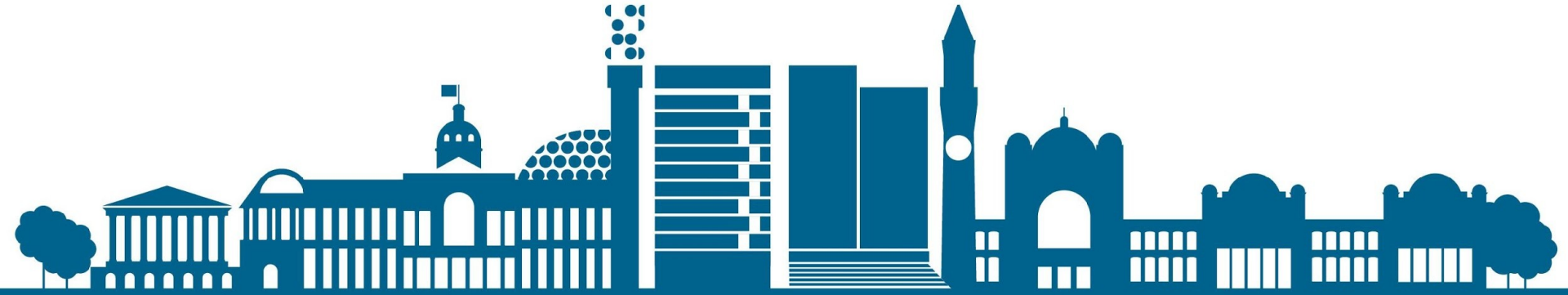
UNIVERSITY OF BIRMINGHAM

Computer Systems
Computer Organization and Architecture



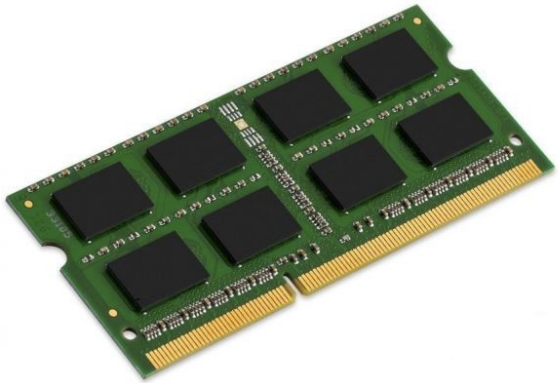
Lecture Overview

In this lecture, we shall see:

- Storage Structure – Memory Hierarchy
- I/O Structure – How the I/O devices function?
- I/O Mechanisms (Polling, Interrrupt Driven, DMA)
- Some General System Architectures

Storage Structure – Main Memory

- CPU can load instructions only from memory
  - Any programs to run must be stored there
- Memory is generally rewritable
  - Main memory or (random-access memory)
  - RAM is commonly implemented in a semiconductor technology called Dynamic RAM (DRAM)



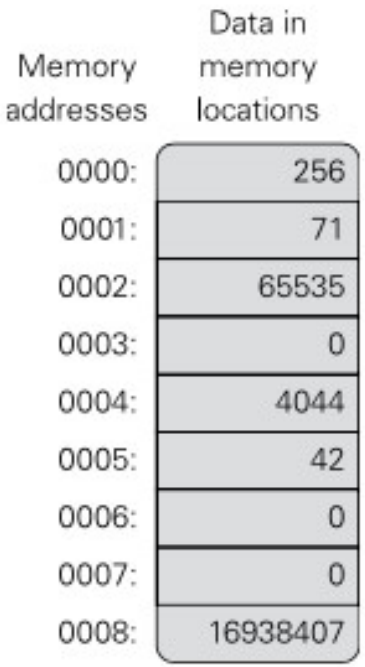
Storage Structure – Other forms of memory

- Read-only Memory (ROM)
- Electrically Erasable Programmable ROM (EEPROM)
- ROM cannot be modified
  - Suitable for bootstrap programs and game cartridges(!)
- EEPROM can only be changed infrequently
  - Most smartphones store factory-bundled programs on EEPROM

Storage Structure – How it works?

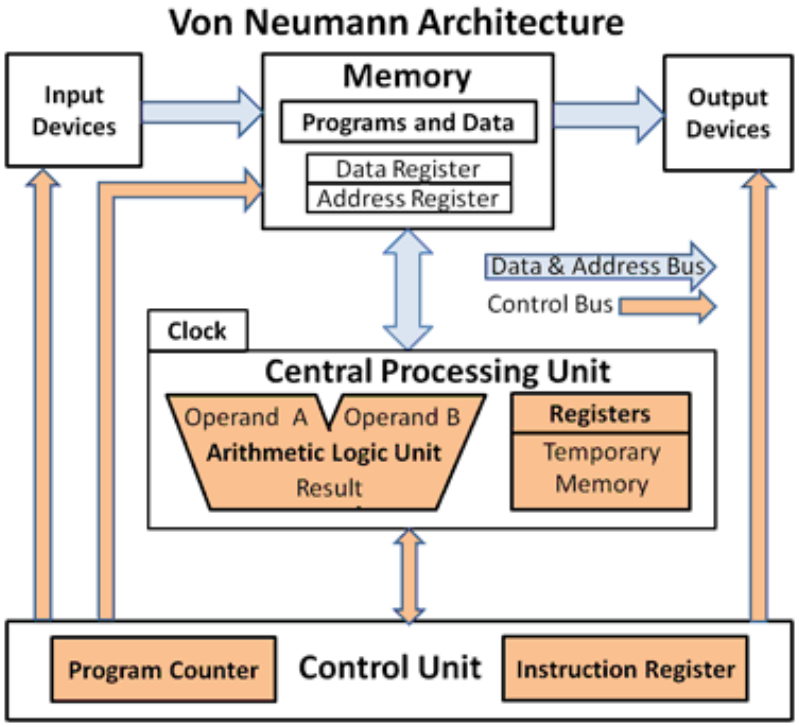
- Memory is an array of bytes
  - Each byte has its own address
  - Memory interaction via load/store instructions
- **Load:**
  - Moves a byte from main memory to an internal register within the CPU
- **Store:**
  - Writing of a byte from CPU register to main memory

  …
- CPU automatically loads instructions from main memory for execution



Von Neumann architecture

- Fetch instruction – store in **instruction register**
  - This is **decoded**, may cause further operands to be fetched and stored in another register
  - Instruction is the **executed** – result stored back into memory
- Memory unit only sees a stream of addresses



