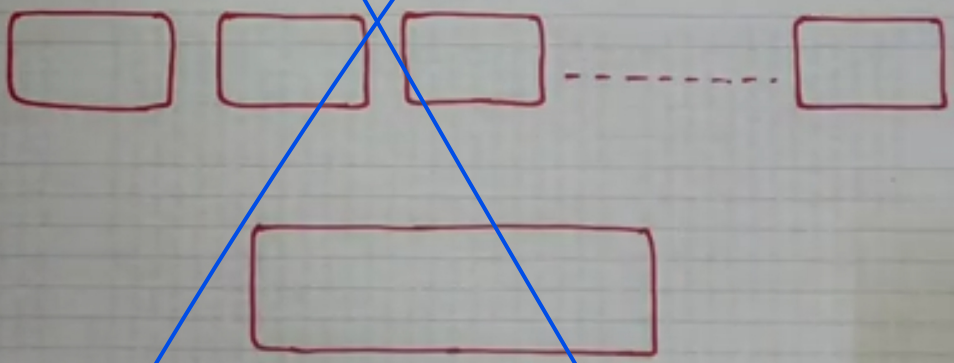
Types of operating System:-

Multiprocessing operating System:-

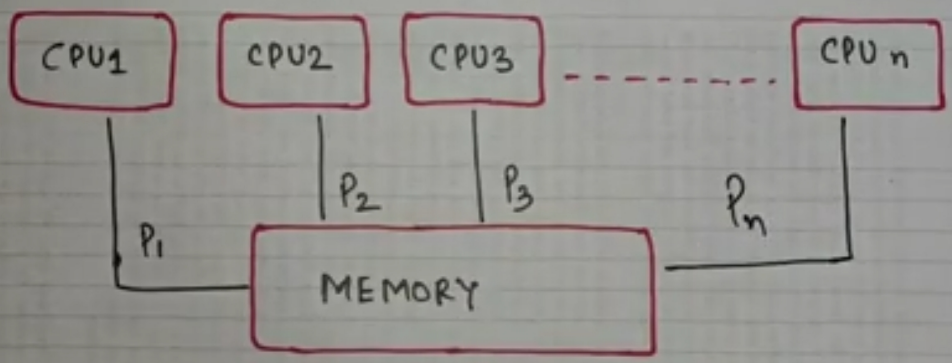
↳ More than one processor is present in the System.
→ It allows more than one process to be executed at same time.
→ These multiple CPU's shares memory bus, memory and devices.
→ All these devices are tightly coupled b/w the processors.



Types of operating System:-

Multiprocessing operating System:-

↳ More than one processor is present in the System.
→ It allows more than one process to be executed at same time.
→ These multiple CPU's shares memory bus, memory and devices.
→ All these devices are tightly coupled b/w the processors.

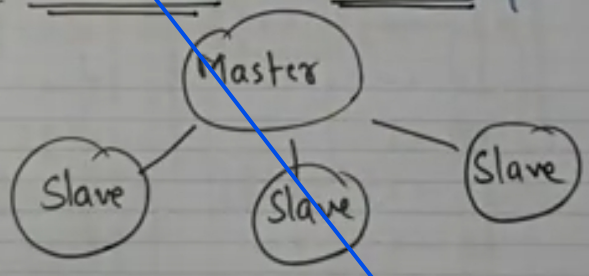


Types of Multiprocessing Operating System:-

	Symmetric	Asymmetric
i)	One OS Controls all CPUs where each CPU has equal rights.	There is a Master Processor that gives instruction to all the other Processors.
ii)	All CPUs are in Peer to Peer relationship.	Master-Slave relationship.

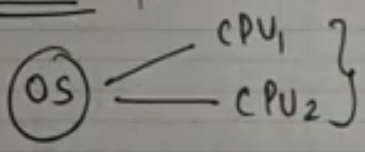
Types of Multiprocessing Operating System:

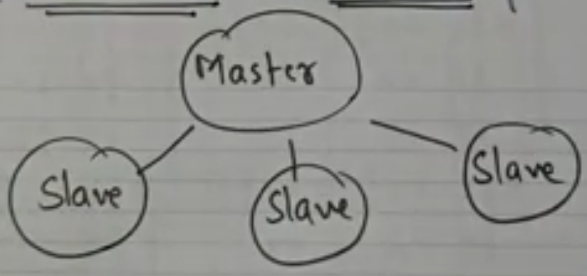
	Symmetric	Asymmetric
i)	One OS Controls all CPUs where each CPU has equal rights.	There is a Master Processor that gives instruction to all the other Processors.
ii)	All CPUs are in Peer to Peer relationship.	Master-Slave relationship.



Types of Multiprocessing Operating System:-

	Symmetric	Asymmetric
i)	One OS Controls all CPUs where each CPU has equal rights.	There is a Master Processor that gives instruction to all the other Processors.
ii)	All CPUs are in Peer to Peer relationship.	Master-Slave relationship.





Multiprocessing operating System:-

Advantages:-

- i) Max Throughput.
- ii) More Reliable. { fault of one processor does not cause loss of work }
- iii) Fast Processing.
- iv) Improved efficiency.

Disadvantages:-

- i) Complicated
- ii) Memory Requirement is more.

Multiprocessing operating System:-

Advantages:-

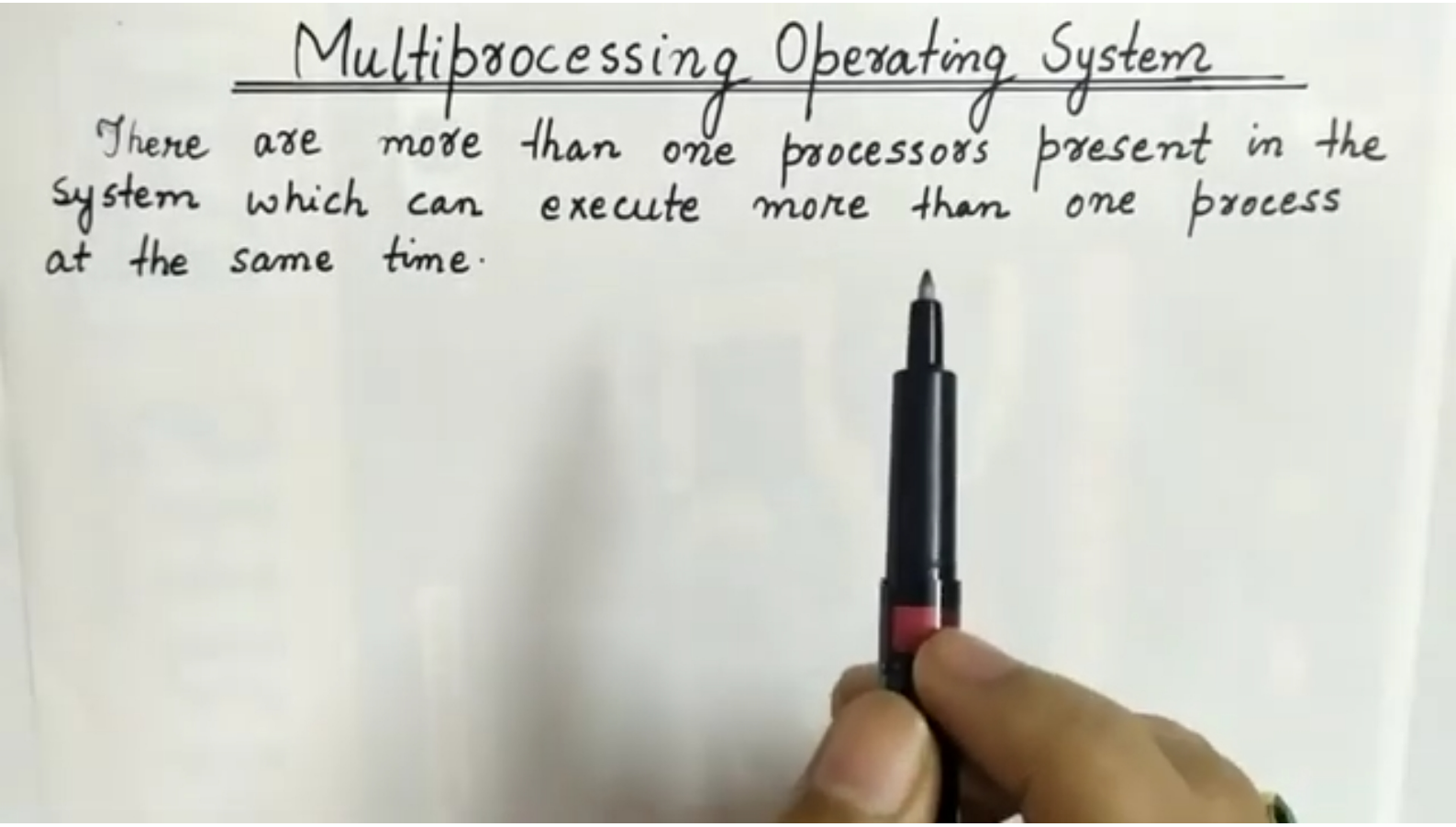
↳ i) Max Throughput.
ii) More Reliable. { Fault of one processor does not cause loss of work }
iii) Fast Processing.
iv) Improved Efficiency.

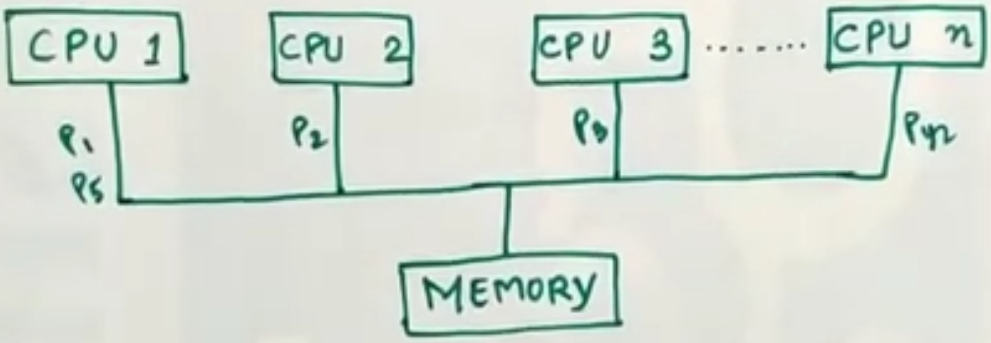
Disadvantages:-

↳ i) Complicated } CPU Scheduling.
ii) Memory Requirement is more.

Multiprocessing Operating System

There are more than one processors present in the system which can execute more than one process at the same time.





Types :

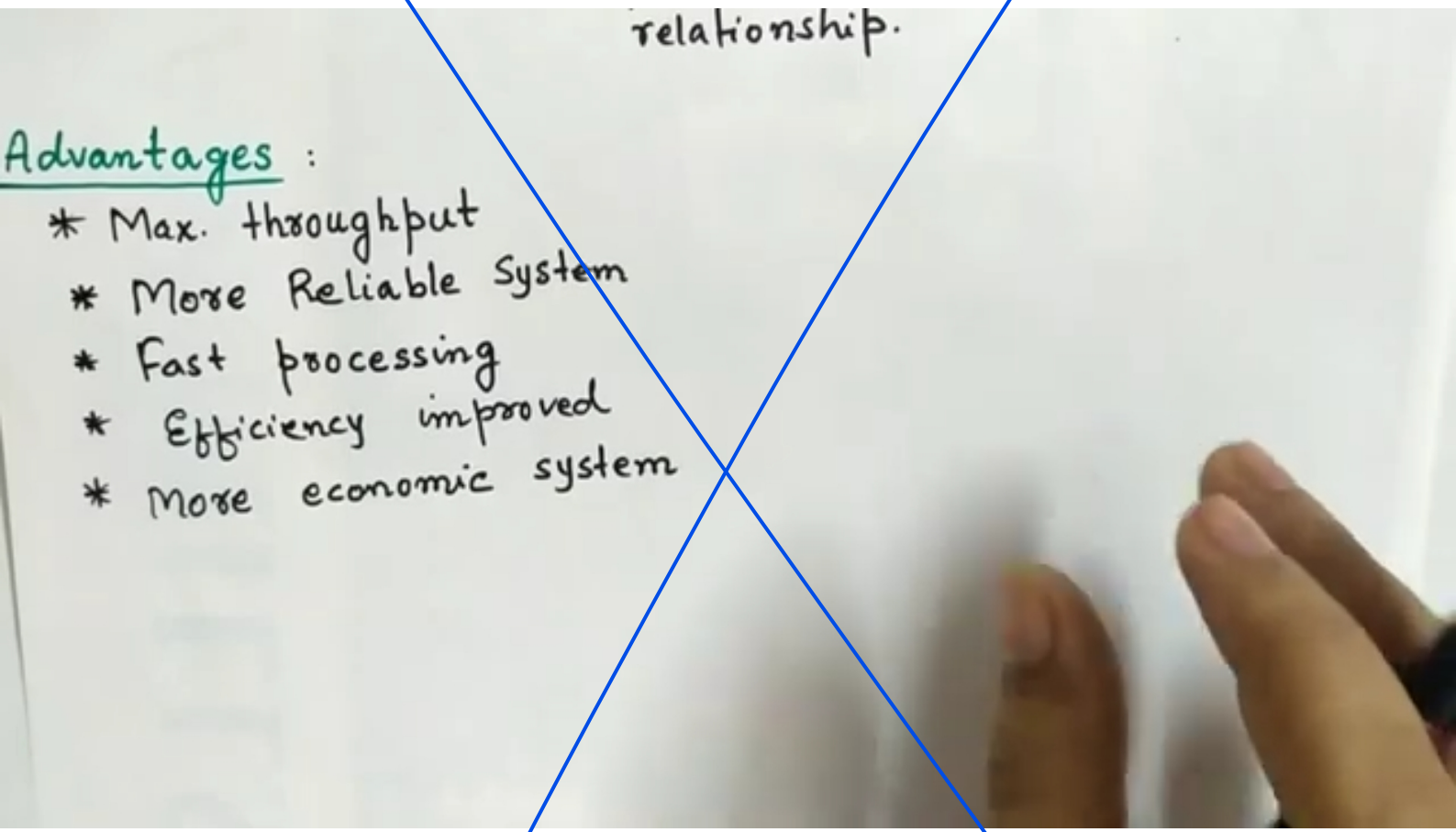
1) Symmetric Multiprocessors : One OS control all CPU, each CPU has equal rights. All the CPU are in peer to peer relationship.

2) Asymmetric Multiprocessors : There is a master process that gives instruction to all the other processors. It contains master-slave relationship.

relationship.

Advantages :

* Max. throughput
* More Reliable System
* Fast processing
* Efficiency improved
* More economic system



Symmetric Multiprocessing

gives instruction to all the processors. It contains master-slave relationship.

Advantages :

* Max. throughput
* More Reliable System
* Fast processing
* Efficiency improved
* More economic system

Disadvantages :

* Complicated
* Large main memory required

Introduction

- A multiprocessor system is an interconnection of two or more CPU, with memory and input-output equipment.
- Multiprocessor can be defined under MIMD category. (As per Flynn's Taxonomy)
- A single job can be divided into independent tasks
  - By Manually
  - By Programmer
  - By Compiler
- Subdivided program can be executed parallelly.
- Due to any kind of fault, any processor is failed If any processor will be failed, task can be assigned to another processor of that faulty processor.

