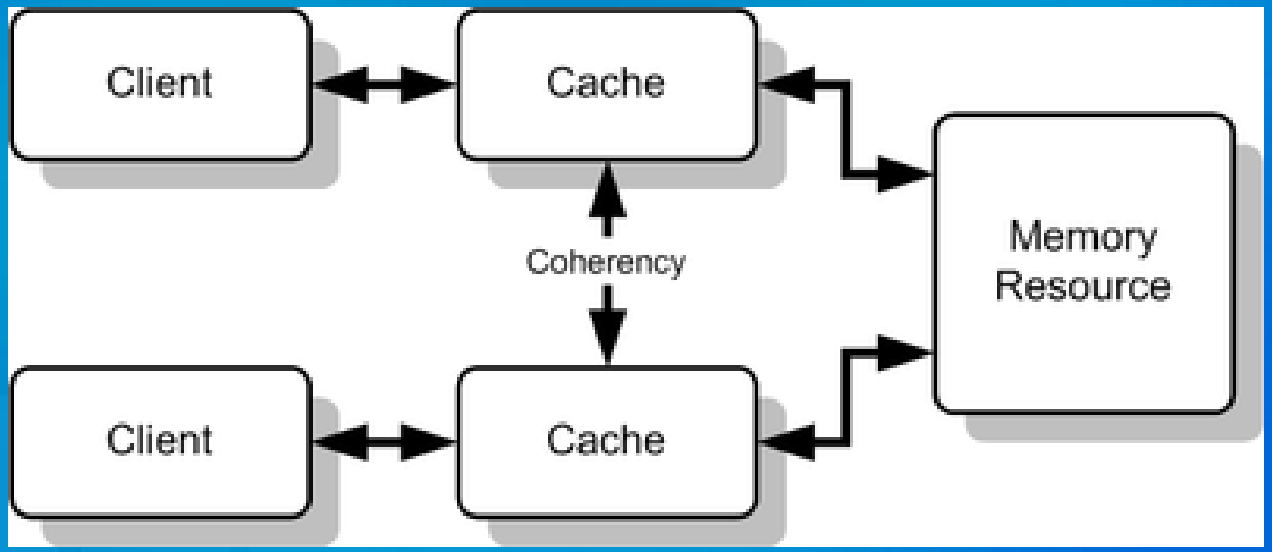
Write Through with No write Allocation



Write Back with write Allocation

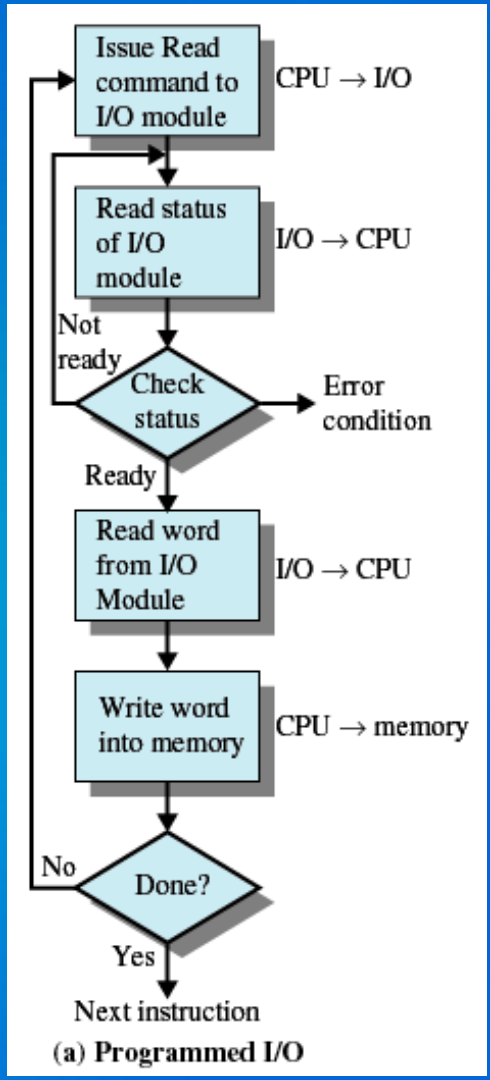
Cache Coherence

- When clients in a system maintain caches of a common memory resource, problems may arise with inconsistent data. This is particularly true of CPUs in a multiprocessing system. Referring to the "Multiple Caches of Shared Resource" figure, if the top client has a copy of a memory block from a previous read and the bottom client changes that memory block, the top client could be left with an invalid cache of memory without any notification of the change.
- Cache coherence is intended to manage such conflicts and maintain consistency between cache and memory.
- What may be the Mechanisms to enforce cache coherence? (Assignment)