In the perfect world …

Volatile: when you switch off computer, the memory in main memory will be gone

- Programs and data would reside in main memory
  - This is difficult as the Main memory is **small** and **volatile**.
- **Solution**: Secondary storage which includes magnetic disks, optical disks, tapes.
  - Programs stored and loaded from here
  - Many programs use the disk as both source and destination of processing
- Cache/CD-ROM/magnetic tapes other forms of storage system
  - Differences lie in **speed**, **cost**, **size** and **volatility**

Storage Comparison

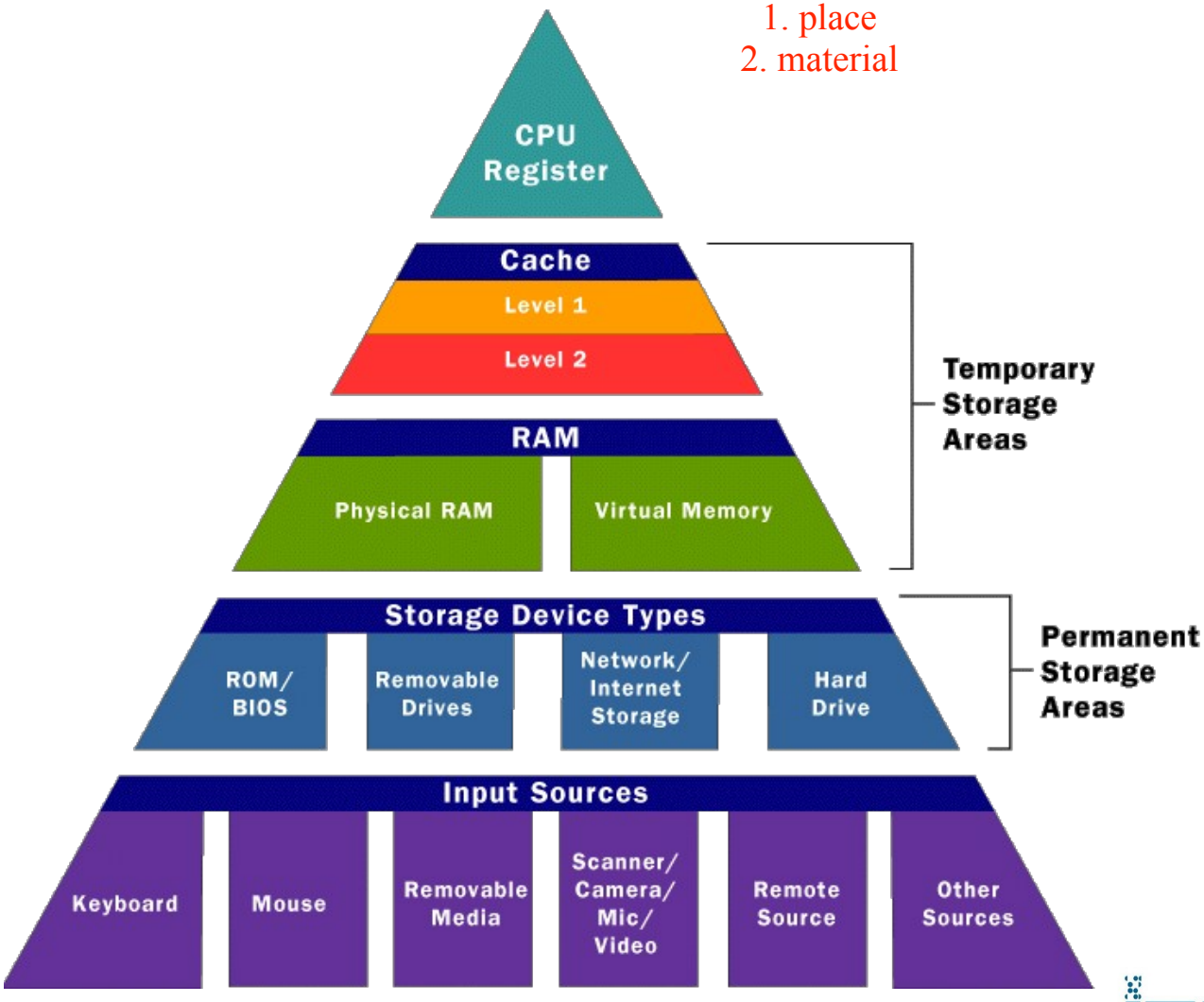
Fastest memory

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

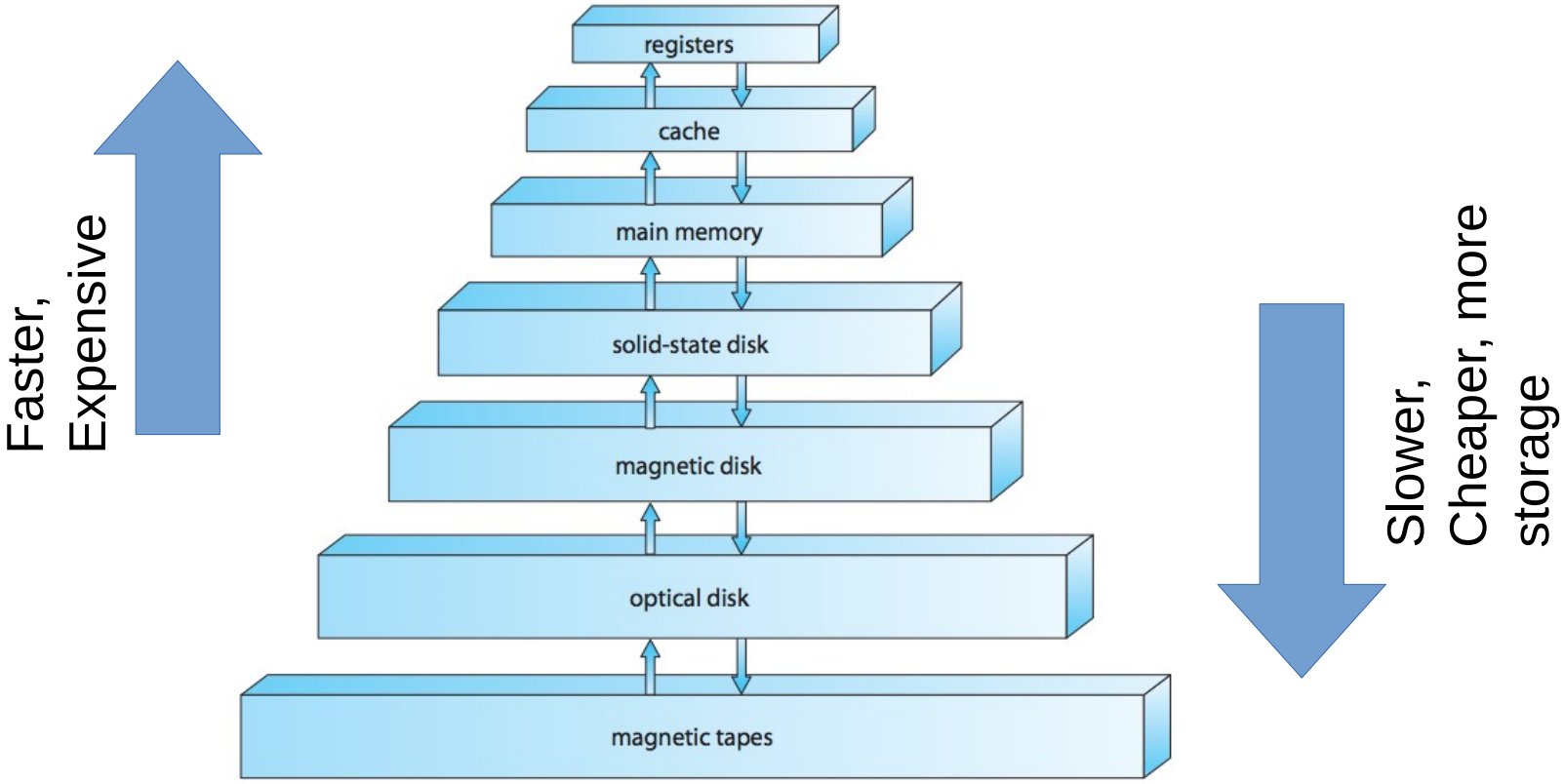
Table of relative instances of memory and their associated speeds from Silberschatz et al (2012)

