
# Introduction to Operating Systems

Prepared by:

Zeyad Ashraf

# I/O Management


Copyright reserved ITI 2022

# Content

- Need of I/O Management
- I/O Subsystem
- I/O Devices Categories:
  - Block Devices
  - Character Devices
- I/O Protocols Categories:
  - Special Instructions I/O
  - Memory-Mapped I/O
  - Direct Memory Access (DMA)
- Interrupt Handling Mechanisms
- Synchronous Vs. Asynchronous I/O
- Synchronization and Critical Sections
  - Mutex
  - Semaphore
- Deadlocks



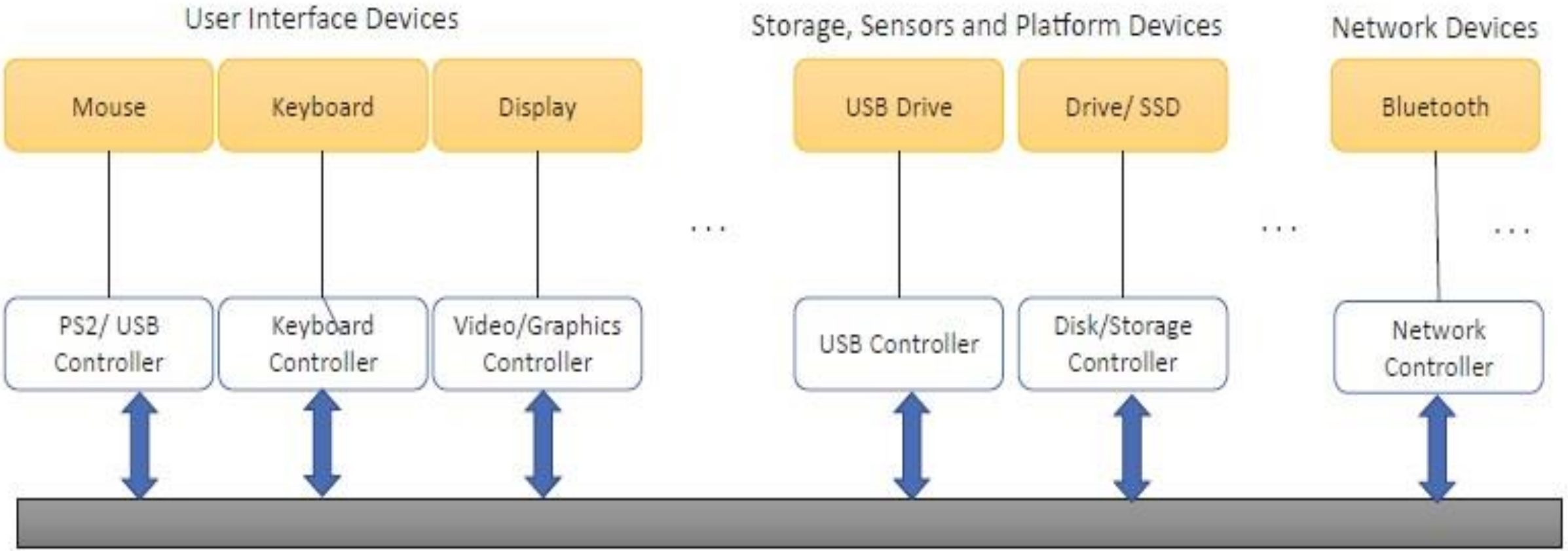
Copyright reserved ITI 2022

# 4.1 Need for I/O Management

- As part of the system, there could be multiple devices that are connected and perform different **input-output functions**.
- These I/O devices could be used for **human interaction** such as display panel, touch panels, keyboard, mouse, and track pads, to name a few.
- Another I/O devices could be to **connect the system to storage devices**, sensors, and so on.
- There could also be I/O devices for **networking** needs that implement certain parts of the networking stack. These could be Wi-Fi, Ethernet, and Bluetooth devices and so on.
- They vary from one to another in the form of **protocols** they use to communicate such as the **data format**, **speed** at which they operate, error reporting mechanisms, and more.
- The OS presents a unified I/O system that **abstracts** the complexity from applications. The OS handles this by establishing protocols and interfaces with each I/O controller. However, *the I/O subsystem usually forms the complex part of the operating system due to the dynamics and the wide variety of I/Os involved*.