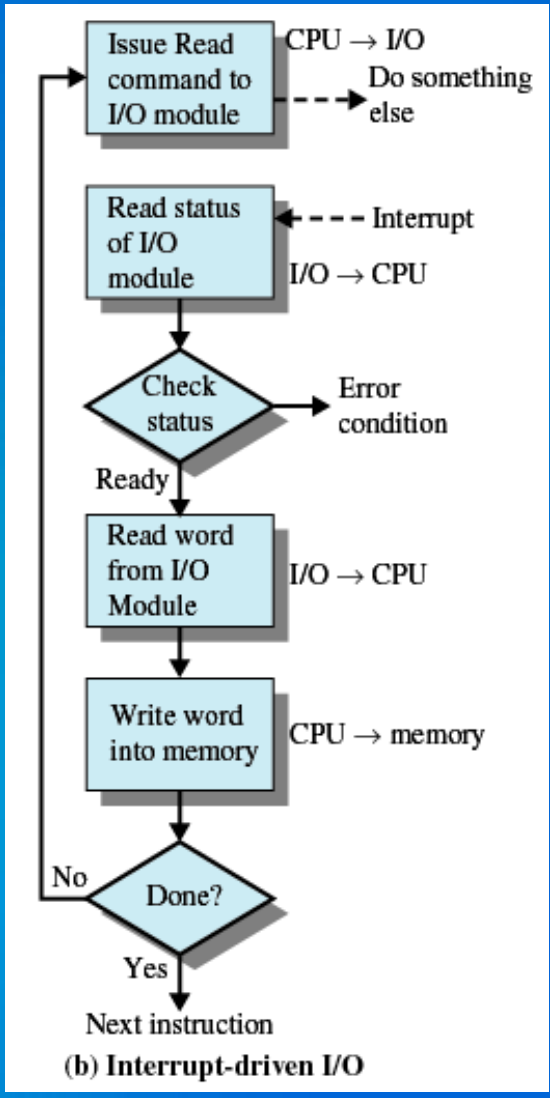
Programmed I/O

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



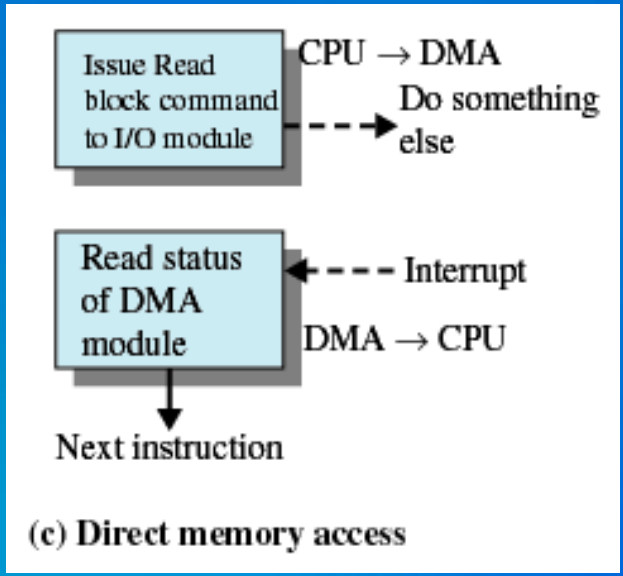
Interrupt-Driven I/O

- 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 or written passes through the processor



Direct Memory Access

- Transfers a block of data directly to or from memory
- An interrupt is sent when the transfer is complete
- Processor continues with other work