Storage Hierarchy

Register processing so fast becuase:
1. place
2. material

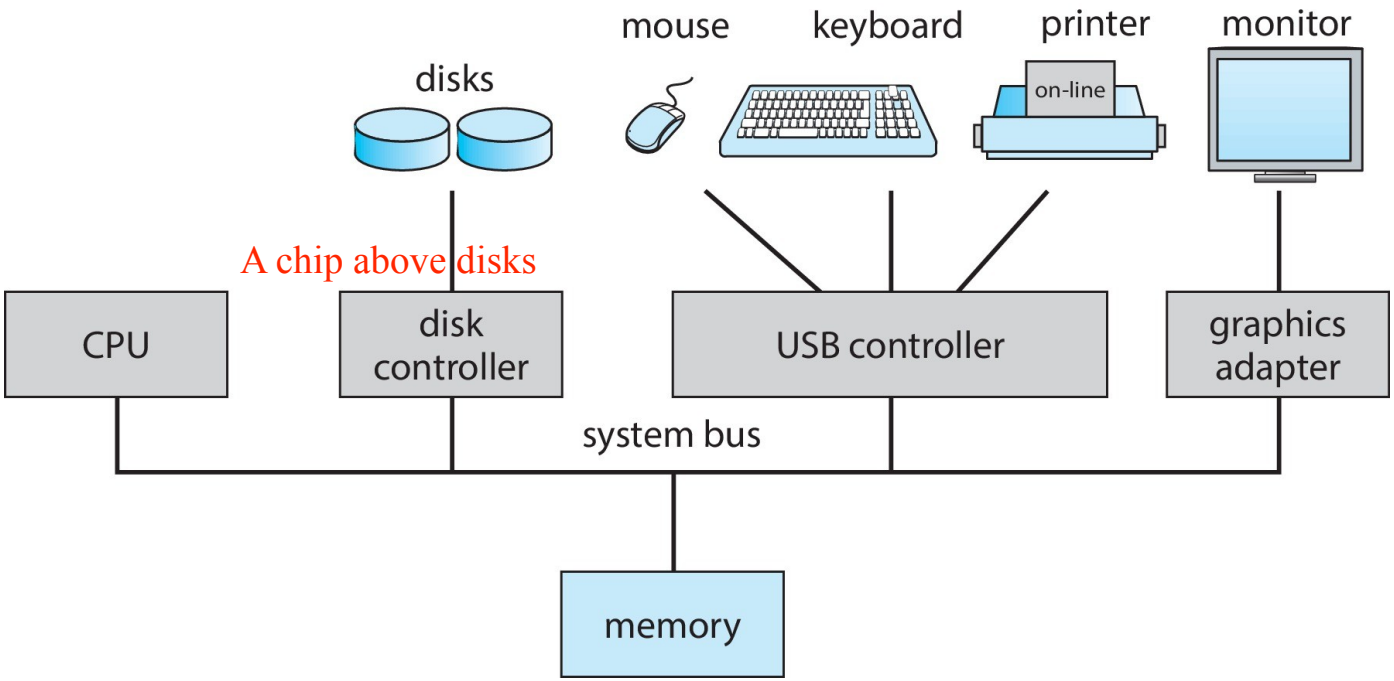


Storage Hierarchy – another view



I/O Structure

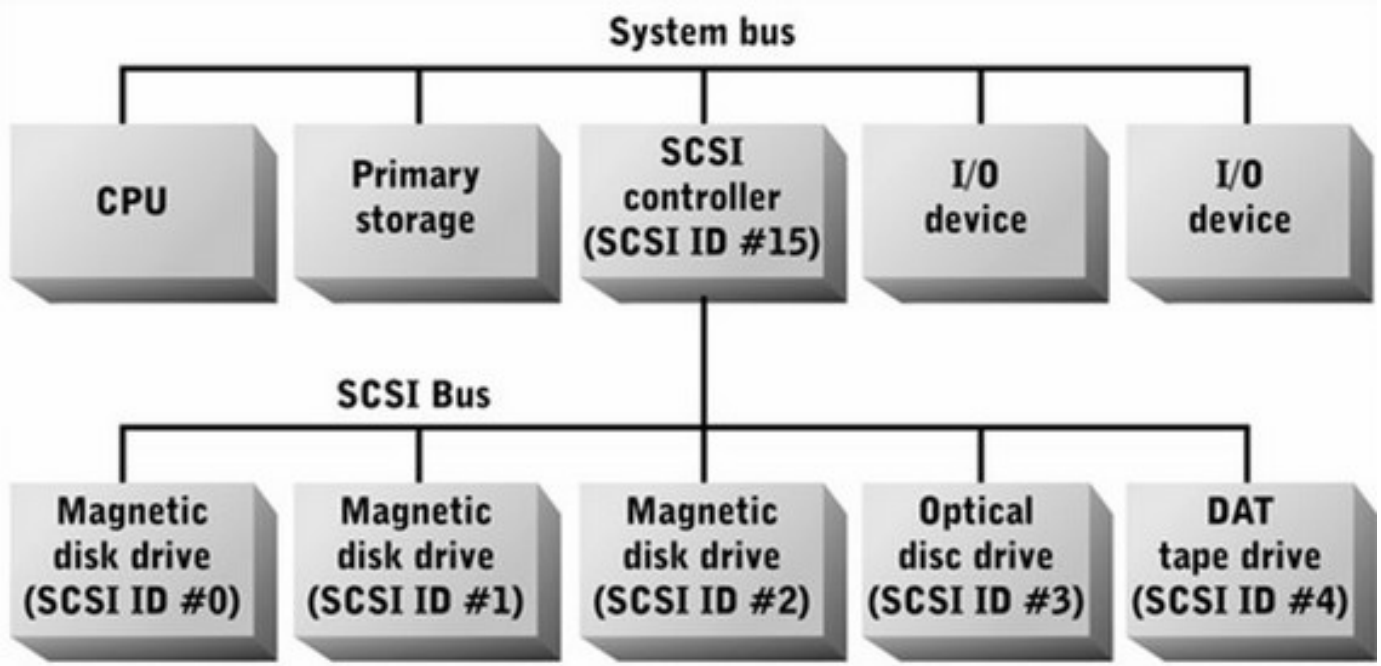
- Large portion of OS code dedicated to managing I/O
- Why is this important?



How does I/O Work?

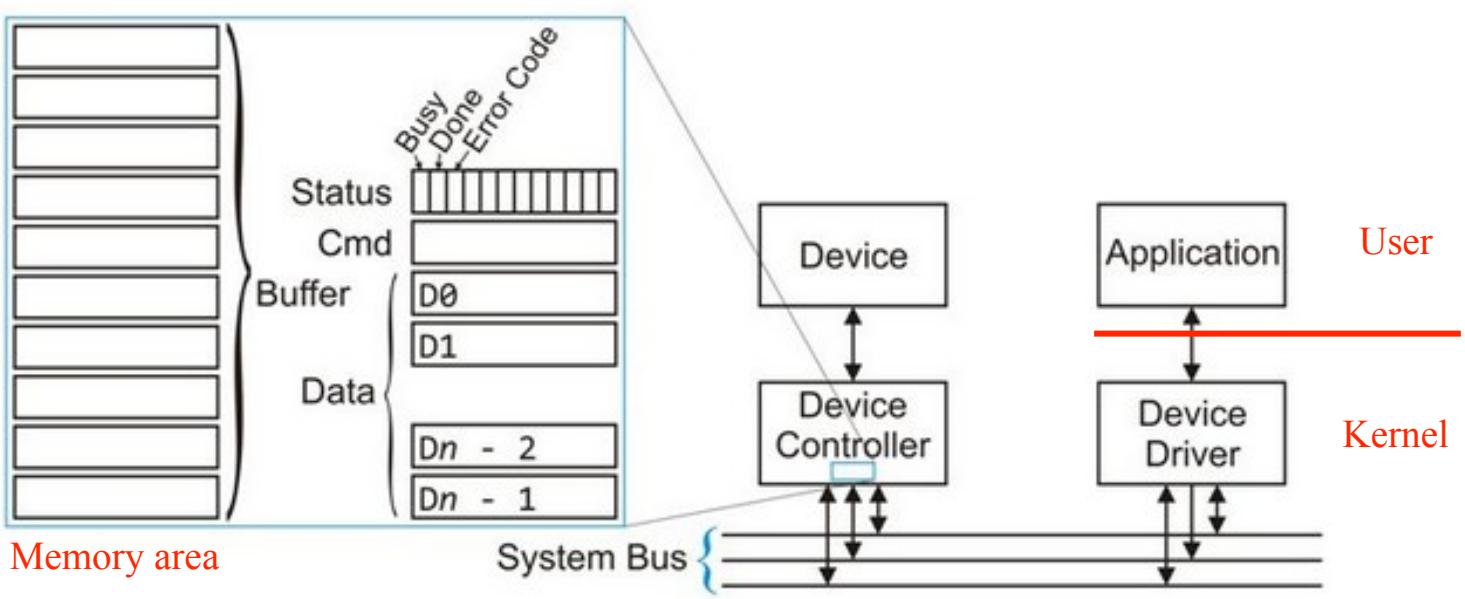
- **Recall:** hardware may trigger an interrupt at any time by sending a signal to the CPU.