Copyright reserved ITI 2022

# 4.1 Need for I/O Management





Copyright reserved ITI 2022

# 4.2 I/O Subsystem

- Input/output devices that are connected to the computer are called ***peripheral*** devices.
- It is communicated with the system through the busses:
- ***Data bus***: to transfer data
- ***Address bus***: used to specify address locations, and
- ***Control bus***: to control a device.
- There ***could be different buses or device protocols*** that an OS may support.
- The most common protocols include:
  - Peripheral Component Interconnect Express (PCIe) protocol,
  - Inter-Integrated Circuit (I2C), and
  - Advanced Configuration and Power Interface (ACPI)
- ***A device*** can be connected over ***one or more*** of these ***interfaces***.


Copyright reserved ITI 2022

# 4.2 I/O Subsystem

- Typically, there is a software component in kernel mode called as the "***device driver***" that handles all interfaces with a device.
- It helps with communicating between the device and the OS and abstracts the device specifics.
- Similarly, there could be a driver at the bus level usually referred to as ***the bus driver***. (Ex: USB Bus driver)

- Most OSs include an inbox driver that implements the bus driver.
- There is usually a driver for each controller and each device.


Copyright reserved ITI 2022

# 4.3 I/O Devices Categories

- The I/O devices can be broadly divided into two categories called ***block*** and ***character*** devices.

1. ***Character:*** data represented sequentially as characters **ex**.. Keyboard ,mouse
2. ***Block:*** data represented as blocks **ex**.. CD, hard disk and flash.

- The ***driver*** would fill the required data and issue a command.
- The device ***firmware*** would respond back to the command and return a code that is utilized by the driver.
- The protocol, size, and format could differ from one device to another.


Copyright reserved ITI 2022

# 4.3.1 Block Devices

- These are devices with which the I/O device controller communicates by sending blocks of data.
- A block is referred to as a group of bytes that are referred together for Read/Write purposes.
- Example: ***flash memory***, ***digital camera***
- The device driver would access by specifying the size of Read/Writes which is **varying** from device to another.


Copyright reserved ITI 2022

# 4.3.2 Character Devices

- Another class of devices are character devices, the subtle difference is that the communication happens by sending and receiving single characters, which is usually a byte or an octet.
- Many serial port devices like ***keyboards***, some ***sensor devices***, and microcontrollers follow this mechanism.


Copyright reserved ITI 2022

# 4.4 I/O Protocols Categories

- The protocols used by the different devices (block devices or character devices) **could vary from one to another.**
- There are three main categories of I/O protocols that are used:
  1. Special Instruction I/O
  2. Memory-Mapped I/O
  3. Direct Memory Access (DMA)