Introduction

- A multiprocessor system is an interconnection of two or more CPU, with memory and input-output equipment.
- Multiprocessor can be defined under MIMD category. (As per Flynn's Taxonomy)
- A single job can be divided into independent tasks
  - By Manually
  - By Programmer
  - By Compiler
- Subdivided program can be executed parallelly.
- Due to any kind of fault, any processor is failed If any processor will be failed, task can be assigned to another processor of that faulty processor.

Multiprocessor

❑Characteristics:

- **Reliability:** Improves the reliability of the system so that a failure in one part has a limited effect on the rest of the system.
- **Performance:** Multiprocessing can improve performance by decomposing a program into parallel executable tasks.
- **Multi Tasking:** More than two processors are available in Multiple program can be executed same time.
- **Increased Throughput:** Increased throughput because of execution of multiple jobs in parallel portions of the same job in parallel.

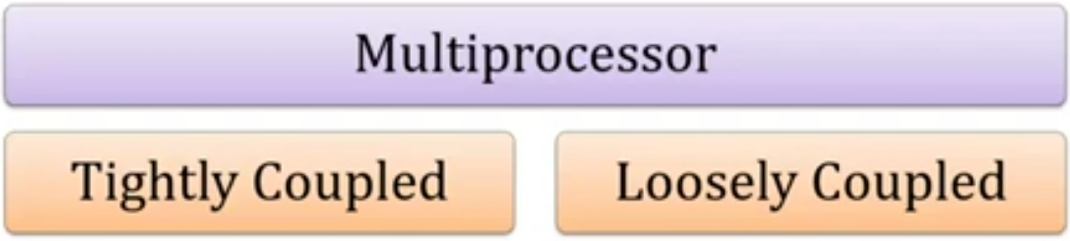
Multiprocessor

- **Advantages**
  - Improve Reliability
  - Enhance Performance
  - Multi Tasking
  - High Throughput
  - More Economic Systems

- **Disadvantages**
  - More Costly
  - Complicated Operating System Required
  - Large Main Memory Required

Multiprocessor

- Multiprocessors are classified by the way their memory is organized.



i. Multiprocessor system with common shared memory is classified as a ***shared memory*** or ***tightly coupled*** multiprocessor.

ii. Another type is the ***distributed memory*** or ***loosely-coupled*** system.

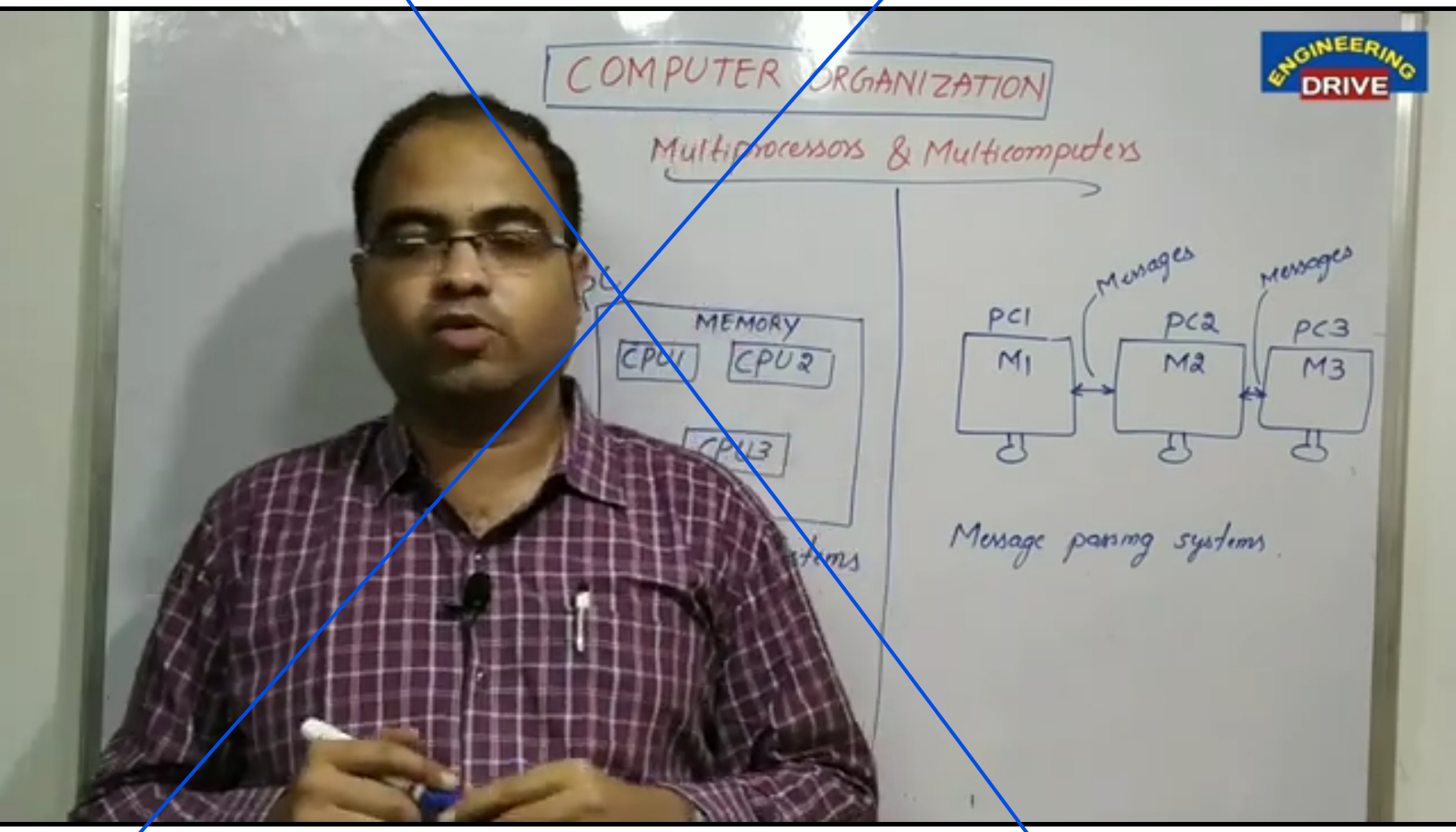
- **Multiprocessor:**

  » Multiprocessing is a type of processing in which two are more processors work togather to process more than one program simultaneously.

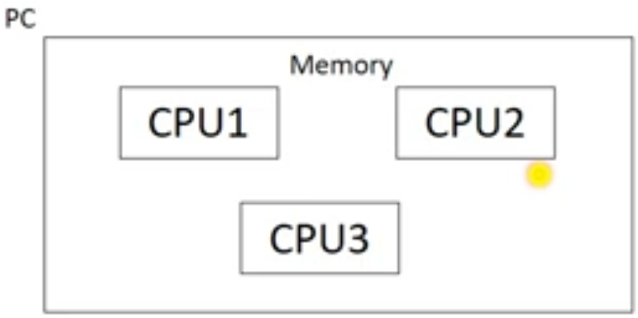
  » It allows the system to do more work in a short time.

  » Multiprocessor system is also known as Parallel system or tightly coupled system.

  » It means that multiple processors are tied together in some manner.

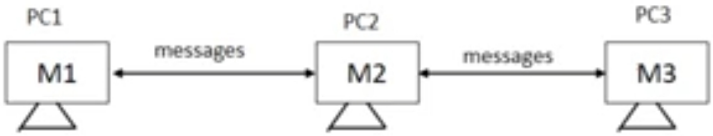

COMPUTER ORGANIZATION
ENGINEERING DRIVE
Multiprocessors & Multicomputers
MEMORY
CPU1
CPU2
CPU3
Messages
Messages
PC1
PC2
PC3
M1
M2
M3
Message passing systems

Multiprocessors



⇒ Multiprocessors system/computer also known as "shared memory systems"
⇒ Multiprocessor is a system in which we have multiple cpus. All the cpus/processors will access same memory called shared memory system
⇒ Here sharing the same memory then only they are communicating each other

Multicomputers



⇒ Connect S computer and increase speed of computer
⇒ A single computer can do limited work. So adding 2 extra computer this forms network of computers
This way we can increase the speed of computer
⇒ Here every computer have separate memory
⇒ They will communicate with the help of messages
⇒ Another name of multicomputer is "Message passing systems"
⇒ The computers are connecting by passing the message called message passing system"

Multiprocessors

- Single computer with multiple processors
- Each PE's do not have their own individual memories. memory & I/O resources are shared - called as shared memory multiprocessors.
- Communication between

Multicomputer

- Multiple autonomous computers.
- Each PE's has its own memory and resources - no sharing - called Distributed Memory Multicomputers.
- Communication between PE's not mandatory

* Tightly coupled - due to high degree of resource sharing.

multiprocessor

* use Dynamic n/w - thus Communication links can be reconfigured

* 3 Types
  - UMA model
  - NUMA model

* loosely coupled as there is no resource sharing.

multicomp

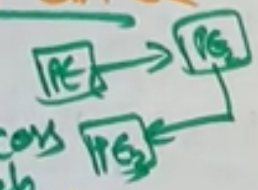
* use static n/w - Connection of switching units is fixed.

* 1 Type
  - NORMA model / Distributed Memory Multicomp.

★ Tightly Coupled - due to high degree of resource sharing.

multiprocessor

★ use Dynamic n/w - thus Communication links can be reconfigured



multiprocessor

★ 3 Types
- UMA model
- NUMA model
- COMA model

★ loosely Coupled as there is no resource sharing.

multicomp

★ use static n/w - Connection of switching units is fixed.

★ 1 Type
- NORMA model / Distributed-Memory Multicomputer.

END
ENGINEER
NEXT DOOR

DC - MOD1 - 1.1

1.1. Characterization of Distributed Systems:

⇒ Hardware and Software concepts :

1. Hardware Concepts:

→ Show organization of hardware, their interconnection and manner in which their comunicate with each other.

→ Now the machines are basically divided into two groups:

1.1 Multiprocessor : Processors share memory

1.2 Multicomputers : Each processors has its own private memory.

END
ENGINEER NEXT DOOR

1.1 Multiprocessors :

→ Multiple processors in a single machine.

→ All processors share a common memory. (shared memory system)

→ Communication is fast through memory access.

→ Interconnection :

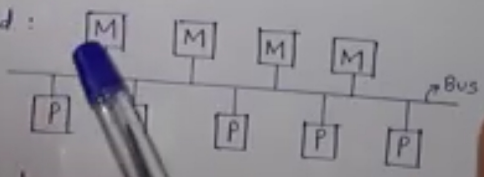
⇒ Bus-based : Single communication bus used.

⇒ Switch-based : High-speed switches for routing data.

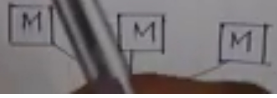
→ Example : Server machines with multiple CPUs.

Diagram : Multiprocessor Shared Memory

① Bus-Based :



② Switch-Based :



→ Communication is fast through memory access.

→ Interconnection :

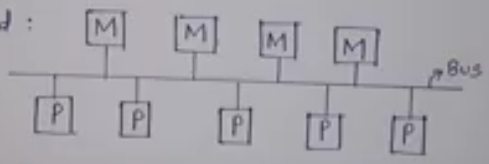
⇒ Bus-based : Single communication bus used.

⇒ Switch-based : High-speed switches for routing data.

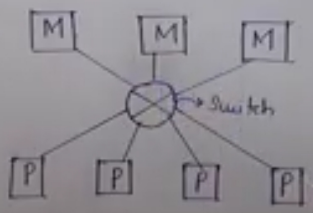
→ Example : Server machines with multiple CPUs.

Diagram : Multiprocessor Shared Memory

① Bus-Based :



② Switch-Based :



(Note : P-processor, M-memory)

1.2 Multicomputers:

→ Independent computers connected over a network (LAN / WAN).

→ Each has its own private memory.

→ Communication happens via message passing over the network.

→ Interconnection:

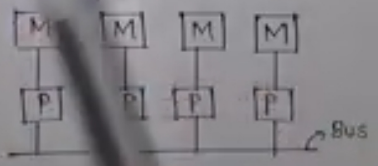
⇒ Bus-based: Simple shared network cable.

⇒ Switch-based: Direct faster connections between nodes.

→ Example: office computer networks.

Diagram: Multicomputer Private Memory

① Bus-based LAN:



② Switch-based LAN: M M

→ Interconnection :

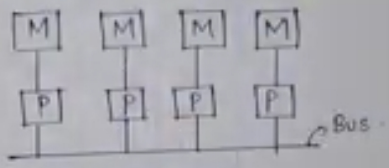
⇒ Bus-based : Simple shared network cable.

⇒ Switch-based : Direct faster connections between nodes.

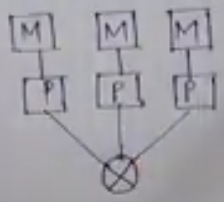
→ Example : Office computer networks.

Diagram : Multicomputer Private Memory

① Bus-based LAN :



② Switch-based LAN :



END
ENGINEER
NEXT DOOR

Summary:

Feature	Multiprocessor	Multicomputer
Memory	Shared memory	Private memory
Interconnection	Internal Bus/Switch	LAN/Switch (External)
Communication	Memory access	Message passing
Example	Server CPUs	office LAN systems.

Q1. What is multiprocessor? Differentiate between multiprocessor and multicomputer.

There are more than on processor present in the system which can execute more than one process at the same time.

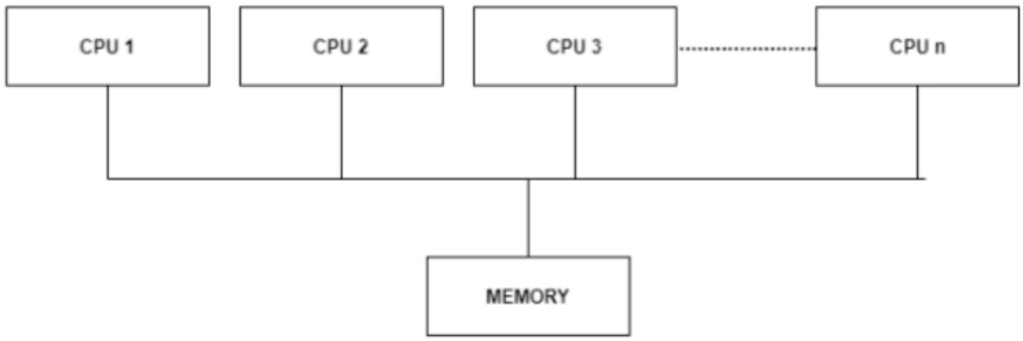
A **multiprocessor** is a computer system containing two or more central processing units (CPUs) that share resources like memory and buses. These processors work together to execute tasks simultaneously, which improves speed and performance.

Multiprocessor can be divided into independent task

- By Manually
- By programmer
- By Complier

Example: Multi-core processors, Supercomputers, Medical Imaging, Large Databases, Scientific Simulations

Uses: In banking systems, scientific simulations, cloud servers, and real-time applications where fast processing is required.