- **Devices** interact via a **device controller** connected through a common **bus** to the CPU
- Small Computer-Systems Interface (SCSI) controller
  - Hardware (card or chip based) that allows SCSI storage device to communicate with the OS using an SCSI bus

How does I/O Work?



Device Controller

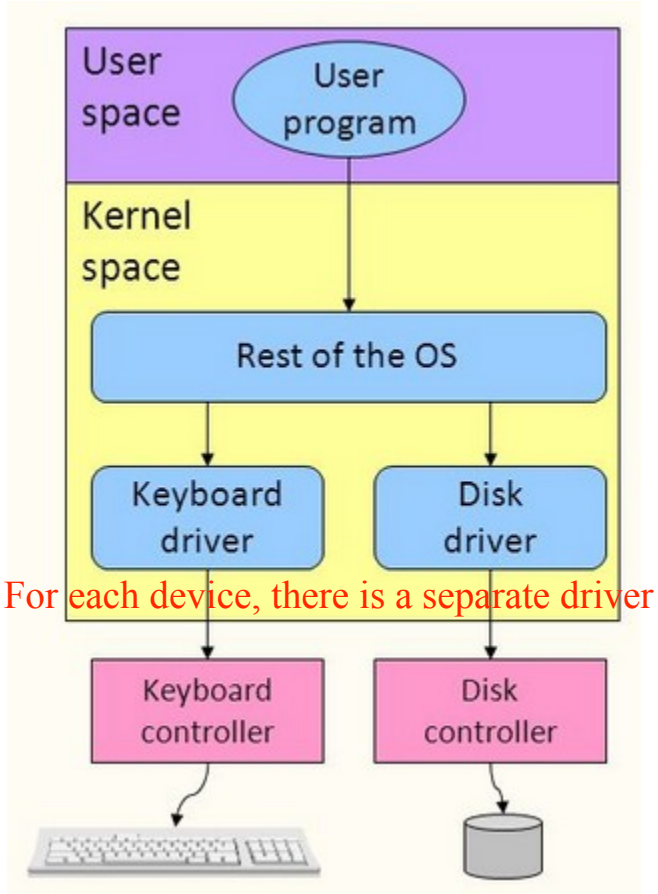
- Maintains some local buffer storage and a set of special-purpose registers
- Device controller moves the data between the peripheral devices that it controls and its local buffer storage