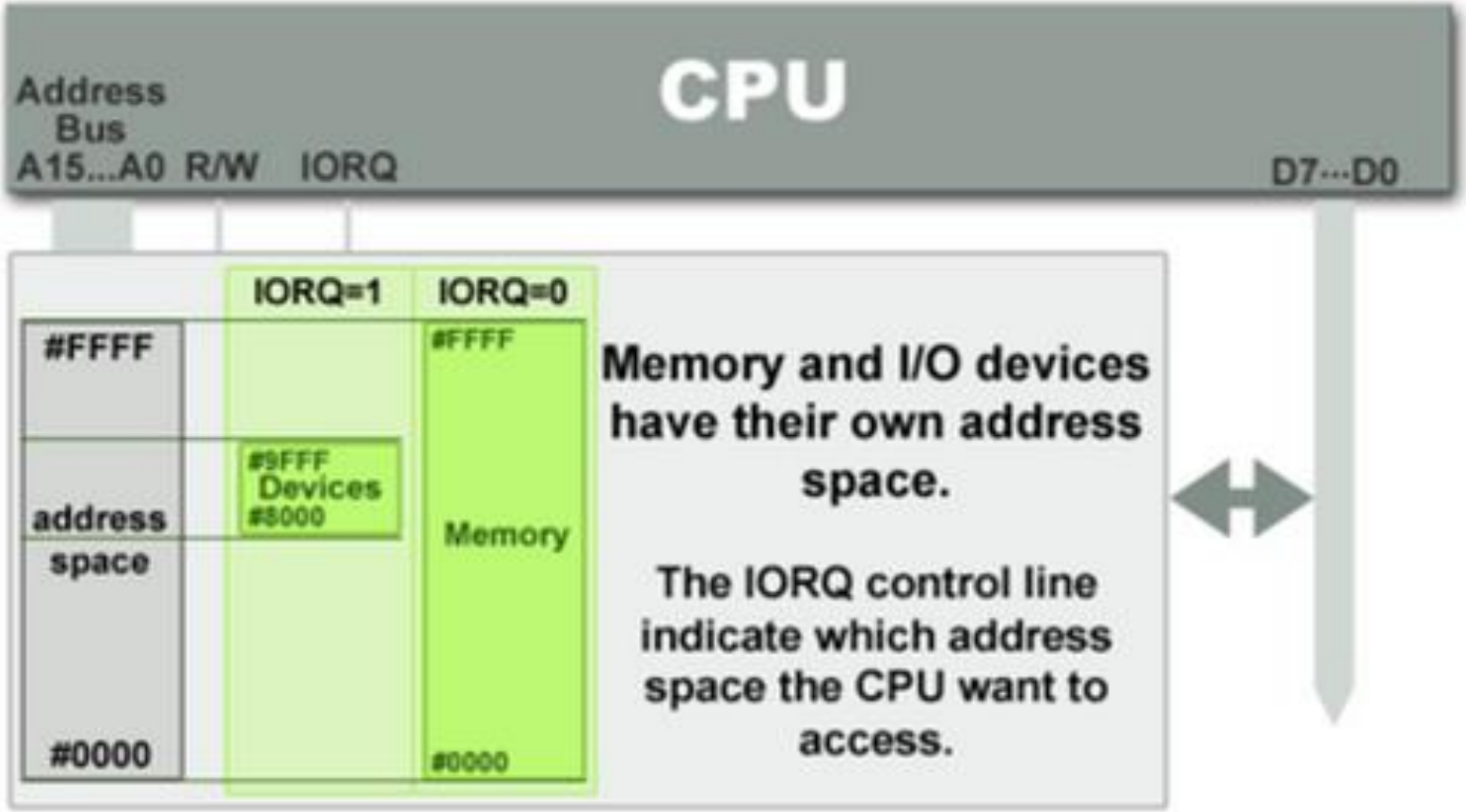

Copyright reserved ITI 2022

# 4.4.1 Special Instruction I/O

- There could be specific **CPU instructions** that are custom developed for communicating with and controlling the I/O devices.
- Each device has a **unique I/O address**
- For example, there could be a CPU-specific protocol to communicate with the **embedded controller**.
- This may be needed for **faster and efficient** communication.
- However, such type of I/Os are special and smaller in number.