Multiprocessing Architecture

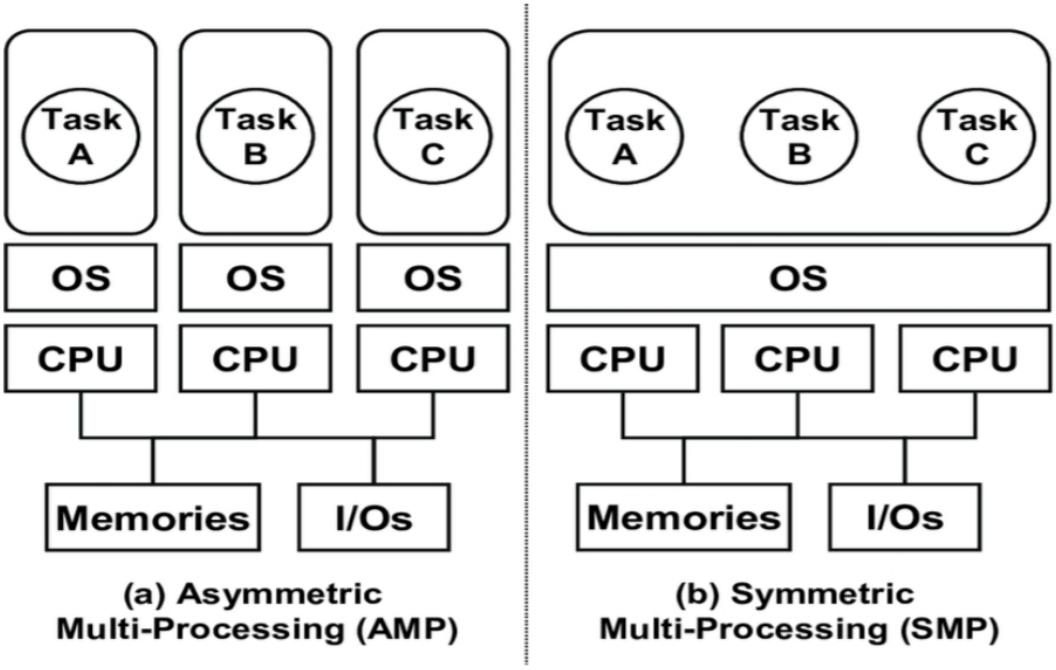
Types of Multiprocessors

1) **Asymmetric Multiprocessors**

   There is a master processor that gives instruction to all the other processor. It contains master-slave relationship.

2) **Symmetric Multiprocessors**

   One OS control all CPU, each CPU has equal right. All the CPU are peer to peer relationship.



Advantages

- Max throughput
- More Reliable system
- Fast Processing
- Efficiency improved
- More Economic system

Dis-Advantages

- More Costly
- Complicated
- Large main memory required

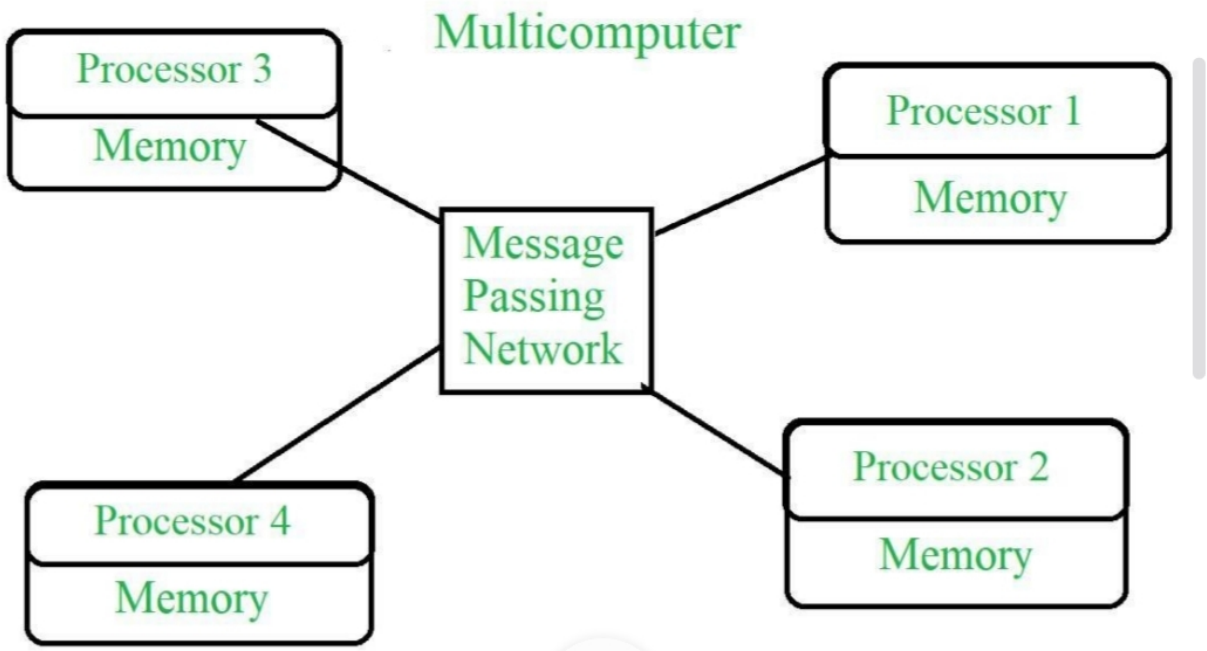
Multicomputer

A **multicomputer** is a system made up of **multiple independent computers** connected through a **network**.

- Each computer has its **own private memory and I/O devices**.
- They communicate through **message passing** instead of shared memory.

Example: Distributed systems, computer clusters, and networks like the Internet.

Uses : Cloud Computing Platforms, Search Engines & Web Services, Big Data & AI, Real-Time Communication



Advantages

- Cost Effective
- Scalability
- Reliability / Fault Tolerance
- Resource Sharing
- Parallel Processing
- Geographical Distribution

Disadvantages

- Complex Network Management
- Security Issues
- Software Complexity
- Latency
- Data Consistency Issues
- Maintenance Cost

Difference

Aspect	Multiprocessor	Multicomputer
System Type	Single computer with multiple CPUs	Collection of independent computers
Architecture	Processors are connected and share memory	Each node has its own memory and is connected by a network
Resource Sharing	All processors share common memory and I/O	No shared memory, only message passing
Communication Method	Shared memory communication	Network-based communication
Cost	Expensive hardware setup	Relatively cheaper (uses normal computers)
Examples	Multi-core processors in servers	Distributed systems, Grid computing

Multiprocessor	Multicomputer
1] A System with two or more CPUs that allows simultaneous processing of programs.	1] A set of processors connected by the communication n/w that works jointly to solve a computation Problem.
2] Easier to process.	2] less easy to program.
3] More difficult and costly to build.	3] Easier and cost effective to build.
4] Multiprocessor's support parallel Computing.	4] Multicomputer's supports distributed Computing.

Interprocess Communication

Processes executing concurrently in the operating system may be either independent processes or cooperating processes.

Independent processes - They cannot affect or be affected by the other processes executing in the system.

Cooperating processes - They can affect or be affected by the other processes executing in the system.

Any process that shares data with other processes is a cooperating process.

Interprocess Communication

Processes executing concurrently in the operating system may be either independent processes or cooperating processes.

Independent processes - They cannot affect or be affected by the other processes executing in the system.

Cooperating processes - They can affect or be affected by the other processes executing in the system.

Any process that shares data with other processes is a cooperating process.

There are several reasons for providing an environment that allows process cooperation:

Information sharing

Computation speedup

Modularity

Convenience

Cooperating processes require an interprocess communication (IPC) mechanism that will allow them to exchange data and information.

There are two fundamental models of interprocess communication:

Cooperating processes require an interprocess communication (IPC) mechanism that will allow them to exchange data and information.

There are two fundamental models of interprocess communication:

(1) Shared memory

(2) Message passing

Cooperating processes require an interprocess communication (IPC) mechanism that will allow them to exchange data and information.

There are two fundamental models of interprocess communication:

(1) Shared memory

(2) Message passing

- In the shared-memory model, a region of memory that is shared by cooperating processes is established.
  Processes can then exchange information by reading and writing data to the shared region.

Cooperating processes require an interprocess communication (IPC) mechanism that will allow them to exchange data and information.

There are two fundamental models of interprocess communication:

(1) **Shared memory**

(2) **Message passing**

- In the shared-memory model, a region of memory that is shared by cooperating processes is established.
  Processes can then exchange information by reading and writing data to the shared region.

- In the message passing model, communication takes place by means of messages exchanged between the cooperating processes.

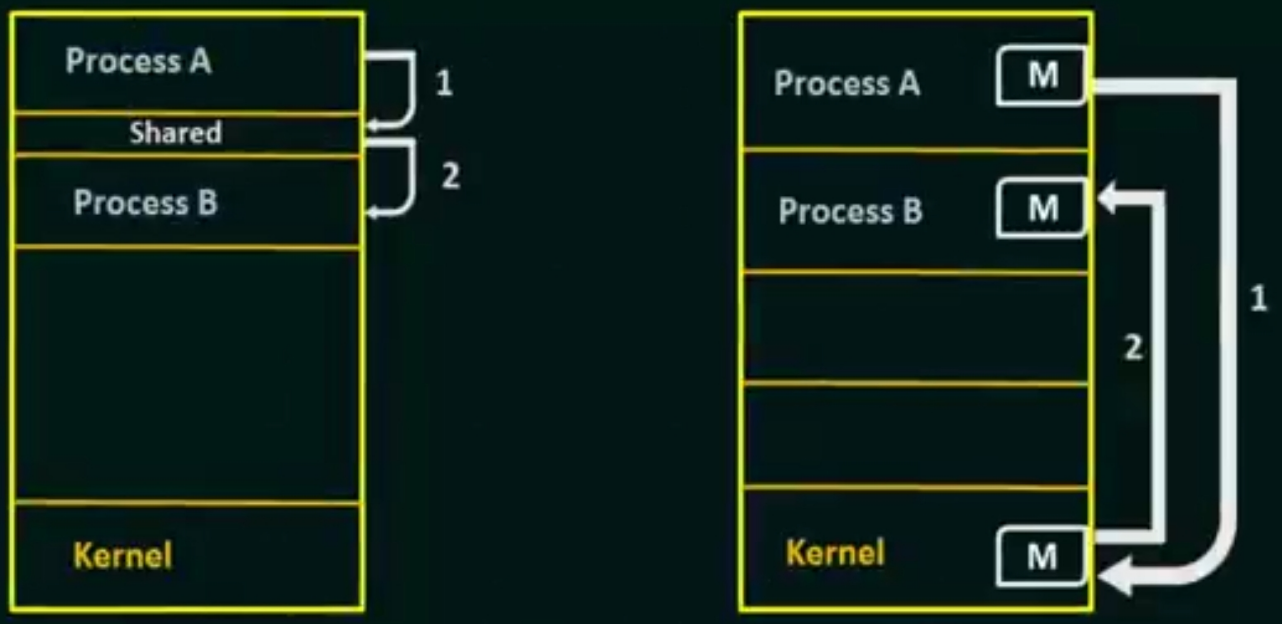
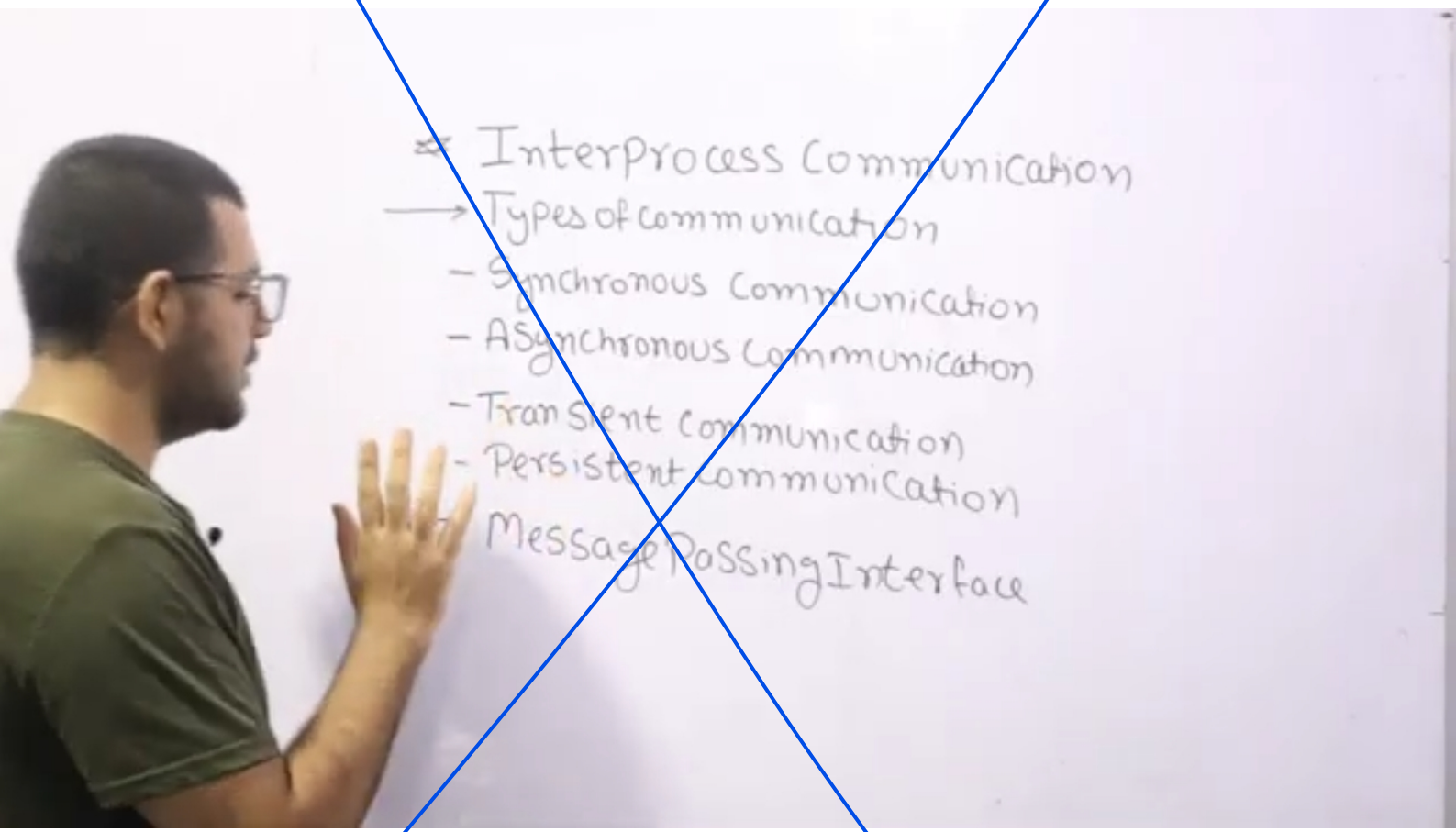
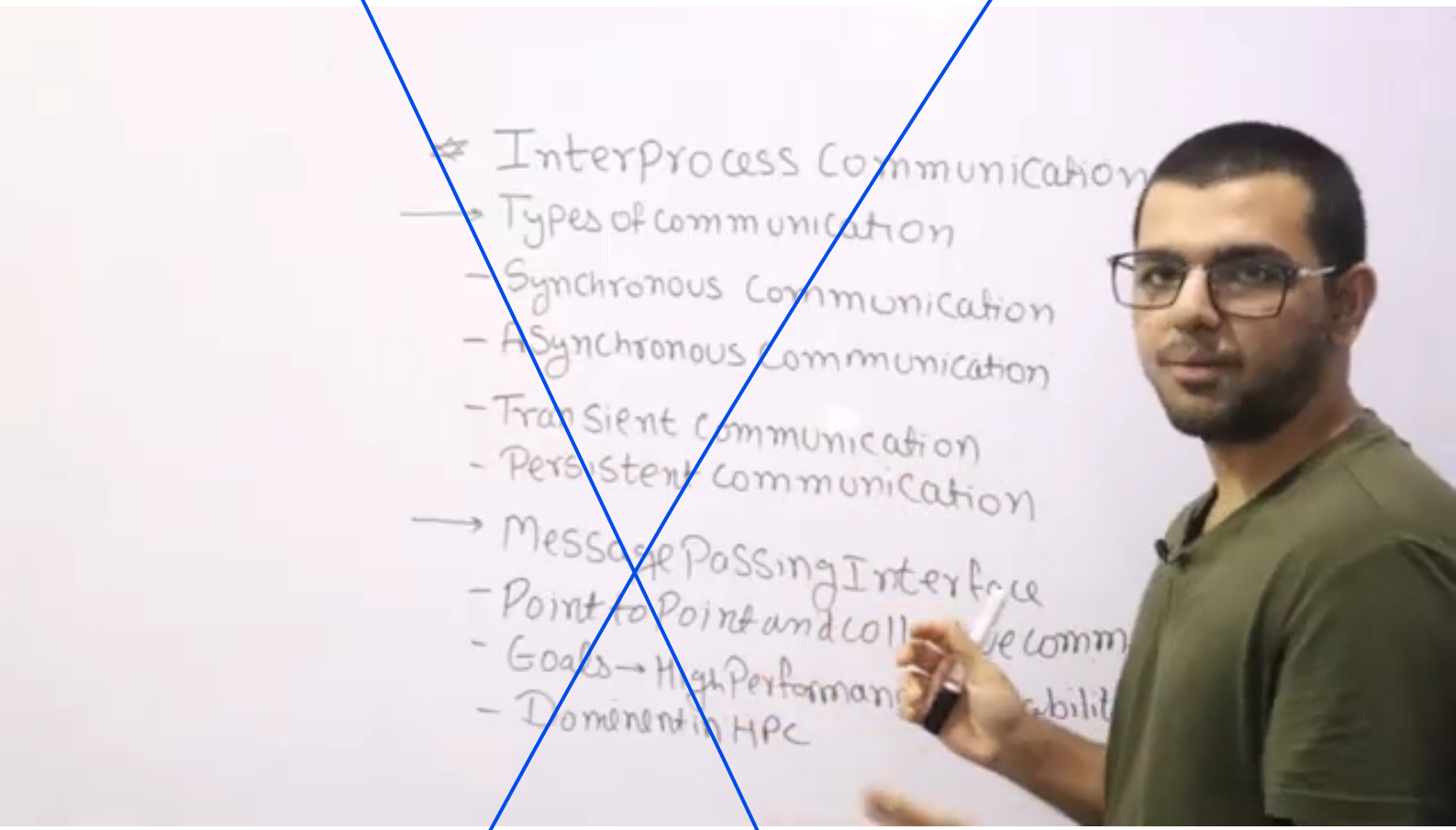