Device Driver

- Operating systems have a device driver for each device controller. These are typically downloaded!
- The device drivers understand the device controller and provides the rest of the OS with a uniform interface to the device. This is the same no matter the device.

in Linux: VFS - Virtual File System



For each device, there is a separate driver

Drives are connected one layer above, higher layer

I/O Mechanisms

Three types of I/O mechanisms are possible:

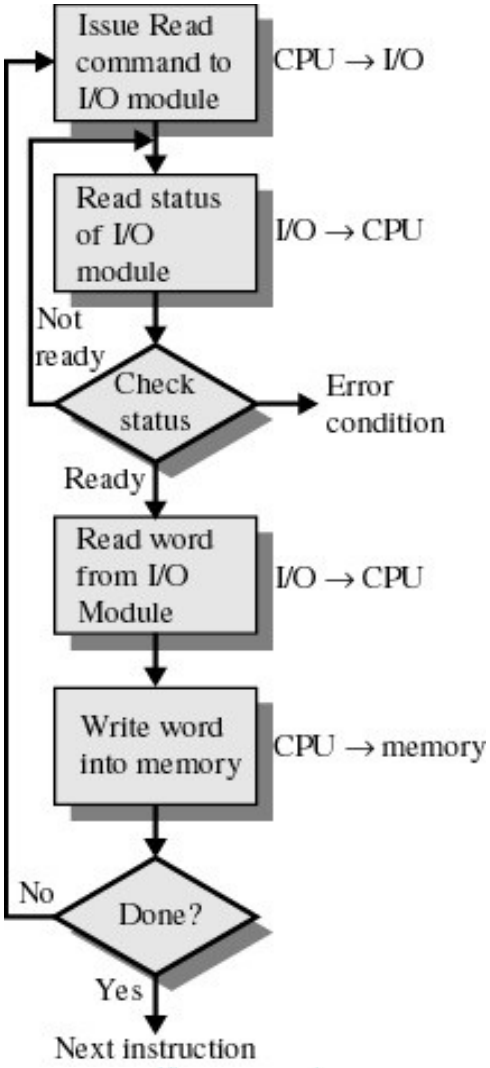
1) **Programmed I/O** (also known as Polling)

2) **Interrupt Driven I/O**

3) **Direct Memory Access (DMA)**

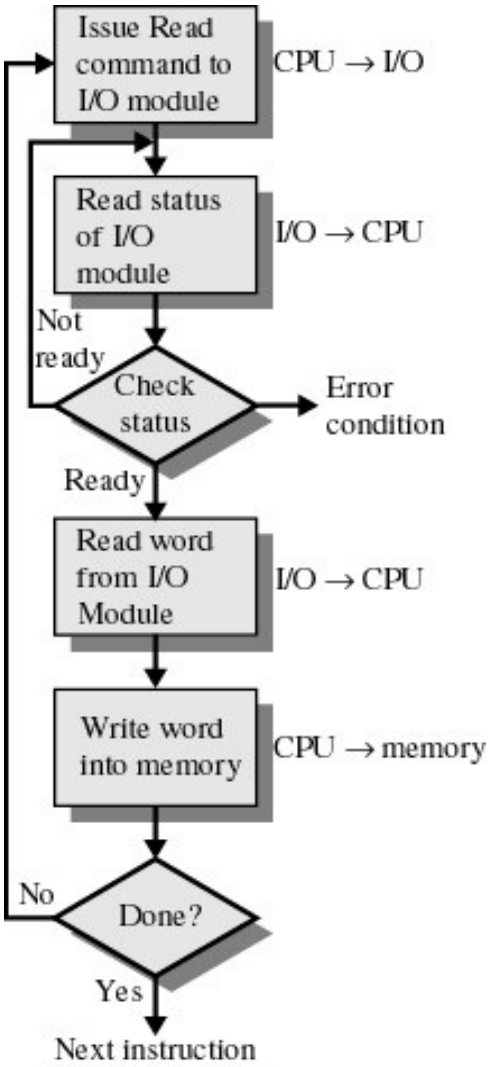
I/O Mechanisms - Programmed I/O

- Programmed I/O requires the processor to "poll" the status of the I/O module (controller)
  - Processor sends the I/O request to I/O module
  - Processor checks, whether I/O module is finished?
  - When I/O finished, data is transferred between I/O modules, processor and main memory.