Copyright reserved ITI 2022

# 4.4.1 Special Instruction I/O





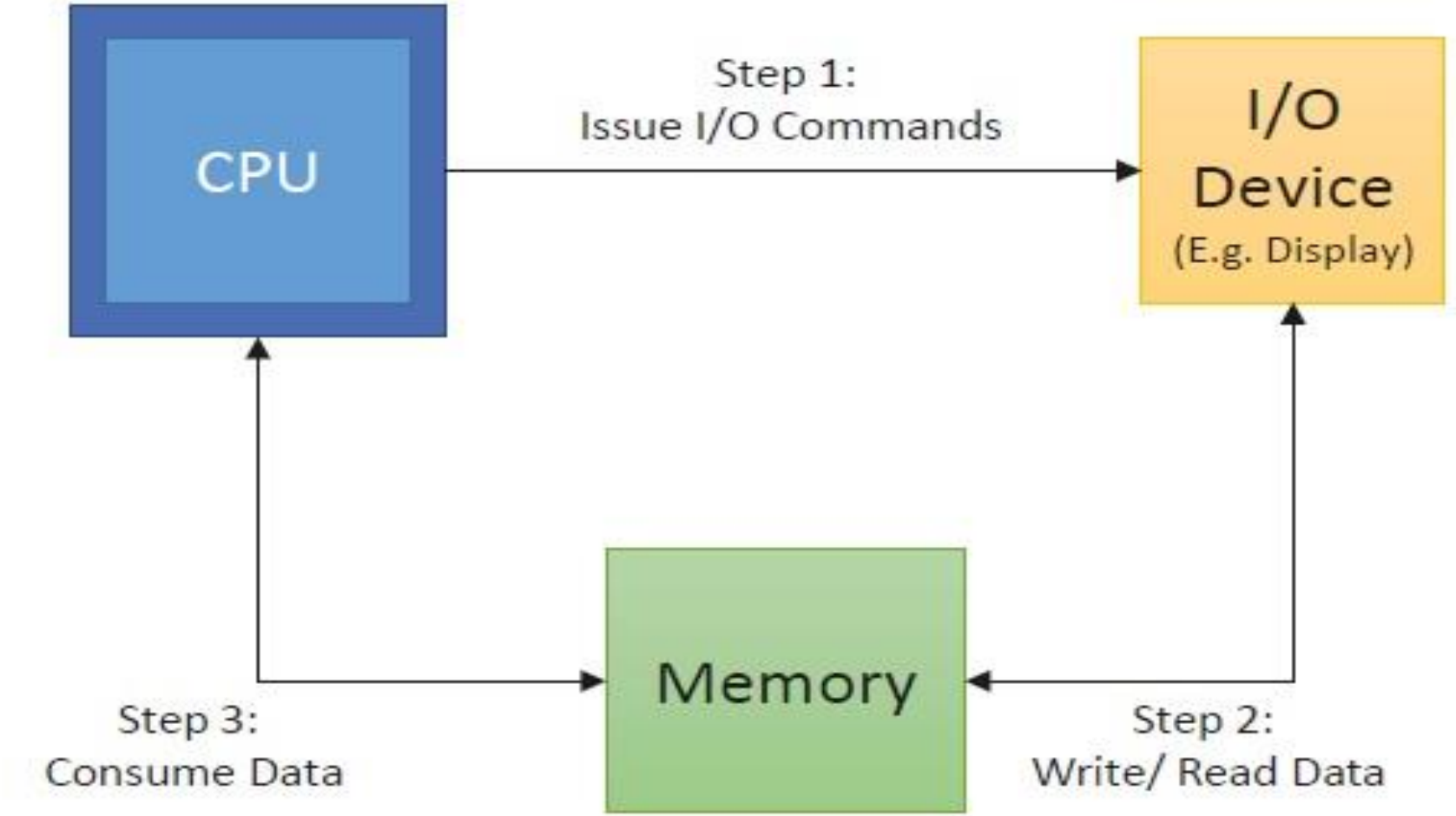
Copyright reserved ITI 2022

# 4.4.2 Memory-Mapped I/O

- The **most common** form of I/O protocol is memory-mapped I/O (MMIO).
- The device and OS agree on **a common address range** carved out by the OS, and the **I/O device** makes reads and writes from/to this space to communicate to the OS.
- OS components such as drivers will communicate using this **interface** to talk to the device.
- Hence, it is used to enable **high-speed communication** for network and graphics devices that require high data transfer rates due to the volume of data being passed.