Fig: Communications models, (a) Shared memory, (b) Message Passing.


InterProcess Communication
Types of communication
- Synchronous Communication
- ASynchronous Communication
- Transient Communication
- Persistent Communication
Message Passing Interface

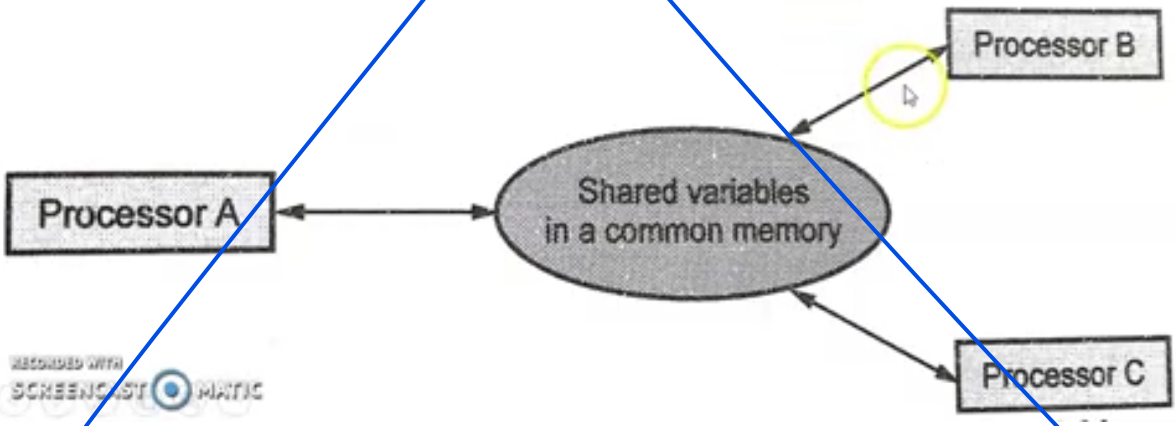

InterProcess Communication
Types of communication
- Synchronous Communication
- ASynchronous Communication
- Transient communication
- Persistent communication
Message Passing Interface
- Point to Point and coll
- Goals → High Performan
- Dominent in HPC

Inter-Process Communication

- In a multiprocessing environment processors implies parallelism by concurrent processing.
- The concurrent processing requires ***sharing of resources*** between the processors and inter-processor communication.
- Basically there are two ways by which inter process communication is achieved
  - Using Shared Variables
  - Using Message Passing

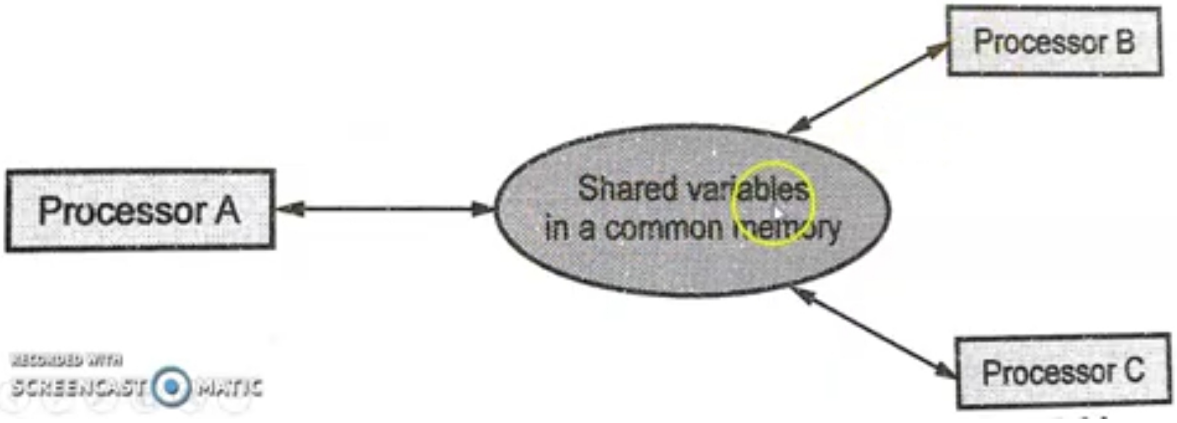
IPC using shared variable

- In this system shared variable are stored in common memory which is accessible to all processors in the system.



IPC using shared variable

- In this system shared variable are stored in common memory which is accessible to all processors in the system.
- While sharing common resources or shared variables ***conflict problem*** may arise.



IPC using shared variable

- It is necessary to ***prevent conflict use of shared resources*** by several processors. This task is done by operating system.

1. Master-slave operating system
2. Separate operating system
3. Distributed operating system

IPC using shared variable

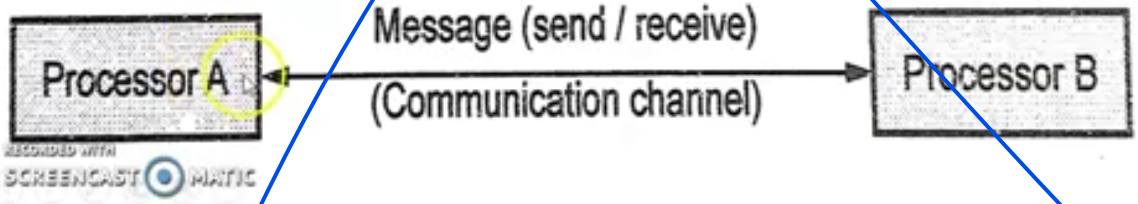
1. **Master-slave operating system**: In this system, processor that executes the operating system function is called ***master*****.** The other processors are called ***slave.*** When slave needs operating system service it request it by interrupting the master.

2. **Separate operating system**: In each processor has entire copy of operating system can be execute operating system functions. This organization is more suitable for ***loosely couple system.***

3. **Distributed operating system**: In this operating system, routines are distributed among the available processors. Such type operating system is also known as ***floating operating system.***

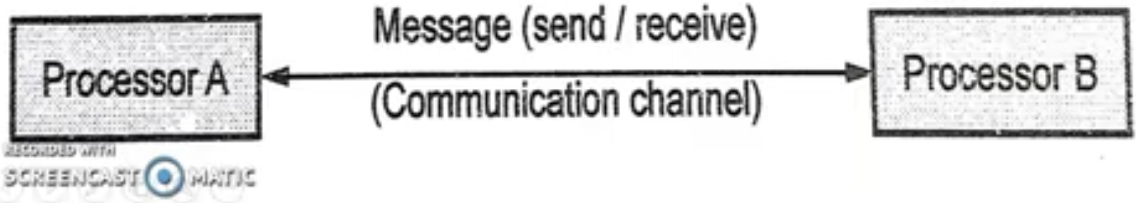
IPC using message passing

- In multiprocessor system with no shared memory we use message passing mechanism to perform inter-process communication.
- When processor wants to communicate with another processor, it uses a special procedure which initiates communication.
- It identifies the destination processor and once source and destination processors are identified a communication channel is established.



IPC using message passing

- In multiprocessor system with no shared memory we use message passing mechanism to perform inter-process communication.
- When processor wants to communicate with another processor, it uses a special procedure which initiates communication.
- It identifies the destination processor and once source and destination processors are identified a communication channel is established.
- A message is then sent through the communication channel.



Inter-Processor Synchronization

- At the higher level of parallelism, the program is partitioned into process that are executed in different processors.
- This technique is called ***concurrent processing.***
- During concurrent processing when two or more processors need the same resource at a time, the ***contention problem*** arise. Such problem can be solved by ***synchronization.***
- To achieve synchronization a set of ***hardware primitives*** are used to automatically read and modified a memory location without any interruption between the two operations.

Inter-Process Synchronization

- Such mechanisms are necessary to protect data from being changed simultaneously by two or more processors. This mechanism is known as ***mutual exclusion***.

- A program sequence which accesses the shared resources, one begun, must complete execution before another processor accesses the same shared address.

- Thus program sequence which accesses the shared memory is known as ***critical section*** of the program.

Inter-Processor Synchronization

- Such mechanisms are necessary to protect data from being changed simultaneously by two or more processors. This mechanism is known as ***mutual exclusion***.
- A program sequence which accesses the shared resources, one begun, must complete execution before another processor accesses the same shared address.
- Thus program sequence which accesses the shared memory is known as ***critical section*** of the program.
- Inter-process communication done by mutual exclusion with ***semaphore*** and mutual exclusion using ***load and store conditional instruction.***

Lecture : Inter Process Communication :-

① What is Inter process Communication ? IPC

→ Inter process Communication is a mechanism that allow processes to Communicates and Synchronize when running Concurrently in an OS.

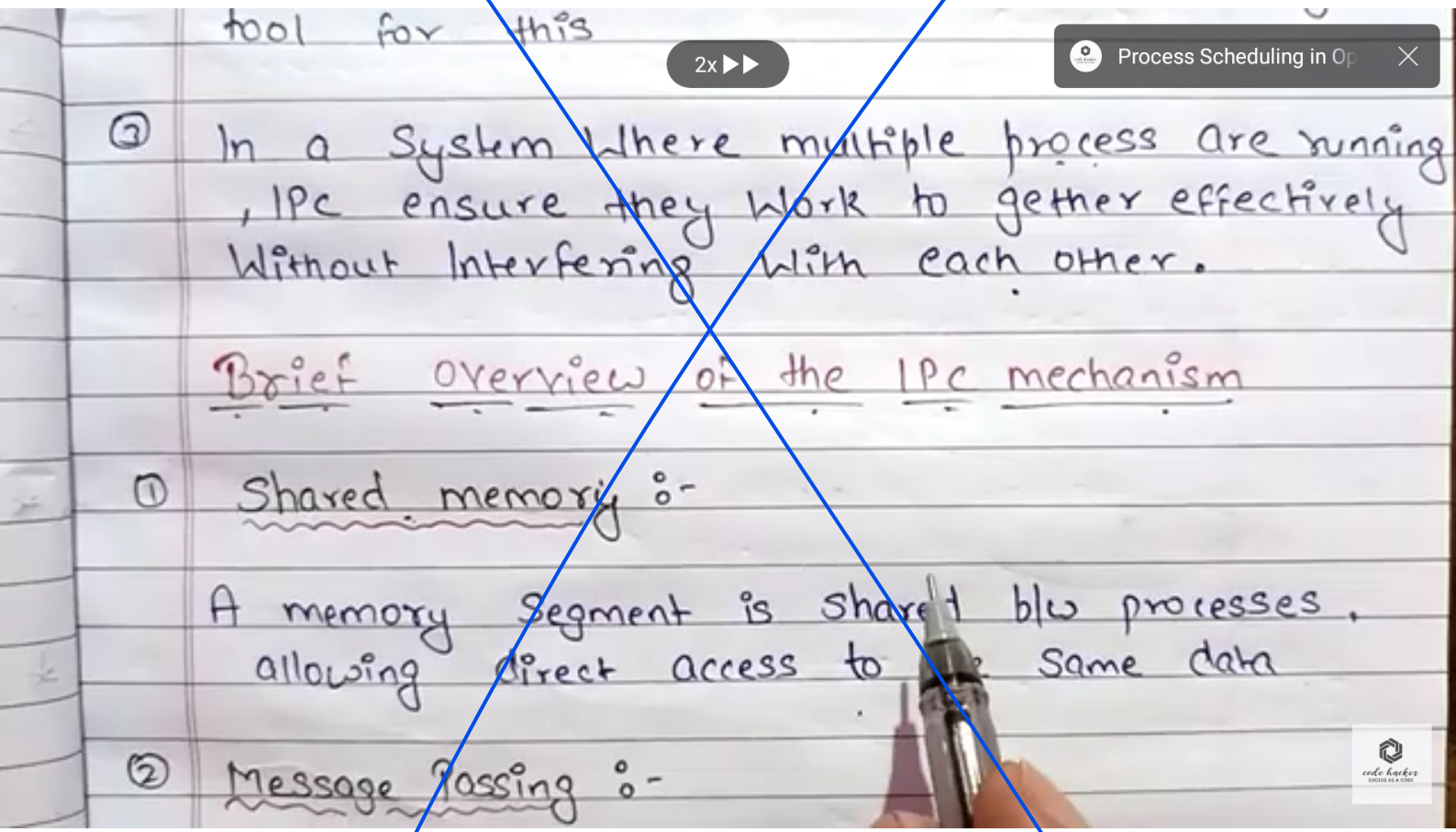
→ Process need to Communicate with each other to share data, Coordinate tasks or notify about events and IPC provide necessary tool for this

allow processes to communicates and synchronize when running concurrently in an OS.