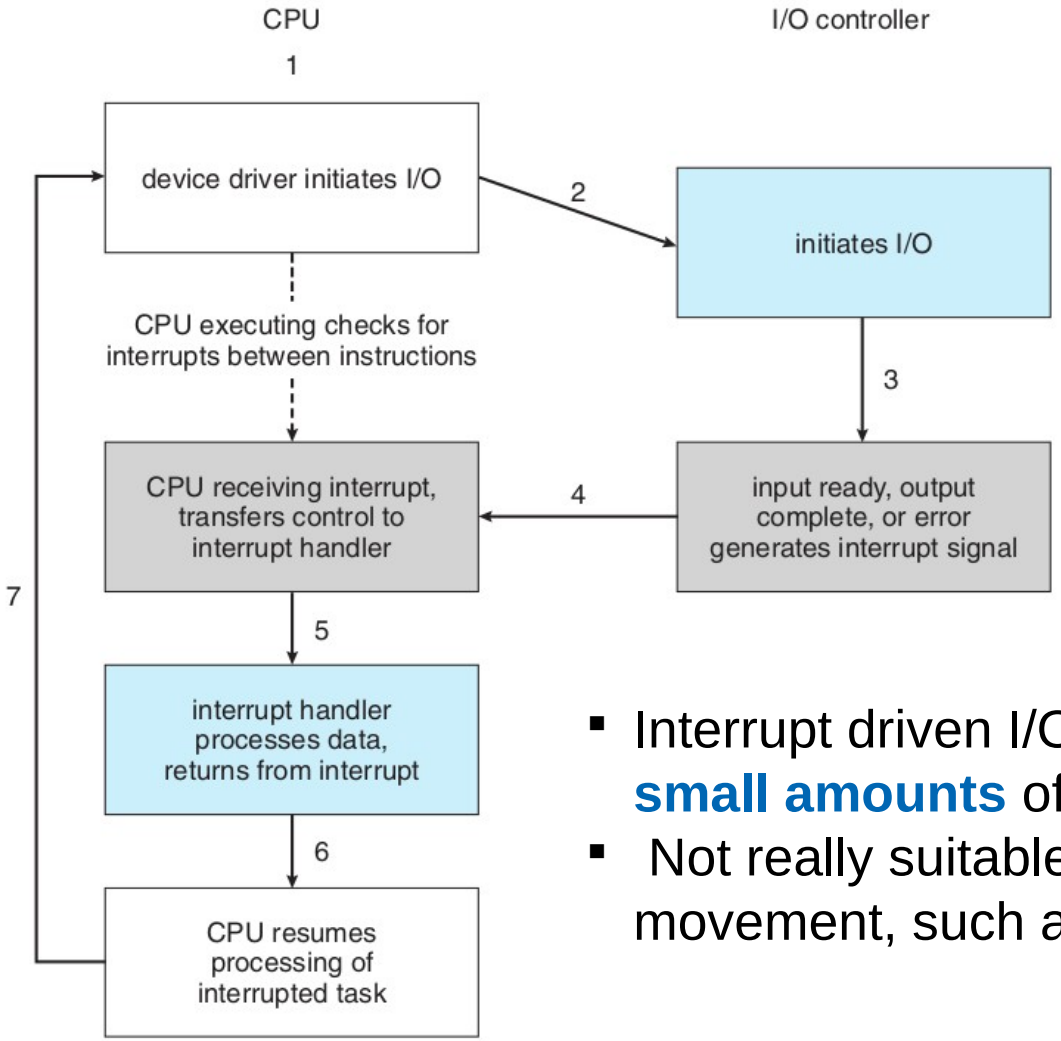


I/O Mechanisms - Programmed I/O

- Processor has special instructions
  - Control I/O device
  - Test status
  - Transfer of data (read/write)
- Performance Issues
  - Processor is busy checking the I/O status of the I/O module



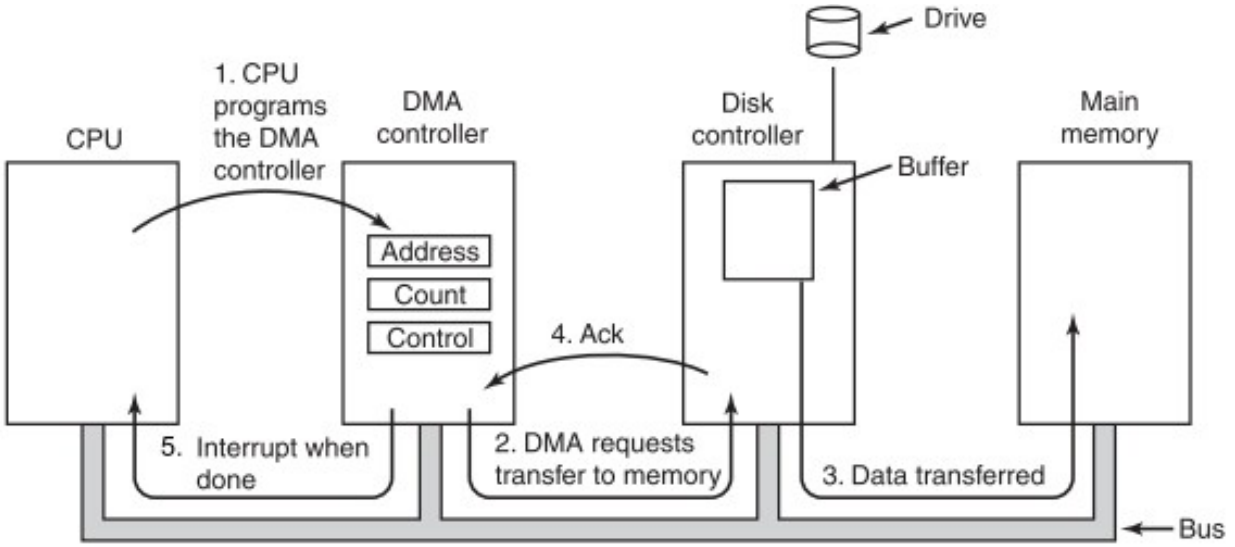
I/O Mechanisms – Interrupt Driven I/O



- Interrupt driven I/O is fine for moving **small amounts** of data
- Not really suitable for **bulk data** movement, such as disk I/O

I/O Mechanisms – Direct Memory Access

- DMA controller transfers data directly from the I/O device to memory
  - Buffers, pointers and counters for the I/O device are setup
  - Device controller transfers an entire block of data directly to or from its own buffer storage to memory.
- No intervention by the CPU during transfer



Why DMA is preferable?

- Only a single interrupt is generated
  - Much better that one interrupt per byte generated for low-speed devices
- So, the CPU is freed up to do other, perhaps more interesting tasks!
- However, both DMA controller and processor share the bus connecting different hardware components.
  - Processor will experience slowdown of processing during DMA transfer, as it has to wait for getting access to the bus.