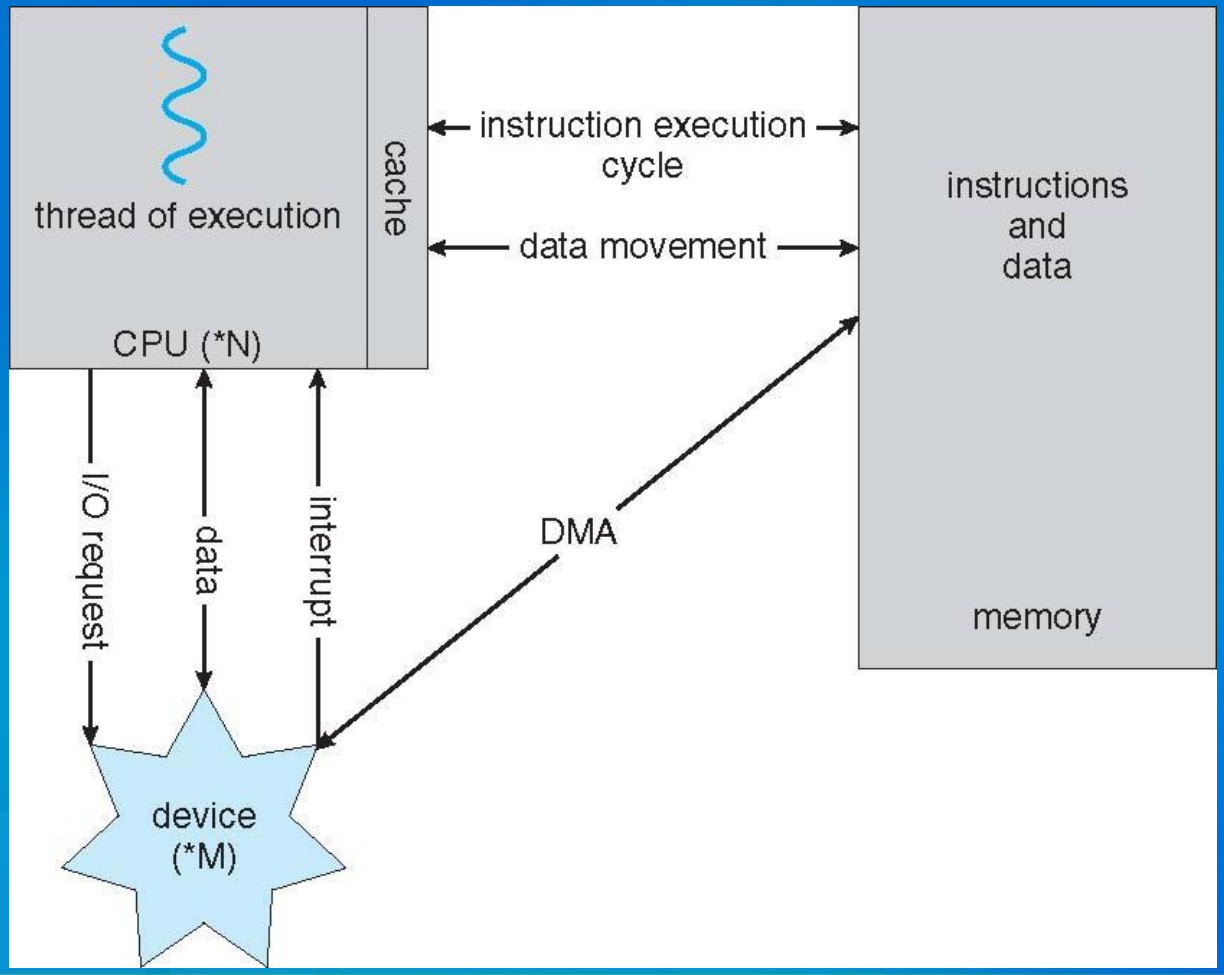


Computer-System Architecture

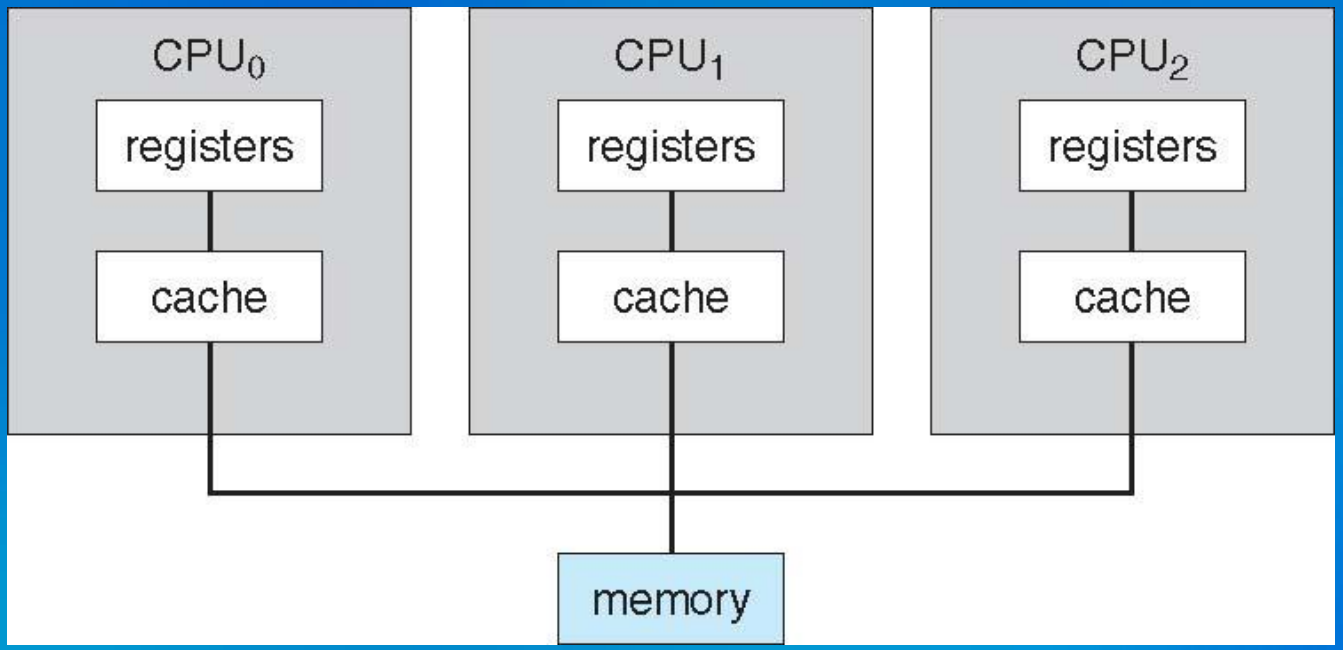
- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- Multiprocessors systems growing in use and importance
  - Also known as parallel systems, tightly-coupled systems

    Common Clock, memory, bus, I/0 devices
  - Advantages include
    1. Increased throughput
    2. Economy of scale
    3. Increased reliability - graceful degradation or fault tolerance
  - Two types
    1. Asymmetric Multiprocessing
    2. Symmetric Multiprocessing