Copyright reserved ITI 2022

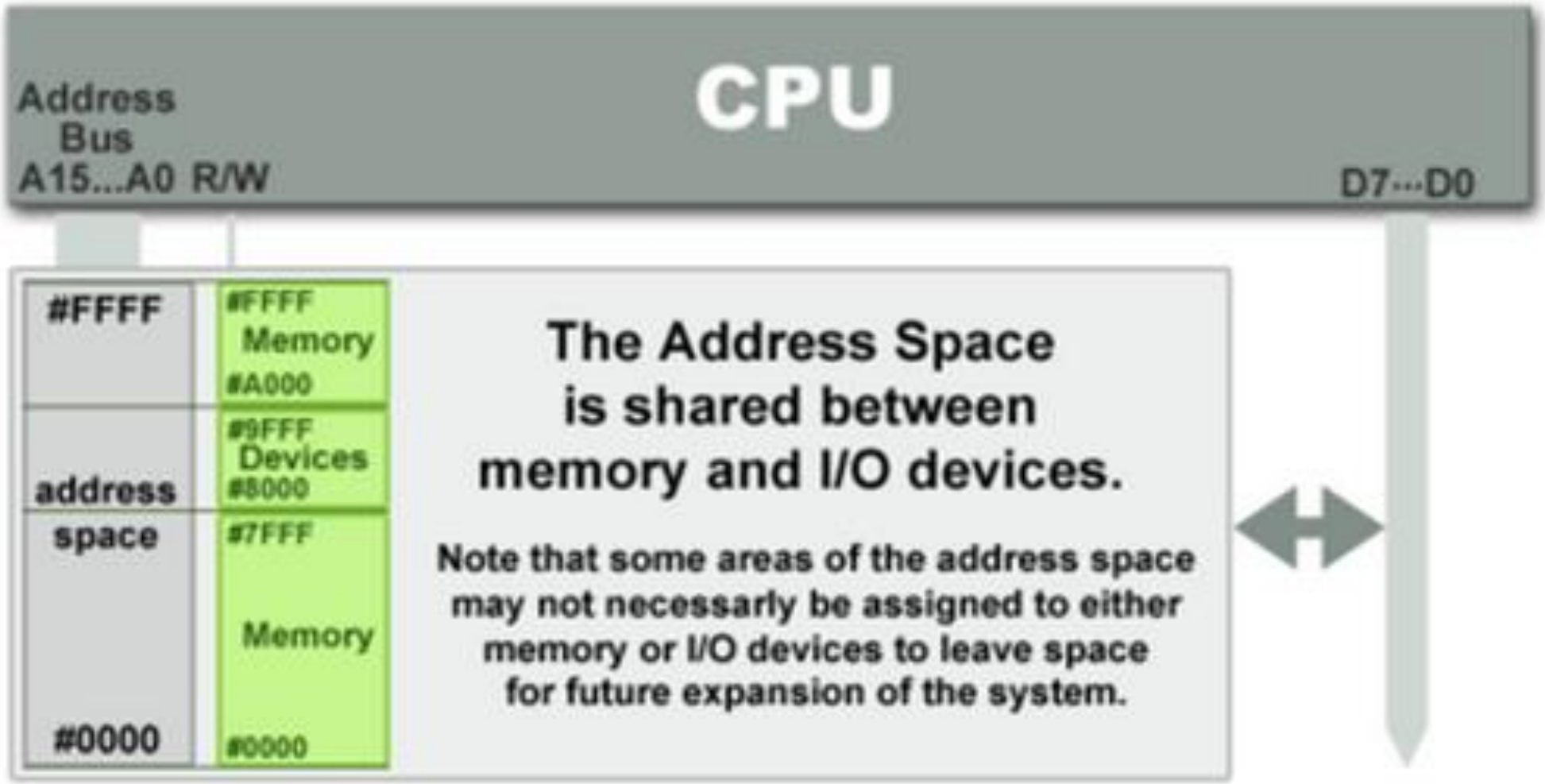
# 4.4.2 Memory-Mapped I/O





Copyright reserved ITI 2022

# 4.4.2 Memory-Mapped I/O





Copyright reserved ITI 2022

# 4.4.3 Direct Memory Access (DMA)

- There could be **devices that run at a slower speed than CPU** or the bus it is connected on. In this case, the device can leverage DMA.
- Here, the **OS *grants authority to another controller***, usually referred to as the *direct memory access controller*, to ***interrupt*** *the CPU after a specific data transfer is complete*.
- The devices running at a smaller rate **can communicate back to the DMA controller** after completing its operation.
    - *As a programmer, you could be interacting with devices that may perform **caching** and have different **error reporting** mechanisms, protocols, and so on.*
- ***Ex*** *:Video & audio devices, hard disks .*


Copyright reserved ITI 2022

# 4.5 Interrupt Handling Mechanisms

- Polling or programmed I/O:
  - In this mechanism, **each period of time examine interrupt** information and calls a **specific routine** (driver)

- Interrupt-Driven I/O:
  - In This mechanism, there is an ***interrupt vector*** which contains the addresses for all ***Interrupt Service Routines*** (ISR) –***drivers***- for all I/O devices, according the ***Interrupt Request Number*** (IRQ) –which ***unique*** for each device- the OS acquire the address of the address of the correct service routine
  - Most of OSs using this mechanism


Copyright reserved ITI 2025

# 4.5 Interrupt Handling Mechanisms

- **Synchronous I/O: Example**: Printing Operation
  - Process request I/O operation
  - I/O operation is started
  - I/O Operation is complete
  - Control is returned to the user process
- **Asynchronous I/O**: Example: Using Networking sending and receiving using threads
  - Process Request I/O operation
  - I/O operation is started
  - Control is returned immediately to the user process
  - I/O continues while system operations occur