→ Process need to communicate with each other to share data, coordinate tasks or notify about events and IPC provide necessary tool for this

② In a system where multiple process are running, IPC ensure they work together effectively without interfering with each other.

Brief overview of the IPC mechanism


tool for this
2x
Process Scheduling in Op
② In a System where multiple process are running, IPC ensure they work to gether effectively Without Interfering With each other.
Brief overview of the IPC mechanism
① Shared memory :-
A memory segment is shared blw processes, allowing direct access to same data
② Message Passing :-

Brief overview of the IPC mechanism

① Shared memory :-

A memory segment is shared b/w processes, allowing direct access to the same data.

② Message Passing :-

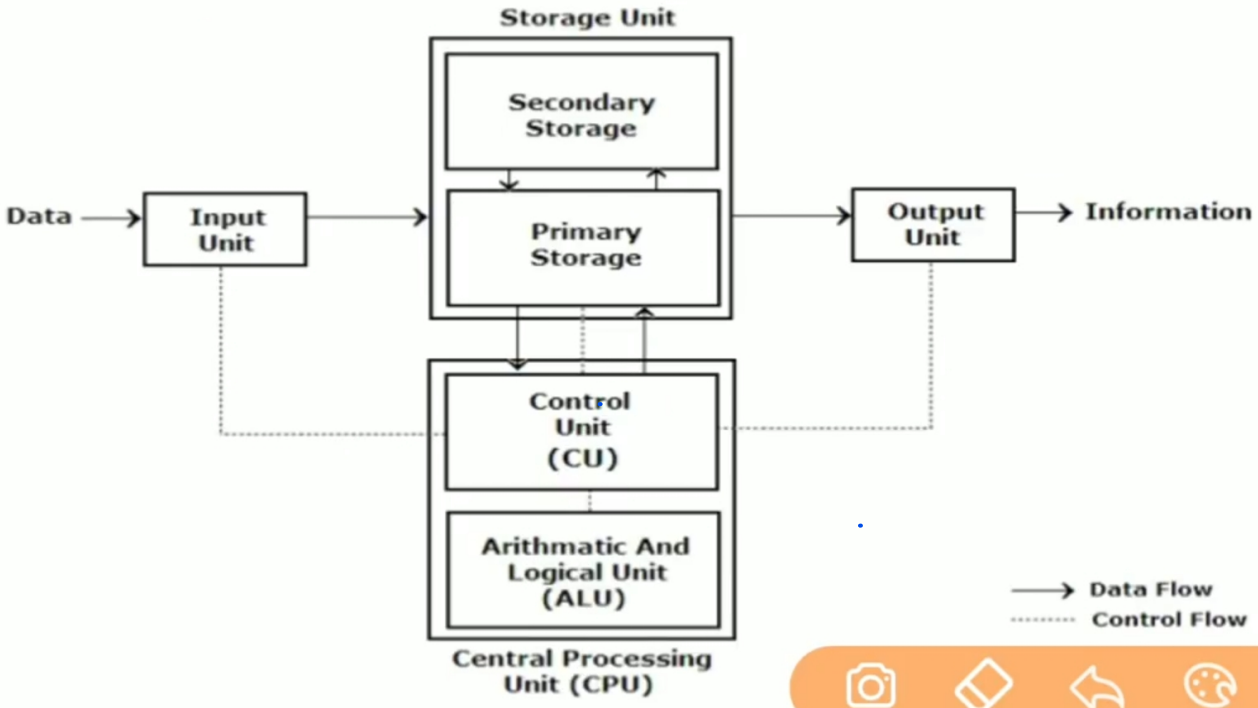
Processes exchange information by sending and receiving messages.

→ A Queue that stores messages from multiple processes until they are retrieved by a receiving process.

⑤ Sockets :-

Used for Communication b/w processes on different machines or network via TCP / IP

Block diagram of computer



Storage Unit

- **Primary memory** is the main memory (Hard disk, RAM) where the operating system resides. Primary storage, also known as main storage or memory, is the area in a computer in which data is stored for quick access by the computer's processor. It is volatile, however, which means when the computer is turned off everything in RAM is deleted
- **Secondary memory** can be external devices like CD, floppy magnetic discs etc.secondary storage device is a non-volatile device that holds data until it is deleted or overwritten. Secondary storage is about two orders of magnitude cheaper than primary storage

- **1.) Arithmetic Logical Unit (ALU) Logical Unit**:

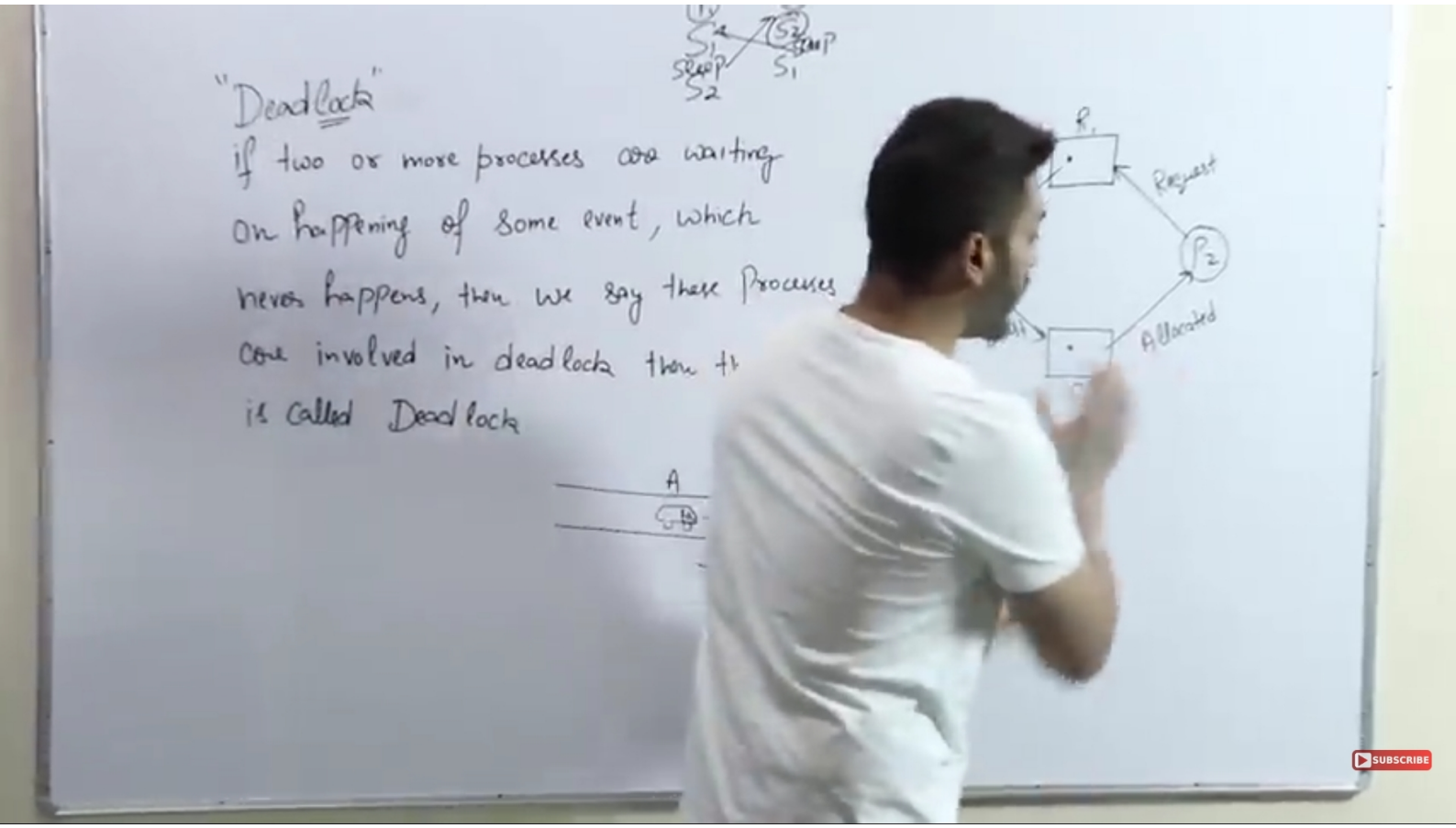
After you enter data through the input device it is stored in the primary storage unit. The actual processing of the data and instruction are performed by Arithmetic Logical Unit. After processing the output is returned back to storage unit for further processing or getting stored.

- **2.) Control Unit**

Acts like the supervisor seeing that things are done in proper fashion. Control Unit is responsible for co-ordinating various operations

- **3.) Central Processing Unit (CPU)**

The ALU and the CU of a computer system are jointly known as the central processing unit. You may call CPU as the brain of any computer system.


"Deadlock"
if two or more processes are waiting
on happening of some event, which
never happens, then we say these Processes
are involved in deadlock then th
is called Deadlock
S1
S2
S2
S1
P1
Request
P2
Allocated
A
SUBSCRIBE

"Deadlock"

if two or more pr

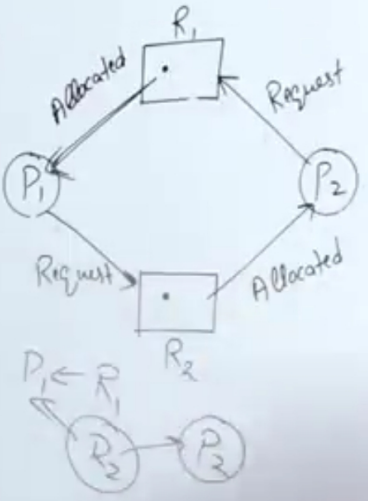
On happening of some of

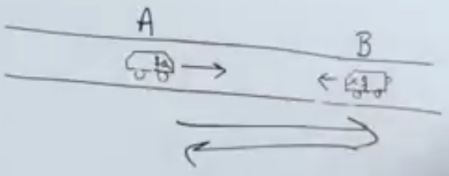
1) Mutual Exclusion
2) No Preemption
3) Hold & Wait
4) Circular Wait

never happens, then we say

core involved in deadlock then that state

is called Deadlock.





Like

What is deadlock?

⇒ Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

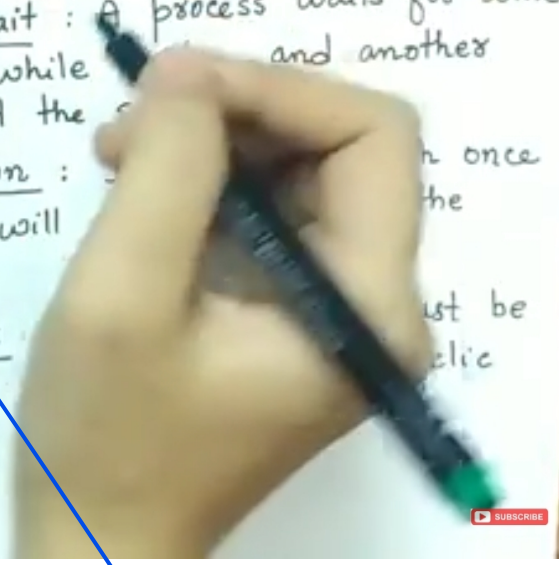
* Necessary condition for Deadlocks

1) Mutual Exclusion : If two process cannot use same ~~process~~ resource at same time.

2) Hold and Wait : A process waits for some resources while and another resource at the

3) No preemption : once Scheduled will the completion.

4) Circular wait ust be waiting for clic manner.



What is deadlock?

⇒ Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

* Necessary condition for Deadlocks

1) Mutual Exclusion : If Two process cannot use same ~~process~~ resource at same time.

2) Hold and Wait : A process waits for some resources while holding and another resource at the same time.

3) No preemption : The process which once scheduled will be executed till the completion.

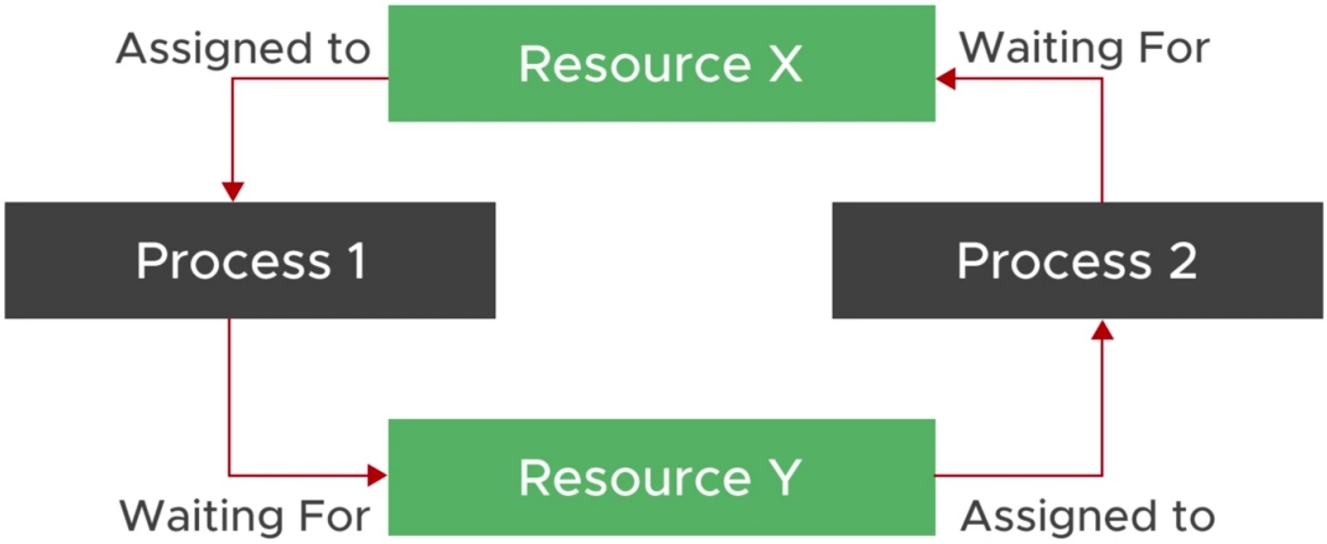
4) Circular wait : All the processes must be waiting for the resource in a cyclic manner.