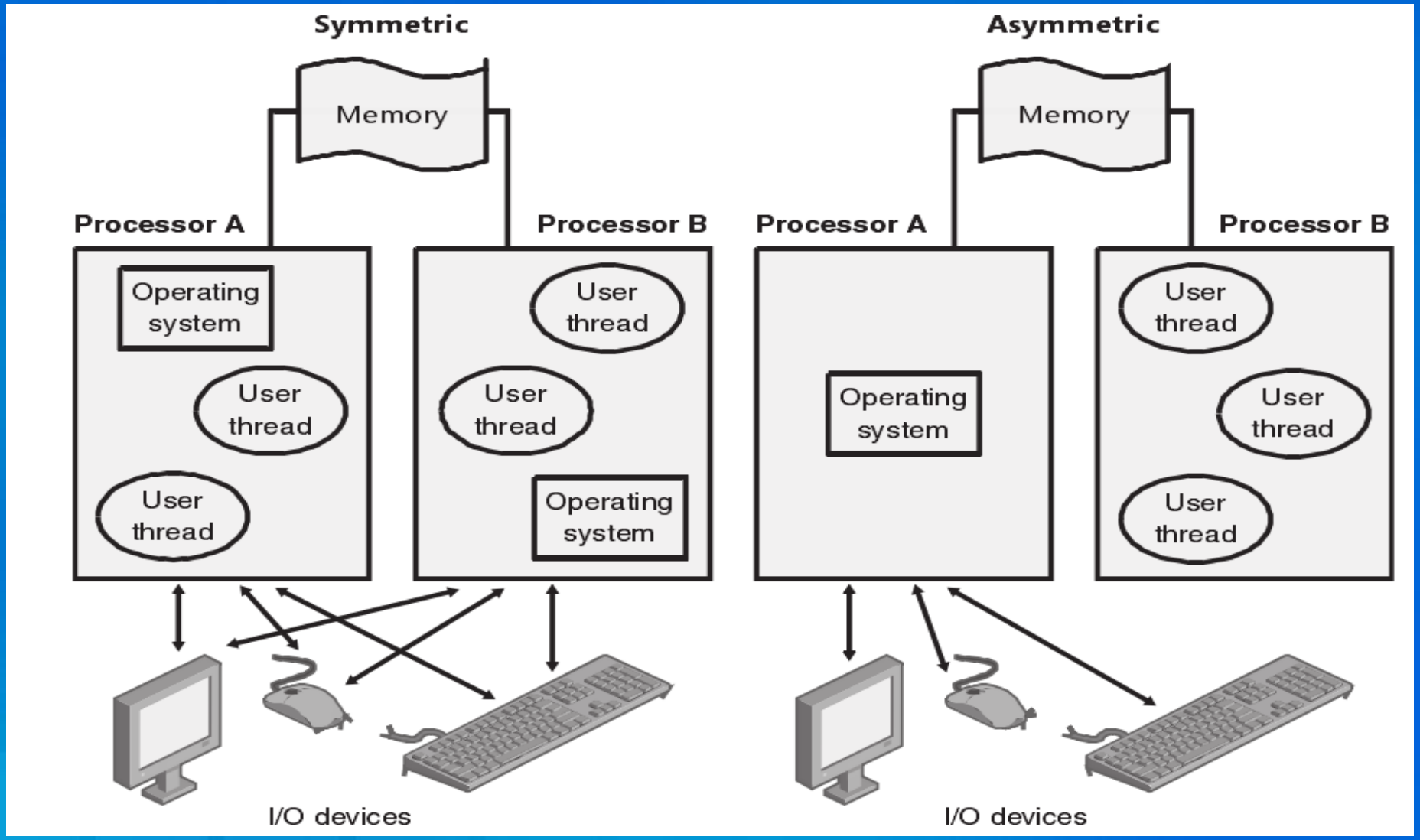
How a Modern Computer Works



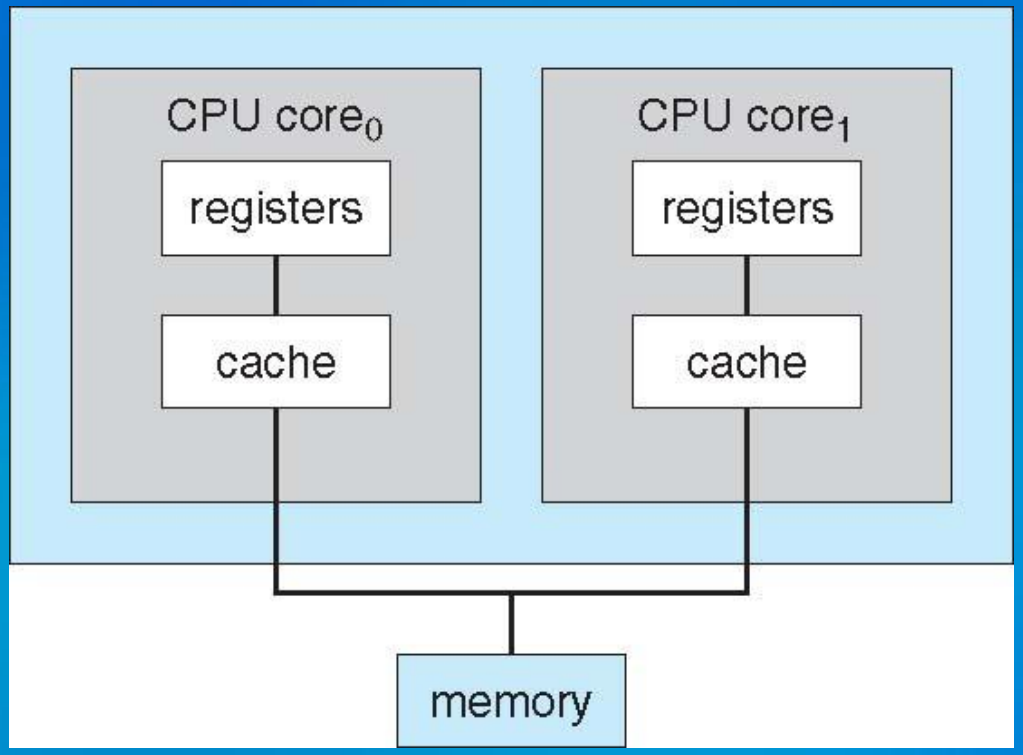
Symmetric Multiprocessing Architecture



Symmetric vs Asymmetric multiprocessing



A Dual-Core Design



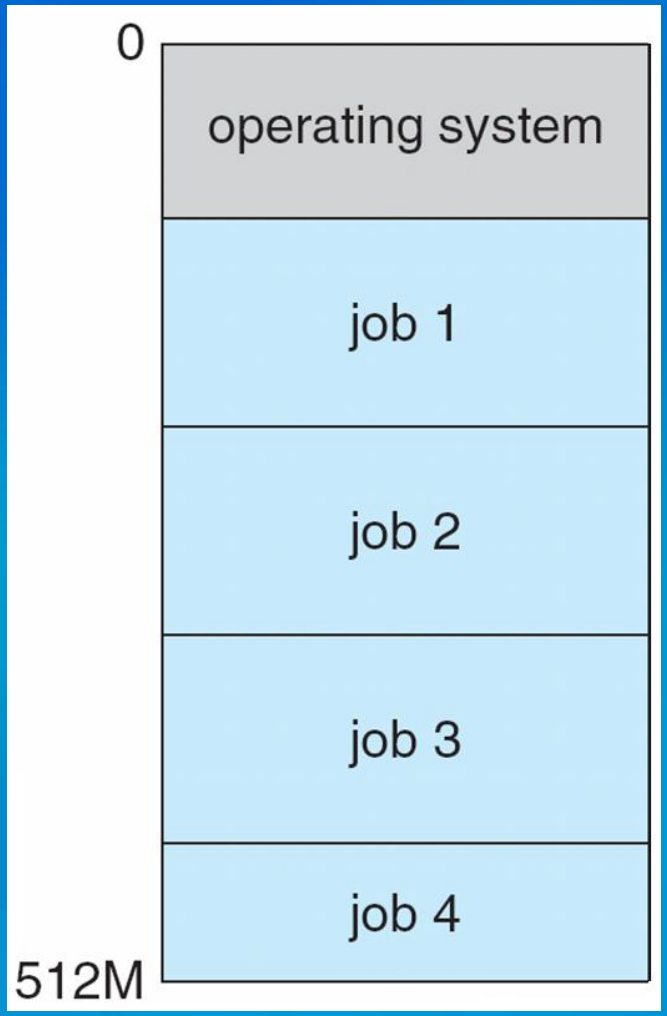
Clustered Systems

- 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