Copyright reserved ITI 2022

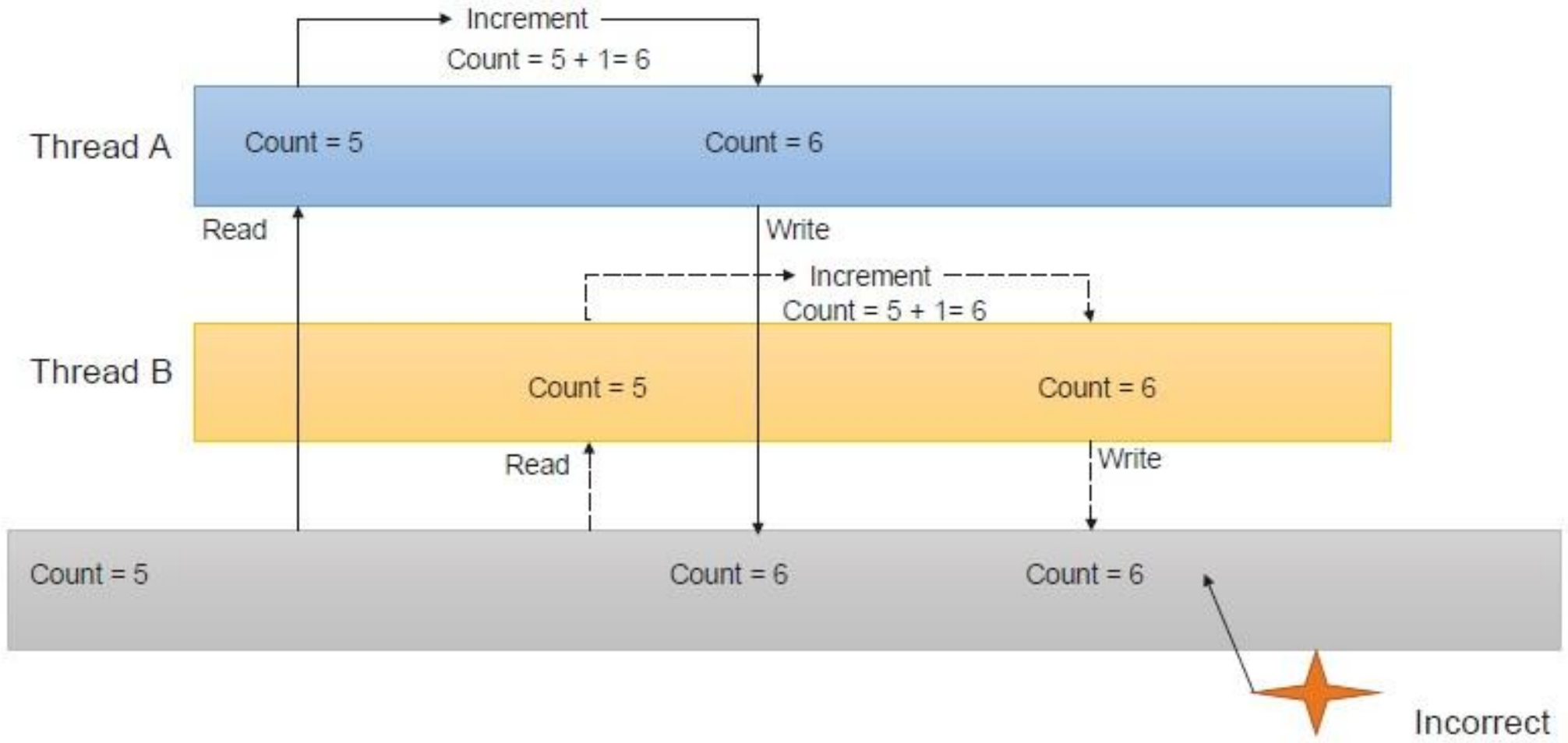
# 4.7 Synchronization and Critical Sections

- In multi-threaded applications, if one thread tries to change the value of **shared data** at the **same time** as another thread tries to read the value, there could be a **race condition** across threads.
- In this case, the result can be **unpredictable**.
- The access to such shared variables via shared memory, files, ports, and other I/O resources **needs to be synchronized** to protect it from being corrupted.

- order to support this, the operating system provides **mutexes** and **semaphores** to coordinate access to these shared resources.