Computer System Architecture

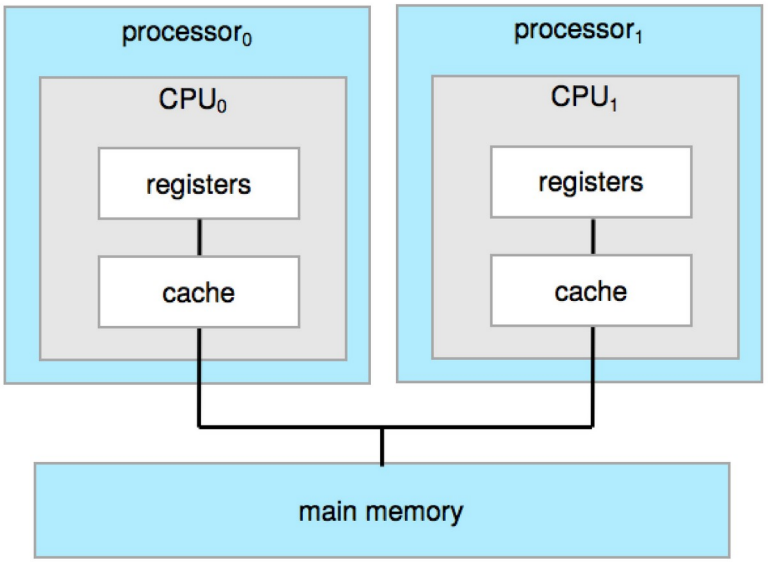
- Single-Processor Systems
  - One CPU – perhaps special purpose processors as well.
- Multiprocessor Systems (Parallel / Multicore)
  - Share bus, clock, memory and peripherals
  - Servers → Smartphones

Multiprocessor Systems – Advantages

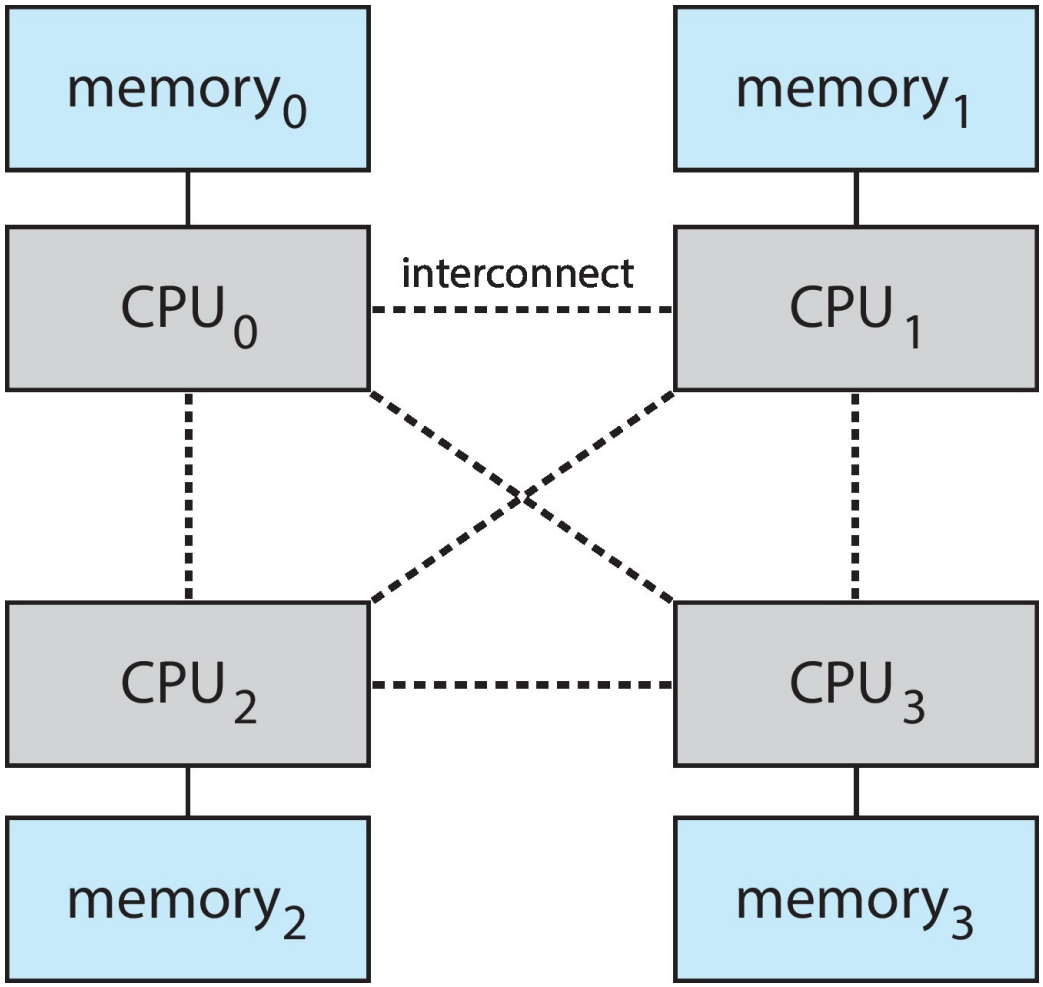
- **Increased throughput**
  - N more processors != Speed-up ratio N (<N)
- **Economy of scale**
  - Cost less – share peripherals, storage, power
- **Increased reliability**
  - Reliable distribution means a failed processor will not stop the whole system – graceful degradation

Types of Multiprocessor Systems

- **Asymmetric**
  - Boss-worker relationship
  - Each processor gets assigned a specific task
- **Symmetric**
  - Peer to peer relationship
  - Each processor is utilised
  - and has equal chances of
  - being used.

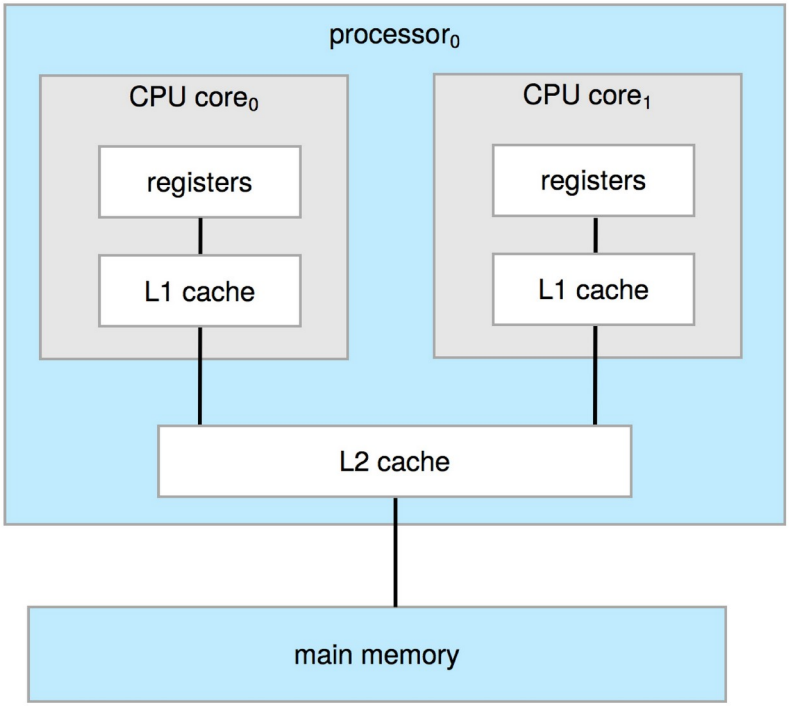


Non-Uniform Memory Access (NUMA)