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

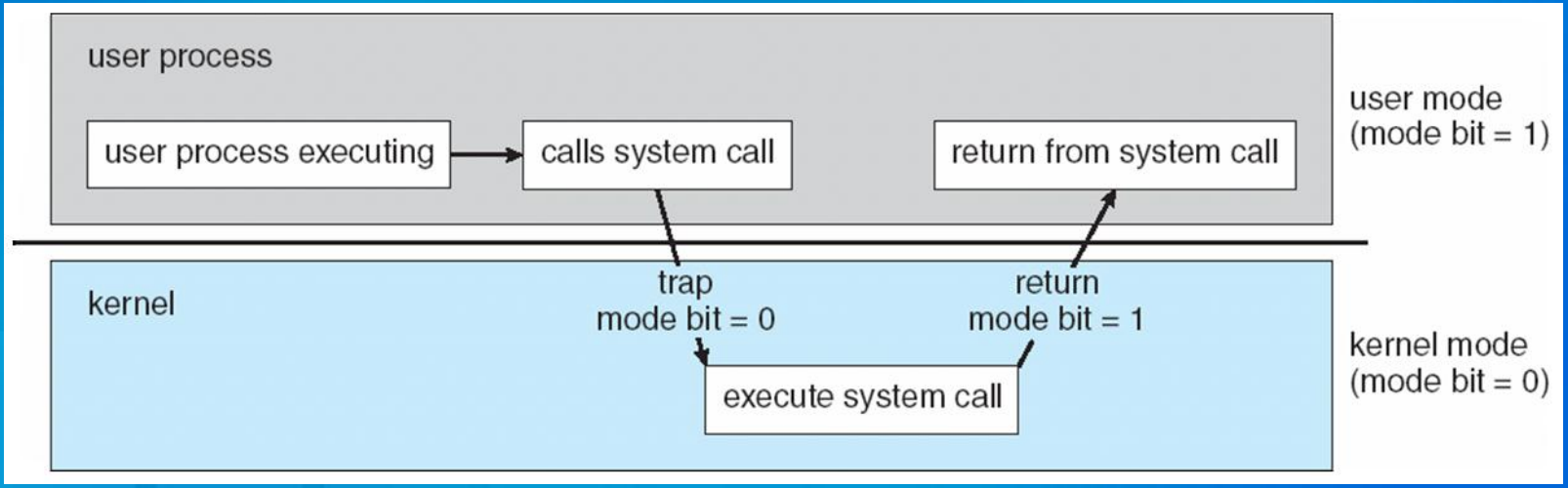


Operating-System Operations

- 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 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
    - 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