Copyright reserved ITI 2022

# 4.7 Synchronization and Critical Sections

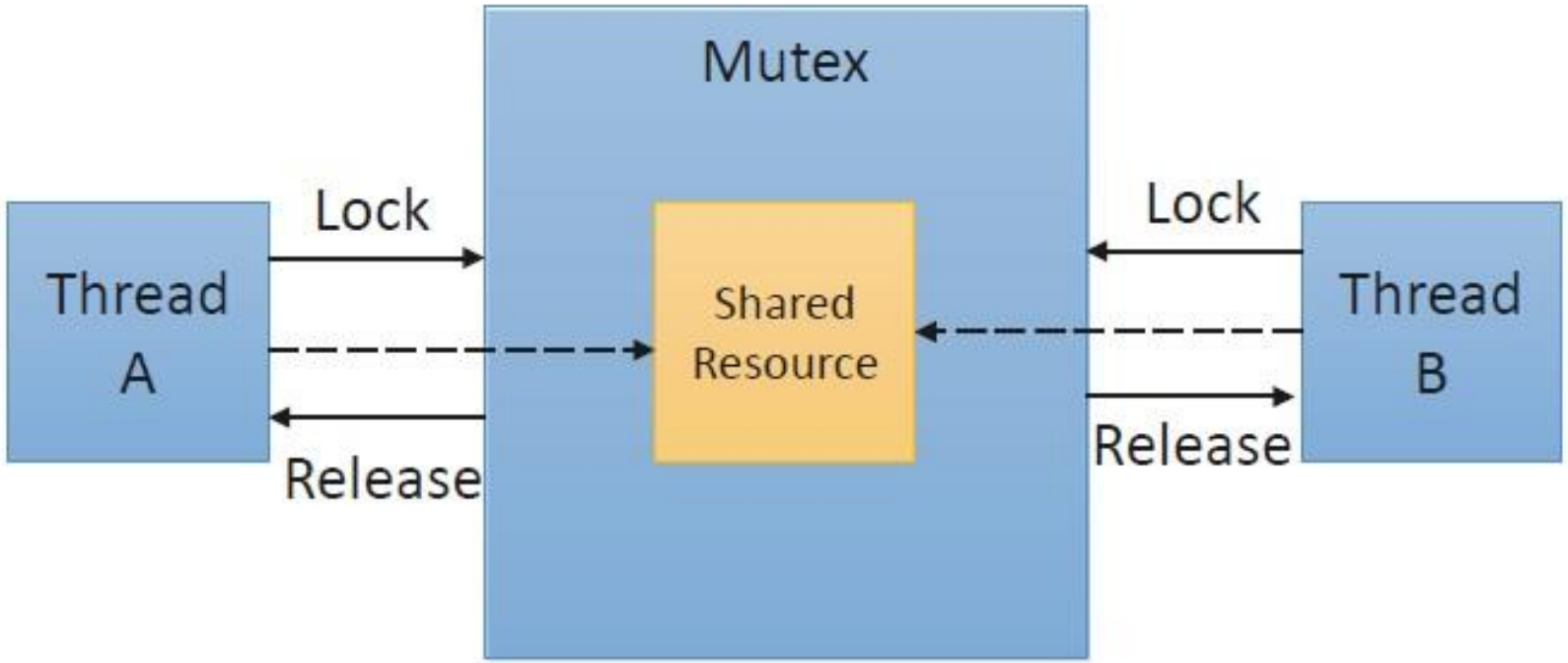




Copyright reserved ITI 2022

# 4.7.1 Mutex

- A mutex is used for implementing ***mutual exclusion***: either of the participating processes or threads can have the key (mutex) and proceed with their work.
- **The other one would have to wait until the one holding the mutex finishes**