Definition

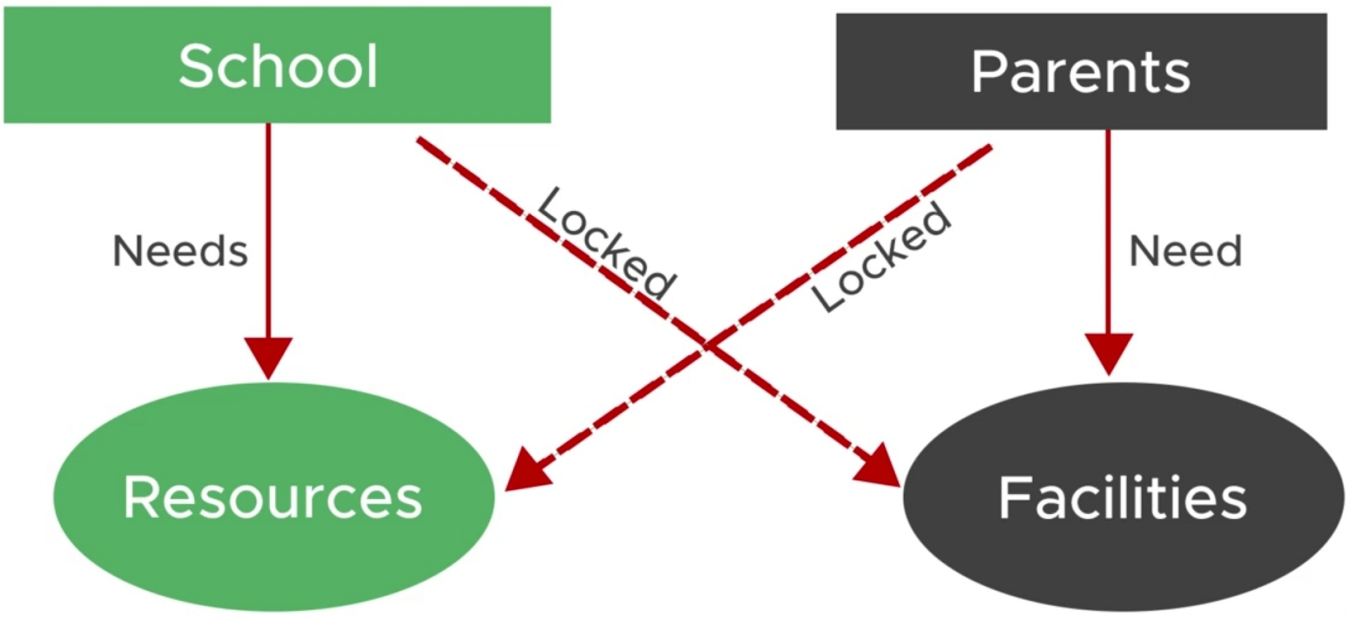
Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Example:
You can't get job without experience and experience can only come when you do some job. So here we end up in a dead lock.

Example Diagram



Example Diagram



Conditions Necessary for Deadlock

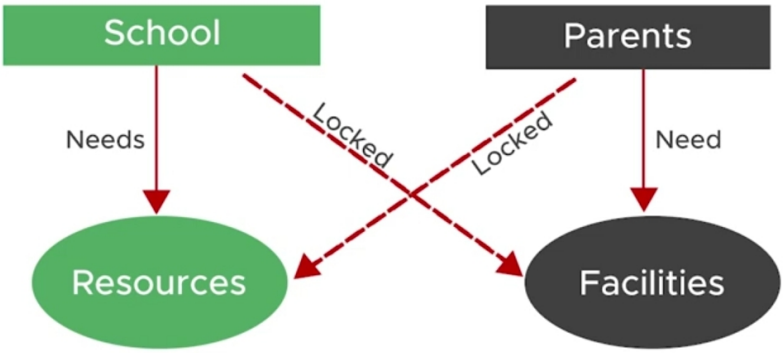
- Mutual Exclusion
- Hold and Wait
- No Preemption
- Circular Wait

Hold and Wait

Hold and Wait: If a process holds some resource and waits for some other resources. If this is not the case then the cycle of deadlock which we saw in the first diagram will not get completed and no deadlock will occurs.

Example:

The above school example is a classic example of Hold and Wait as school holds facilities and parents hold resources and wait for resources and facilities respectively.



No Preemption

No Preemption: A resource cannot be taken from a process unless the process releases the resource. If preemption was allowed deadlock would never occur because then no process would have been able to hold a resource for long amount of time.

For Example:

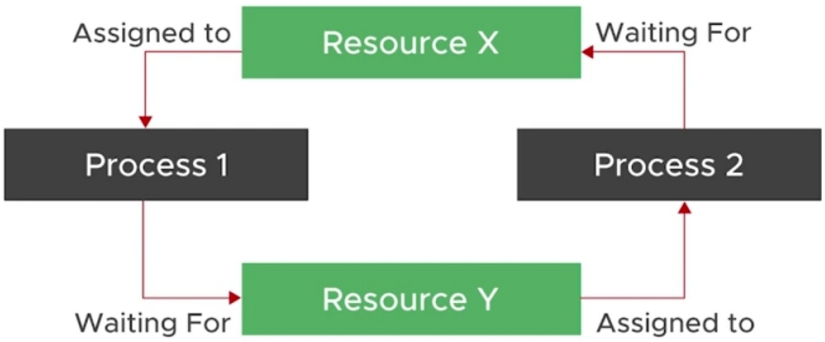
If your speaker is running an audio and after some time you click on some other audio it starts playing but in case if no preemption was allowed we would had to wait for the first audio to end and if in case it was on loop we will end up in a deadlock.

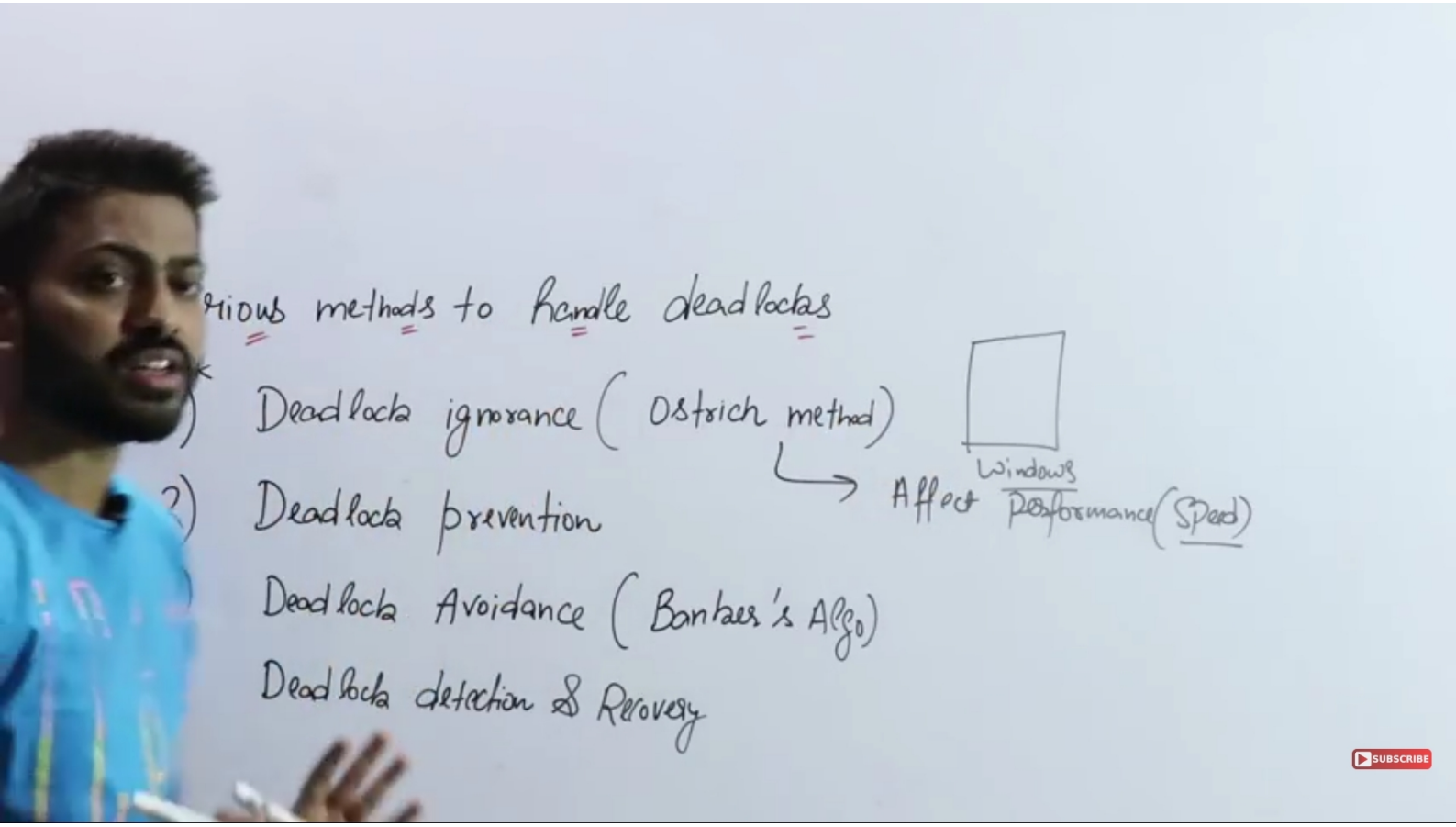
Circular Wait

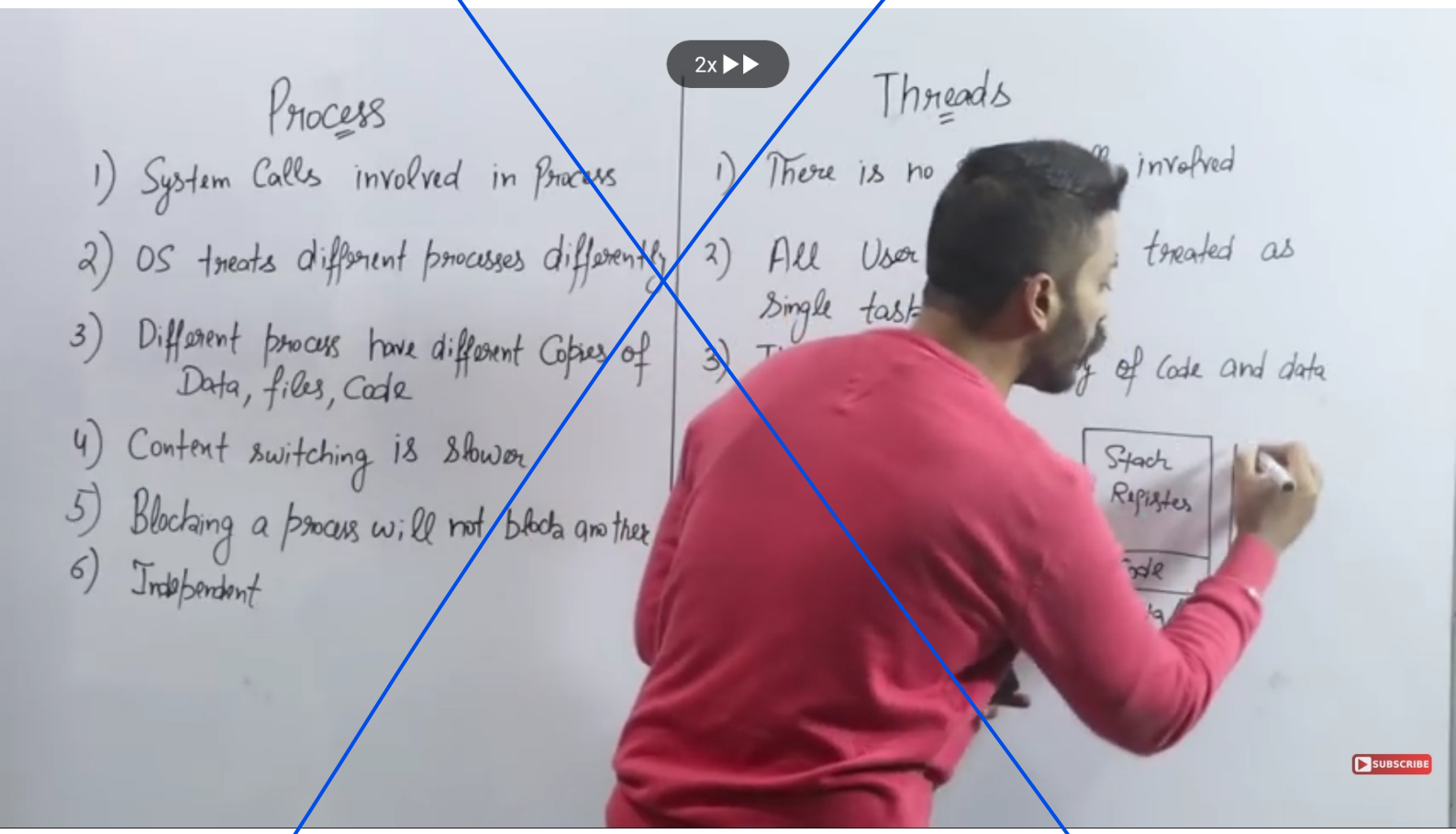
Circular Wait: A set of processes are waiting for each other in circular form.

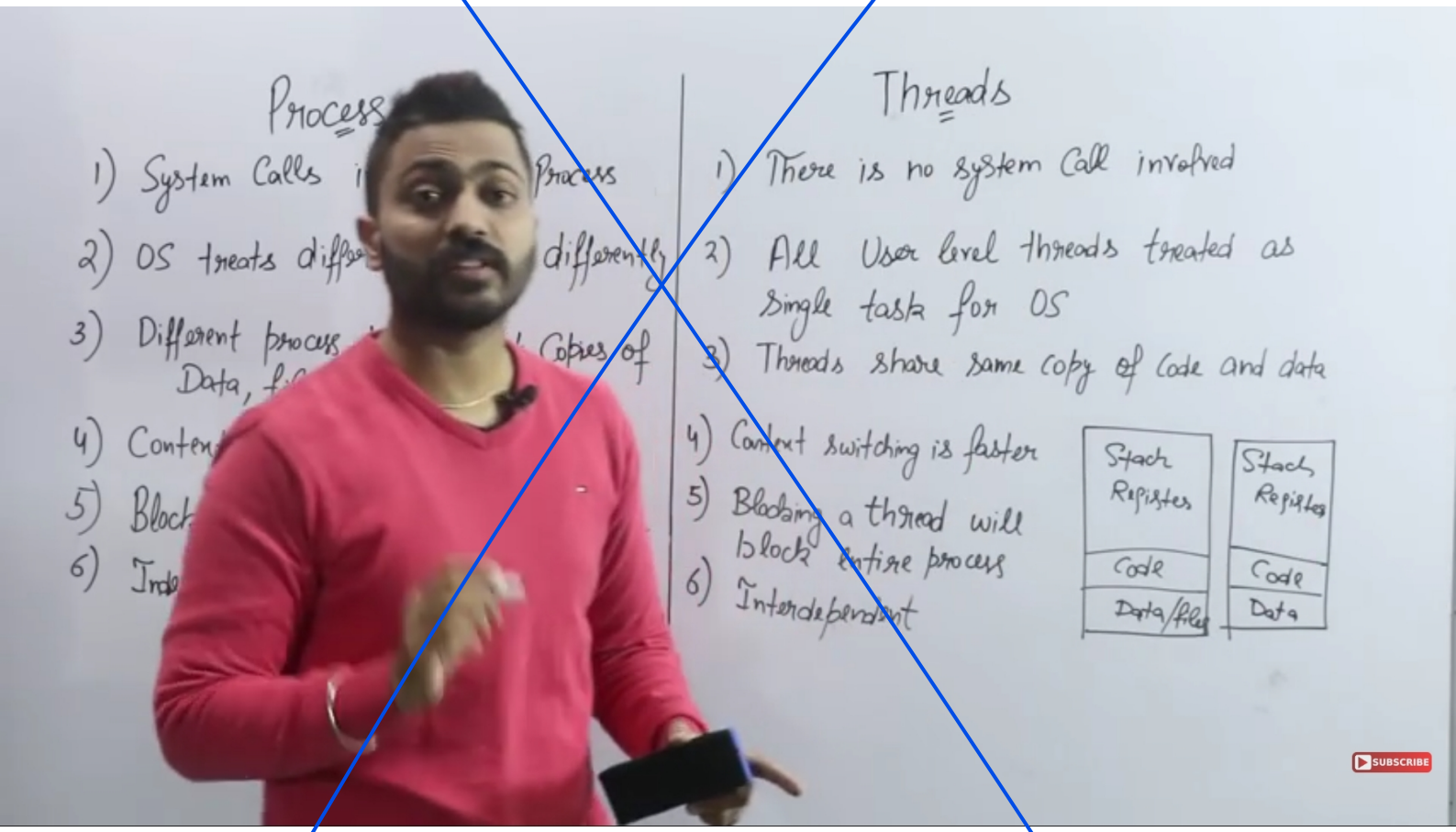
For Example:

If we extend the first example to multiple processes and resources waiting for each others held resources in a circular fashion we will end up in a circular wait condition.




rious methods to handle deadlocks
1) Deadlock ignorance (Ostrich method)
Affect Windows Performance (Speed)
2) Deadlock prevention
Deadlock Avoidance (Banker's Algo)
Deadlock detection & Recovery
SUBSCRIBE


2x
Process
1) System Calls involved in Process
2) OS treats different processes differently
3) Different process have different Copies of Data, files, Code
4) Content switching is slower
5) Blocking a process will not block another
6) Independent
Threads
1) There is no involved
2) All User treated as Single task
3) of Code and data
Stack
Registers
SUBSCRIBE

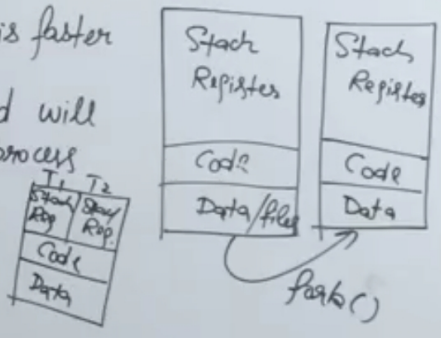

Threads
1) There is no system Call involved
2) All User level threads treated as Single task for OS
3) Threads share same copy of Code and data
4) Context switching is faster
5) Blocking a thread will block entire process
6) Interdependent
Stack Registers
Code
Data/files
Stack Registers
Code
Data
SUBSCRIBE

Process

1) System Calls involved in Process
2) OS treats different processes differently
3) Different process have different Copies of Data, files, Code
4) Content switching is slower
5) Blocking a process will not block another
6) Independent

Threads (User Level)

1) There is no system Call involved
2) All User level threads treated as single task for OS
3) Threads share same copy of Code and data
4) Context switching is faster
5) Blocking a thread will block entire process
6) Interdependent



Threads

A thread is a basic unit of CPU utilization.

It comprises

- A thread ID
- A program counter
- A register set and
- A stack

It shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals.

NESO ACADEMY

Threads

A thread is a basic unit of CPU utilization.

It comprises

- A thread ID
- A program counter
- A register set and
- A stack

It shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals.

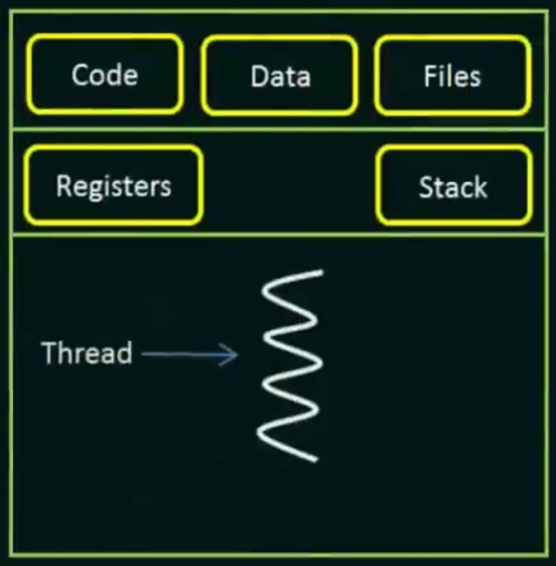
A traditional / heavyweight process has a single thread of control.

If a process has multiple threads of control, it can perform more than one task at a time.

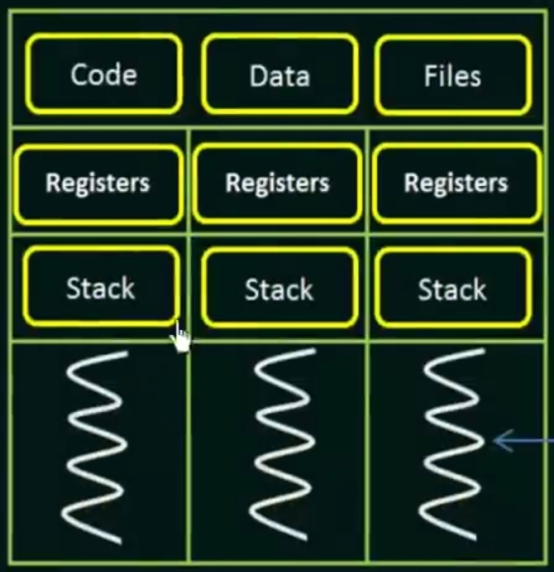
It shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals.

A traditional / heavyweight process has a single thread of control.

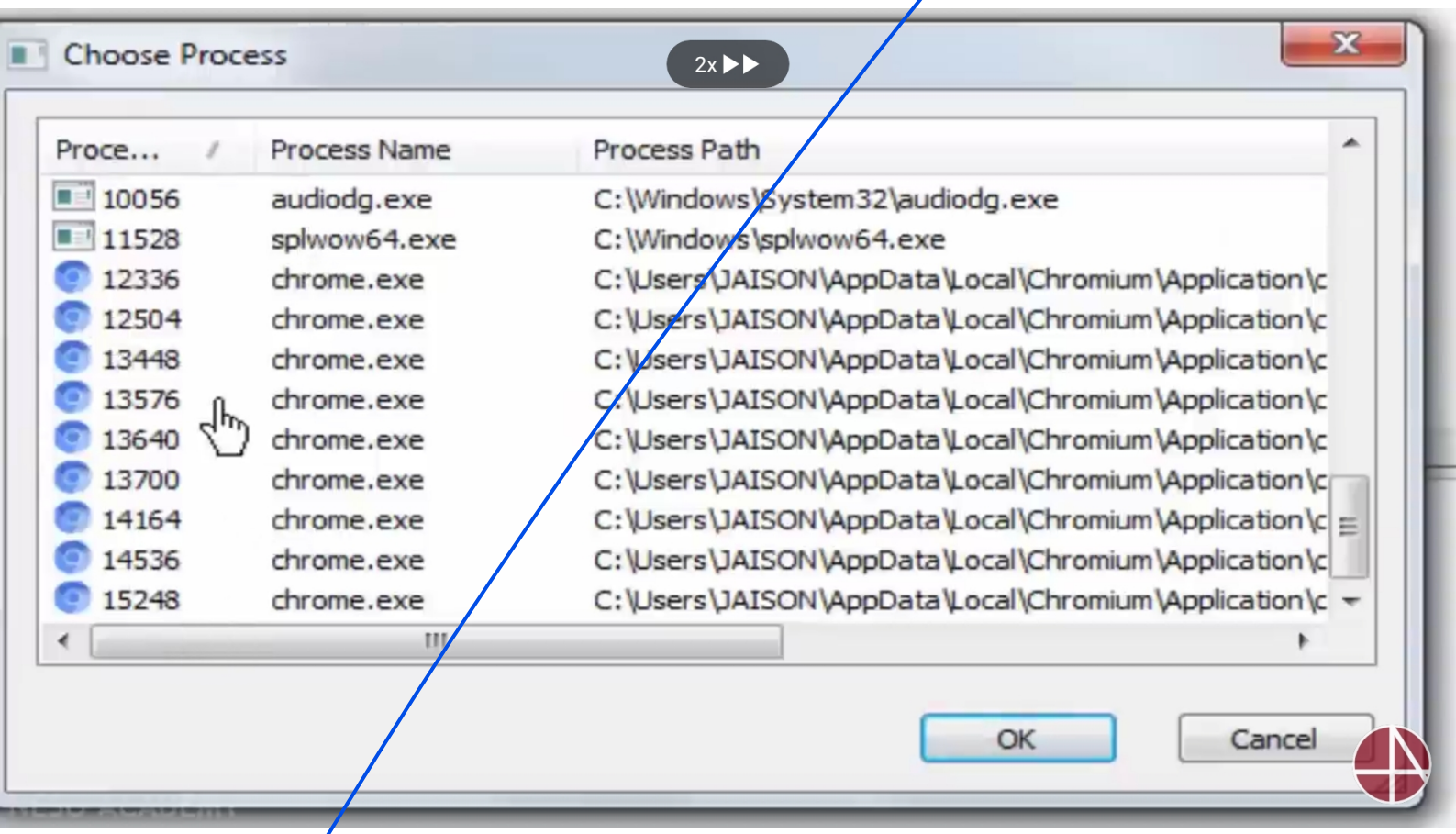
If a process has multiple threads of control, it can perform more than one task at a time.

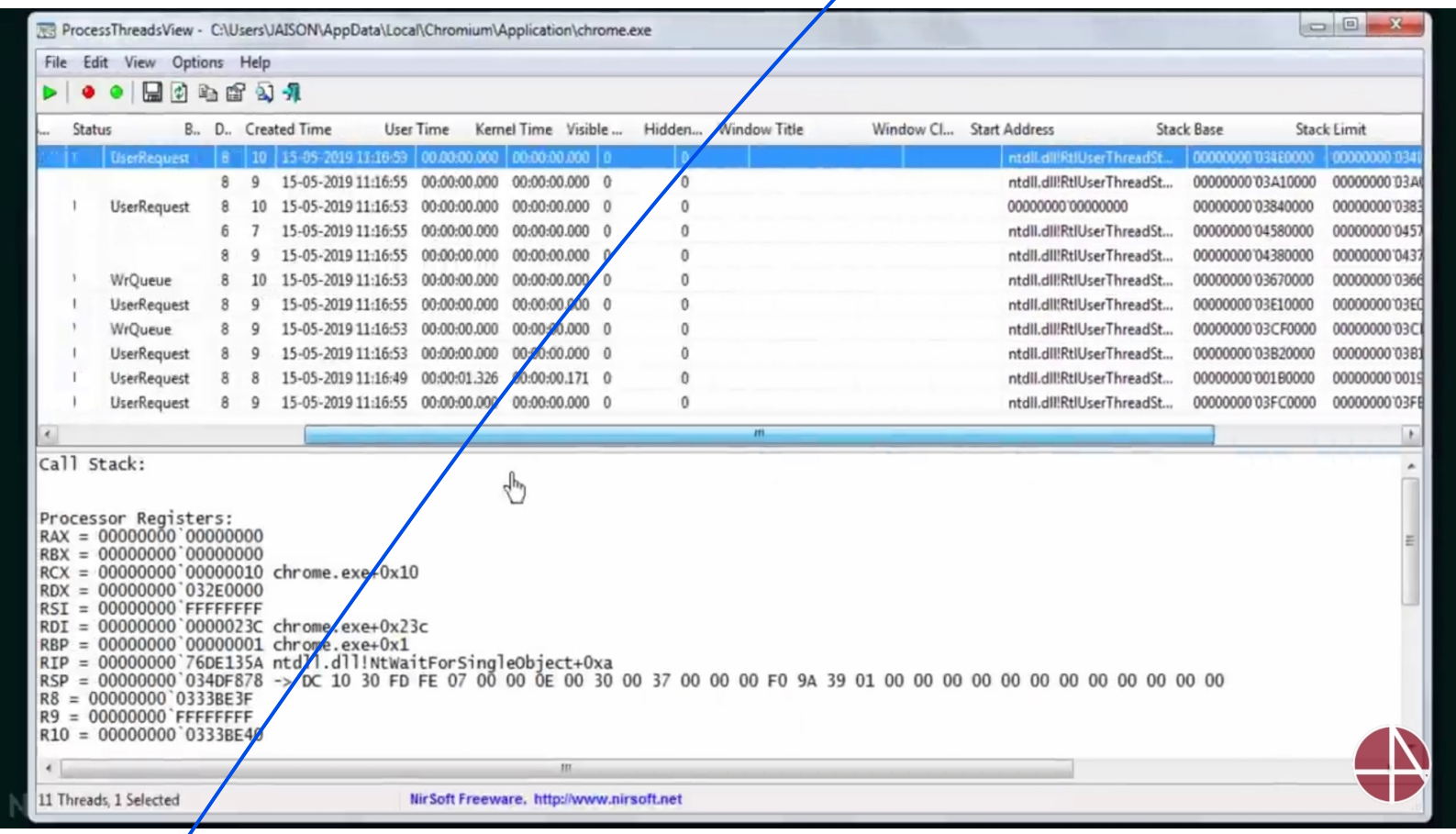


Single-threaded process



Multi-threaded process


Choose Process
2x
Proce...
Process Name
Process Path
10056 audiodg.exe C:\Windows\System32\audiodg.exe
11528 splwow64.exe C:\Windows\splwow64.exe
12336 chrome.exe C:\Users\JAISON\AppData\Local\Chromium\Application\c
12504 chrome.exe C:\Users\JAISON\AppData\Local\Chromium\Application\c
13448 chrome.exe C:\Users\JAISON\AppData\Local\Chromium\Application\c
13576 chrome.exe C:\Users\JAISON\AppData\Local\Chromium\Application\c
13640 chrome.exe C:\Users\JAISON\AppData\Local\Chromium\Application\c
13700 chrome.exe C:\Users\JAISON\AppData\Local\Chromium\Application\c
14164 chrome.exe C:\Users\JAISON\AppData\Local\Chromium\Application\c
14536 chrome.exe C:\Users\JAISON\AppData\Local\Chromium\Application\c
15248 chrome.exe C:\Users\JAISON\AppData\Local\Chromium\Application\c
OK
Cancel


ProcessThreadsView - C:\Users\JAISON\AppData\Local\Chromium\Application\chrome.exe
File Edit View Options Help
Status B.. D.. Created Time User Time Kernel Time Visible ... Hidden... Window Title Window Cl... Start Address Stack Base Stack Limit
UserRequest 8 10 15-05-2019 11:16:53 00:00:00.000 00:00:00.000 0 0 ntdll.dll!RtlUserThreadSt... 00000000`034E0000
8 9 15-05-2019 11:16:55 00:00:00.000 00:00:00.000 0 0 ntdll.dll!RtlUserThreadSt... 00000000`03A10000
UserRequest 8 10 15-05-2019 11:16:53 00:00:00.000 00:00:00.000 0 0 00000000`00000000 00000000`03840000
6 7 15-05-2019 11:16:55 00:00:00.000 00:00:00.000 0 0 ntdll.dll!RtlUserThreadSt... 00000000`04580000
8 9 15-05-2019 11:16:55 00:00:00.000 00:00:00.000 0 0 ntdll.dll!RtlUserThreadSt... 00000000`04380000
WrQueue 8 10 15-05-2019 11:16:53 00:00:00.000 00:00:00.000 0 0 ntdll.dll!RtlUserThreadSt... 00000000`03670000
UserRequest 8 9 15-05-2019 11:16:55 00:00:00.000 00:00:00.000 0 0 ntdll.dll!RtlUserThreadSt... 00000000`03E10000
WrQueue 8 9 15-05-2019 11:16:53 00:00:00.000 00:00:00.000 0 0 ntdll.dll!RtlUserThreadSt... 00000000`03CF0000
UserRequest 8 9 15-05-2019 11:16:53 00:00:00.000 00:00:00.000 0 0 ntdll.dll!RtlUserThreadSt... 00000000`03B20000
UserRequest 8 8 15-05-2019 11:16:49 00:00:01.326 00:00:00.171 0 0 ntdll.dll!RtlUserThreadSt... 00000000`001B0000
UserRequest 8 9 15-05-2019 11:16:55 00:00:00.000 00:00:00.000 0 0 ntdll.dll!RtlUserThreadSt... 00000000`03FC0000
Call Stack:
Processor Registers:
RAX = 00000000`00000000
RBX = 00000000`00000000
RCX = 00000000`00000010 chrome.exe+0x10
RDX = 00000000`032E0000
RSI = 00000000`FFFFFFFF
RDI = 00000000`0000023C chrome.exe+0x23c
RBP = 00000000`00000001 chrome.exe+0x1
RIP = 00000000`76DE135A ntdll.dll!NtWaitForSingleObject+0xa
RSP = 00000000`034DF878 -> DC 10 30 FD FE 07 00 00 0E 00 30 00 37 00 00 00 F0 9A 39 01 00 00 00 00 00 00 00 00 00 00 00
R8 = 00000000`0333BE3F
R9 = 00000000`FFFFFFFF
R10 = 00000000`0333BE40
11 Threads, 1 Selected
NirSoft Freeware. http://www.nirsoft.net

Benefits

The benefits of multithreaded programming can be broken down into four major categories:

Responsiveness

Multithreading an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.