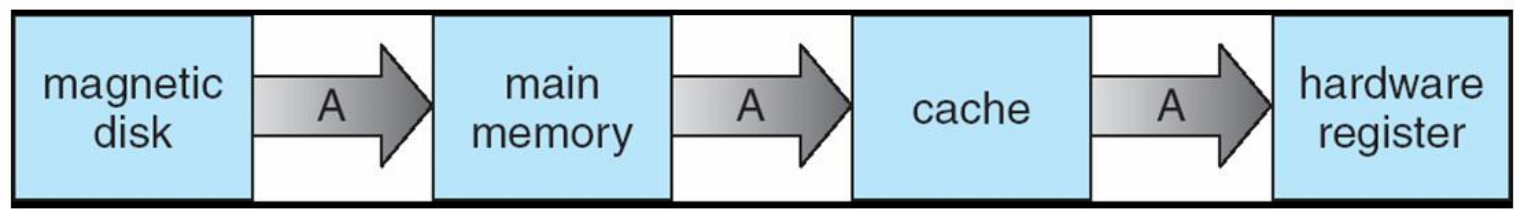
Transition from User to Kernel Mode

- 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



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
  - Various solutions covered in Chapter 17

Protection and Security

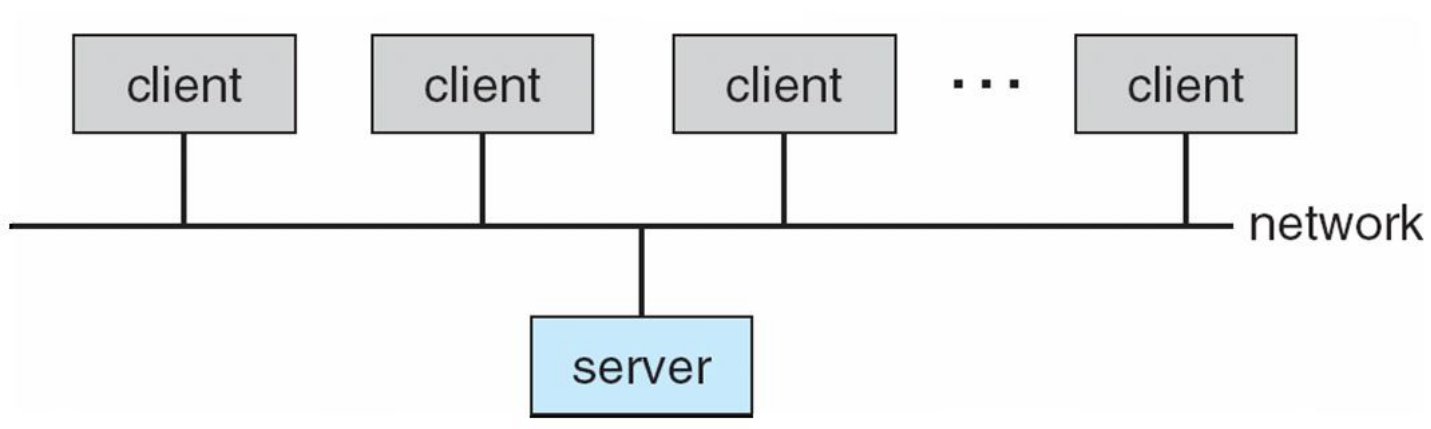
- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first 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

Computing Environments

- Traditional computer
  - Blurring over time
  - Office environment
    - PCs connected to a network, terminals attached to mainframe or minicomputers providing batch and timesharing
    - Now portals allowing networked and remote systems access to same resources
  - Home networks
    - Used to be single system, then modems
    - Now firewalled, networked

Computing Environments (Cont)

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server** provides an interface to client to request services (i.e. database)
    - **File-server** provides interface for clients to store and retrieve files



Peer-to-Peer Computing

- 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*

Web-Based Computing

- Web has become ubiquitous
- PCs most prevalent devices
- More devices becoming networked to allow web access
- New category of devices to manage web traffic among similar servers: **load balancers**
- Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows XP, which can be clients and servers

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, BSD UNIX (including core of Mac OS X), and Sun Solaris