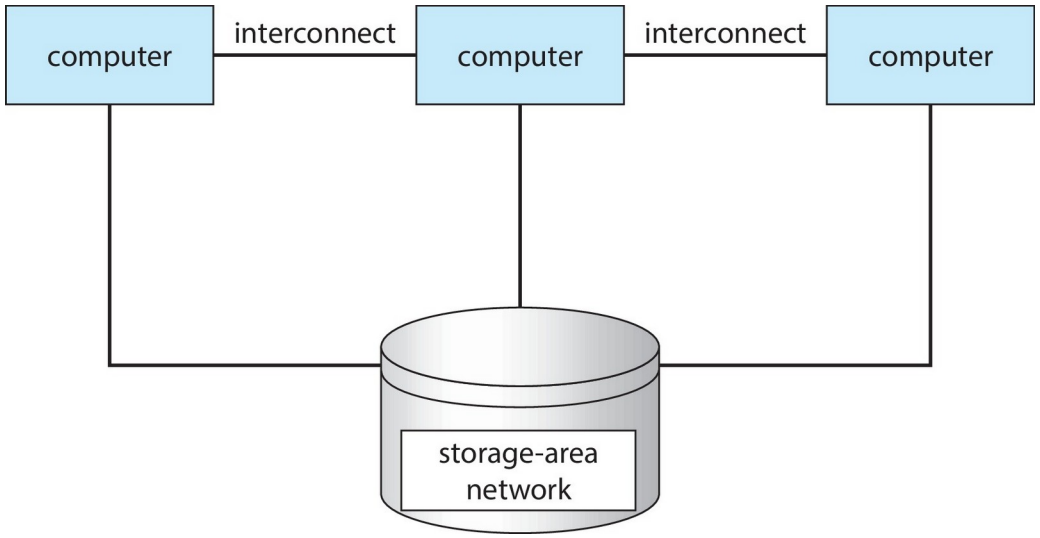


Multicore Systems

- Multiple cores on a single chip
- On-chip communication is faster
- Less power
- Common example:
  - Blade servers

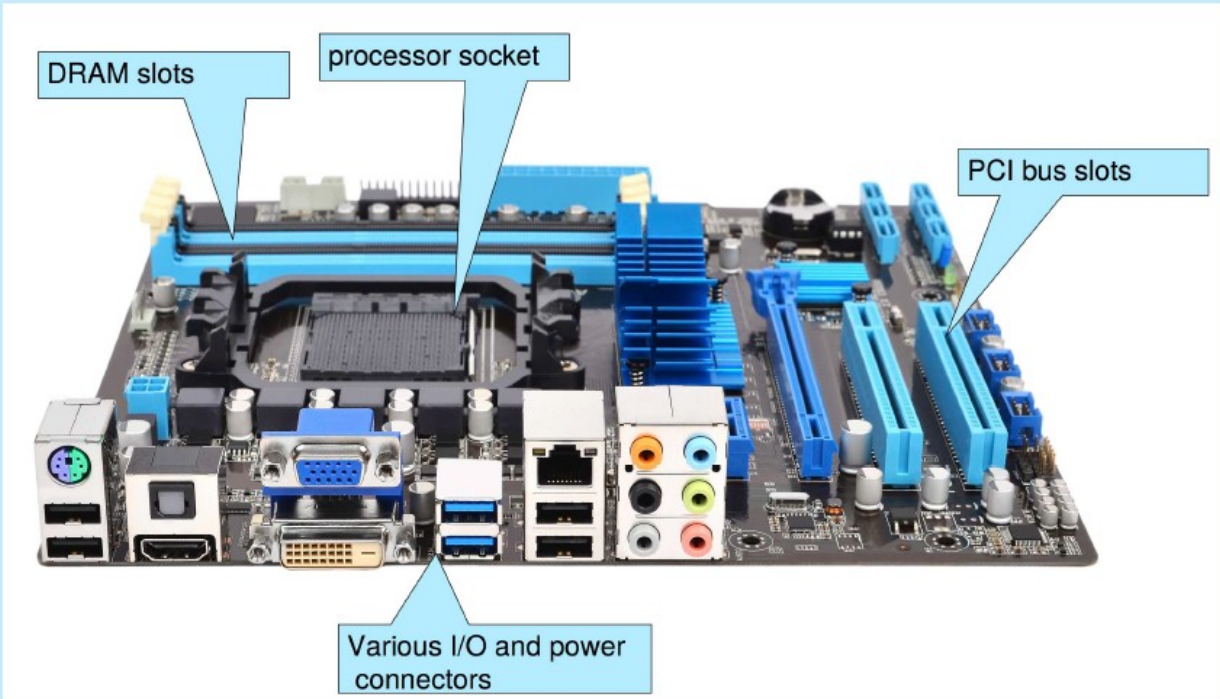


Clustered Systems



- Loosely coupled
- Share storage
- LAN/WAN
- High-availability
- Asymmetric/symmetric – hot-standby
- High-performance computing – BlueBear/Beowulf
- Parallelization of applications

Consider the desktop PC motherboard with a processor socket shown below:



This board is a fully-functioning computer, once its slots are populated. It consists of a processor socket containing a CPU, DRAM sockets, PCIe bus slots, and I/O connectors of various types. Even the lowest-cost general-purpose CPU contains multiple cores. Some motherboards contain multiple processor sockets. More advanced computers allow more than one system board, creating NUMA systems.

Summary

The elements of a computer system that we looked at:

- Storage Structure – Memory Hierarchy
- I/O Structure – Device Controller / Drivers
- I/O Mechanisms
- General System Architectures

References / Links

- Chapter # 1: **Operating System Concepts** (9th edition) by Silberschatz, Galvin & Gagne