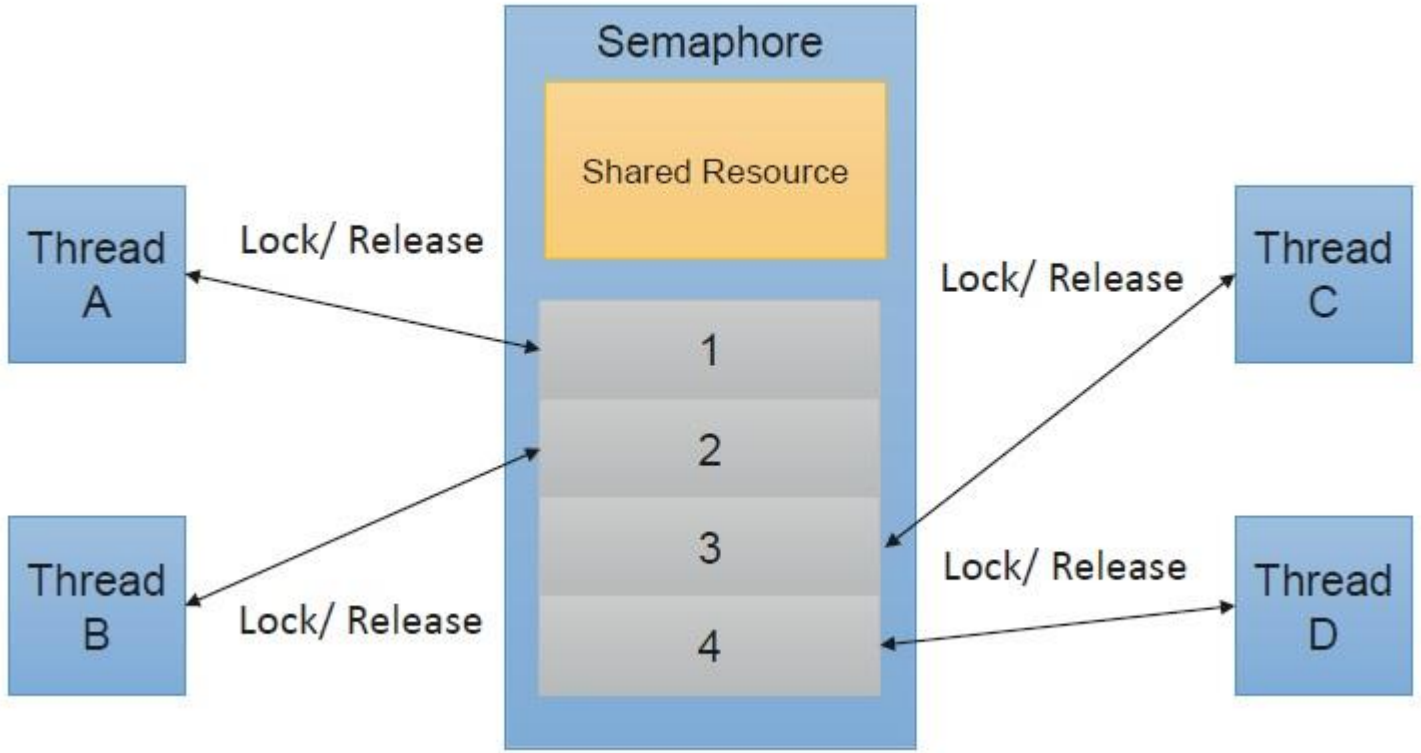




Copyright reserved ITI 2022

# 4.7.2 Semaphore

- A semaphore is a **generalized mutex**.
- A binary semaphore can assume a value of 0/1 and can be used to perform locks to certain critical sections.

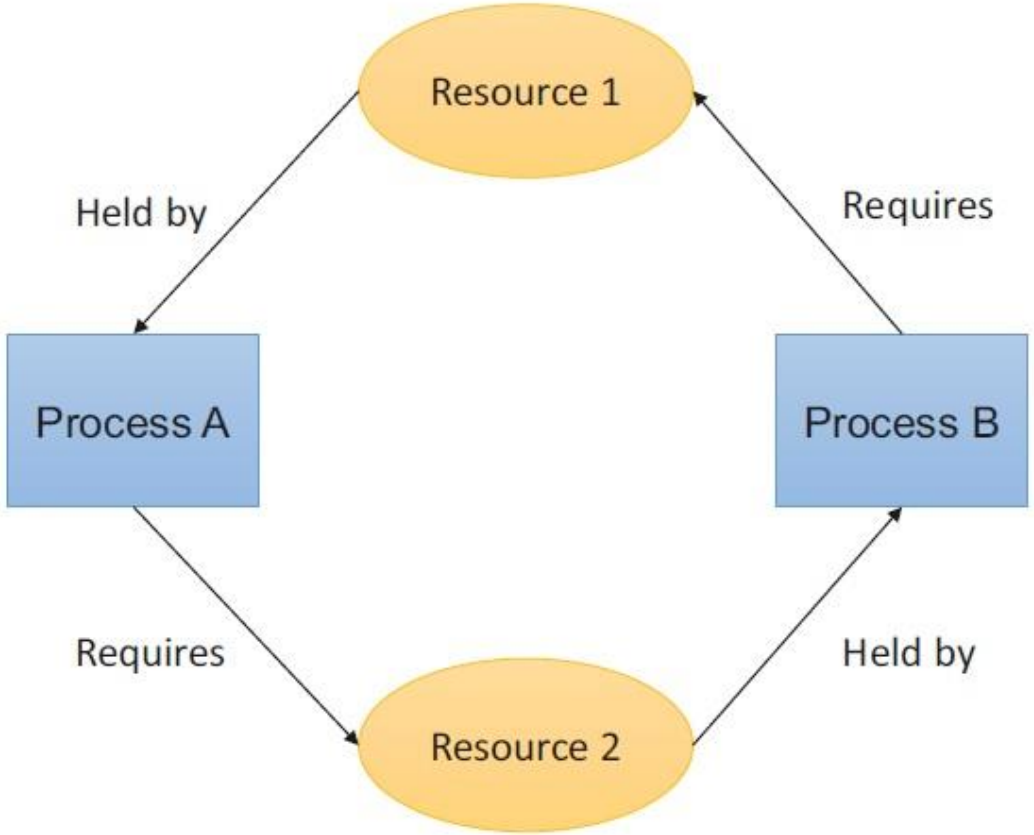




Copyright reserved ITI 2022

# 4.8 Deadlocks

- When a set of processes become blocked because each process is holding a resource and waiting for another resource acquired by some other process. This is called as a ***deadlock***.
- Process A holds Resource 1 and requires Resource 2. However, Process B already is holding Resource 2, but requires Resource 1. Unless either of them releases their resource**.**
- **either of the processes may be able to move forward with the execution.**





Copyright reserved ITI 2022

# 4.8 Deadlocks

- A deadlock can arise if the following four conditions hold:
  - **Mutual Exclusion**: There is at least one resource on the system that is **not shareable**. This means that only one process can access this at any point in time. In the preceding example, Resources 1 and 2 can be accessed by only one process at any time.
  - **Hold and Wait**: A process is **holding at least one resource** and is waiting for other resources to proceed with its action. In the preceding example, both Processes A and B are holding at least one resource.
  - **No Preemption**: A resource cannot be **forcefully** taken from a process unless released automatically.
  - **Circular Wait**: A set of processes are **waiting for each other in circular** form.


Copyright reserved ITI 2022

# Thank You

*With My Best Wishes*

*Zeyad ashraf*