Resource sharing

By default, threads share the memory and the resources of the process to which they belong. The benefit of sharing code and data is that it allows an application to have several different threads of activity within the same address space.

Economy

Allocating memory and resources for process creation is costly. Because threads share resources of the process to which they belong, it is more economical to create and context-switch threads.

The benefits of multithreaded programming can be broken down into four major categories:

Responsiveness

Multithreading an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.

Resource sharing

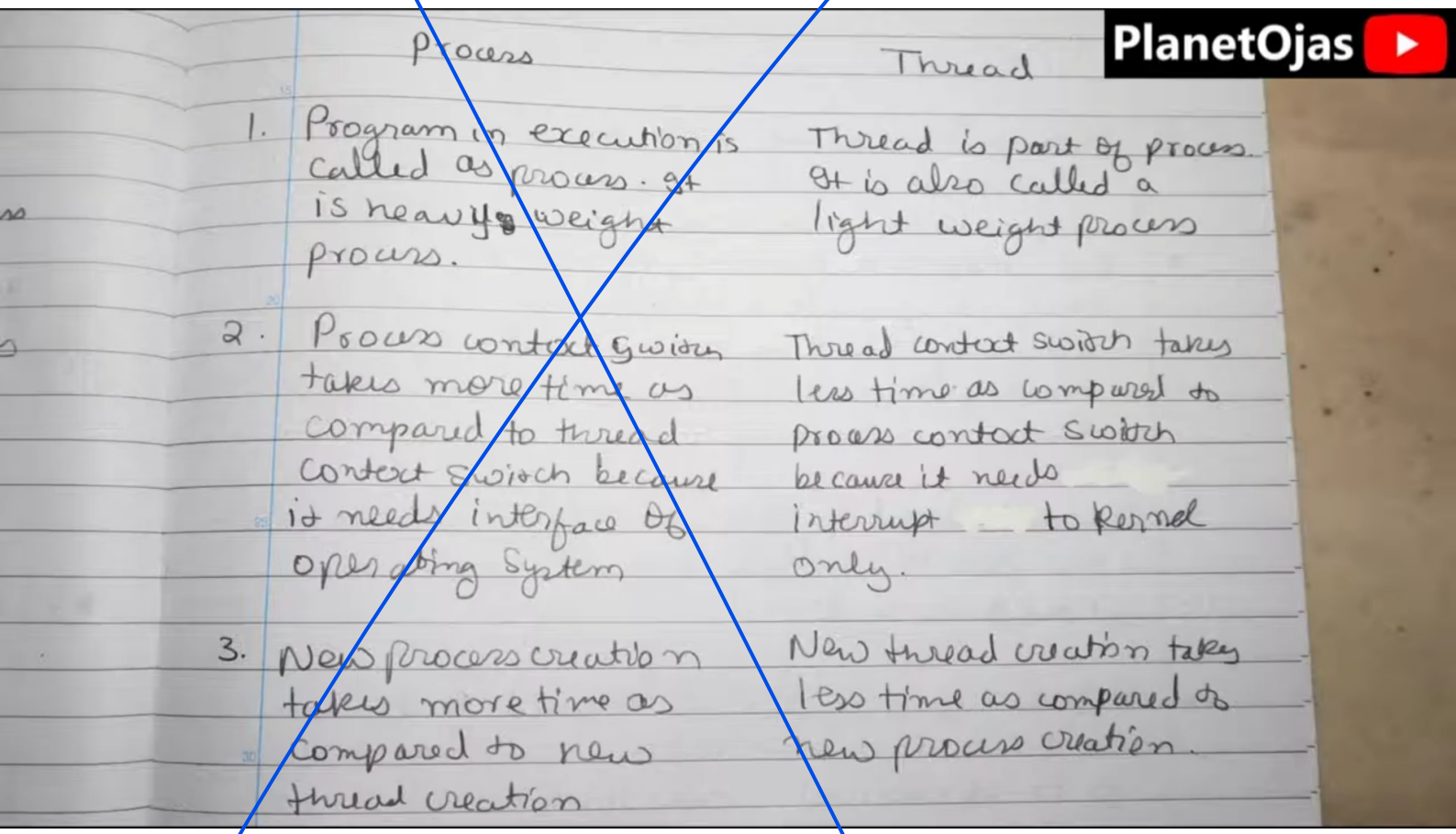
By default, threads share the memory and the resources of the process to which they belong. The benefit of sharing code and data is that it allows an application to have several different threads of activity within the same address space.

Economy

Allocating memory and resources for process creation is costly. Because threads share resources of the process to which they belong, it is more economical to create and context-switch threads.

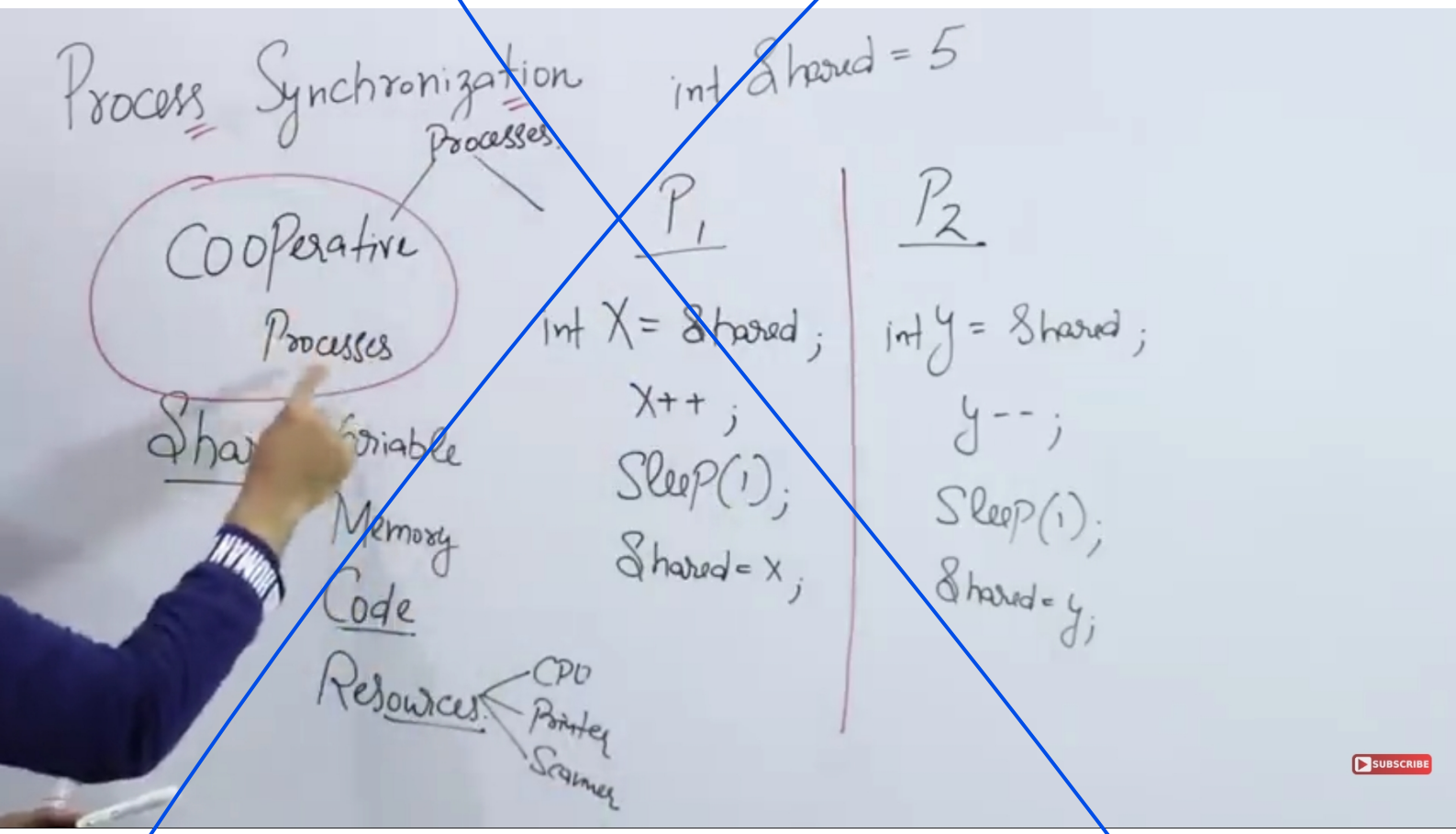
Utilization of multiprocessor architectures

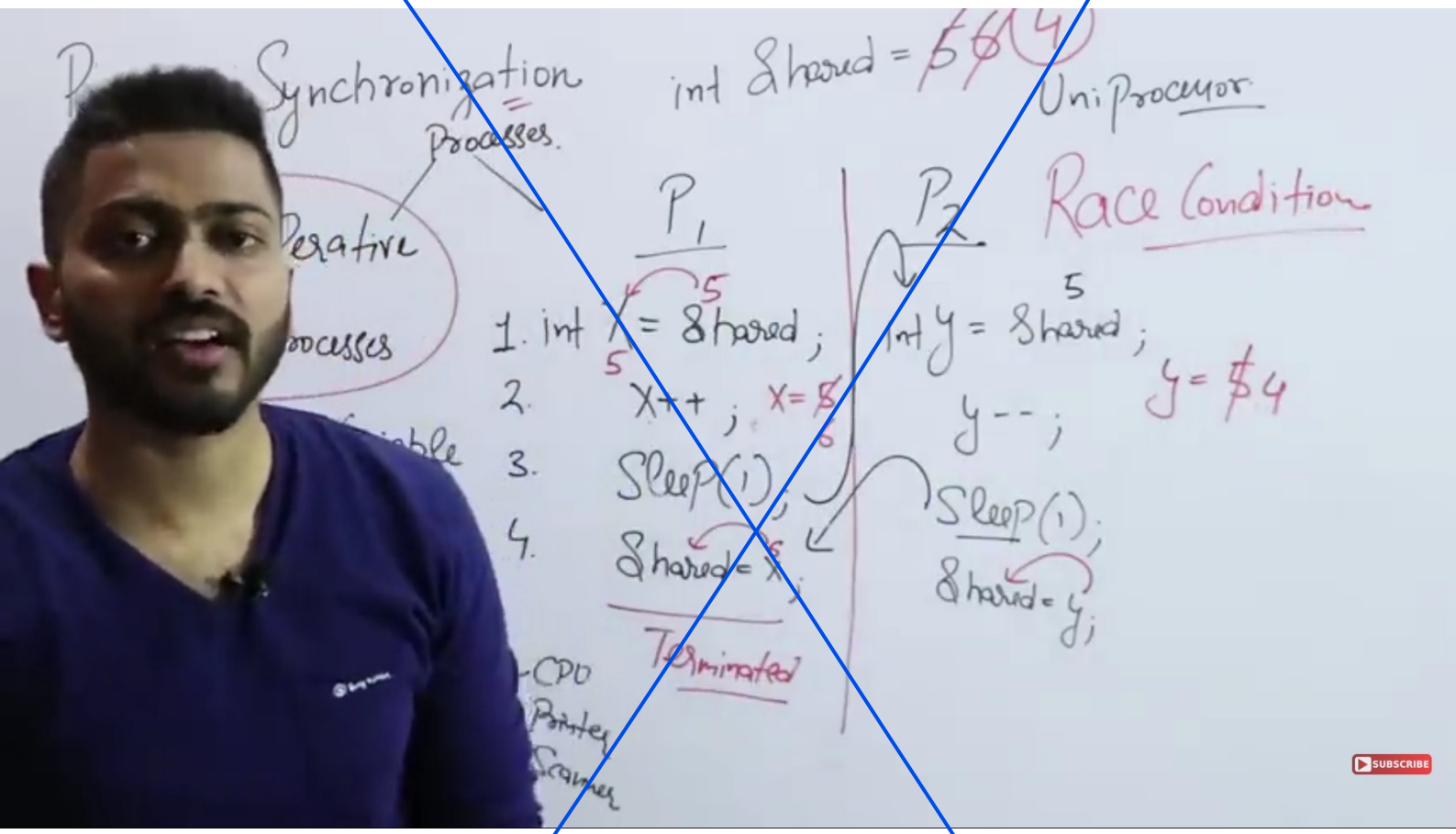
The benefits of multithreading can be greatly increased in a multiprocessor architecture, where threads may be running in parallel on different processors. A single-threaded process can only run on one CPU, no matter how many are available. Multithreading on a multi-CPU machine increases concurrency.



Process	Thread
1. Program in execution is called as process. It is heavy weight process.	Thread is part of process. It is also called a light weight process
2. Process context switch takes more time as compared to thread context switch because it needs interface of operating System	Thread context switch takes less time as compared to process context switch because it needs interrupt to Kernel only.
3. New process creation takes more time as compared to new thread creation	New thread creation takes less time as compared to new process creation.

4.	New process termination takes more time as compared to new thread termination	New thread termination takes more time as compared to new process termination
5.	In process based implementation, if one process is blocked no other server process can execute until the first process is unblocked	In multithreaded server implementation, if one thread is blocked and waiting, second thread in the same process could execute.


Process Synchronization
Processes
Cooperative Processes
Shared Memory
Code
Resources
CPU
Printer
Scanner
int Shared = 5
P1
int X = Shared;
X++;
Sleep(1);
Shared = X;
P2
int y = Shared;
y--;
Sleep(1);
Shared = y;
SUBSCRIBE

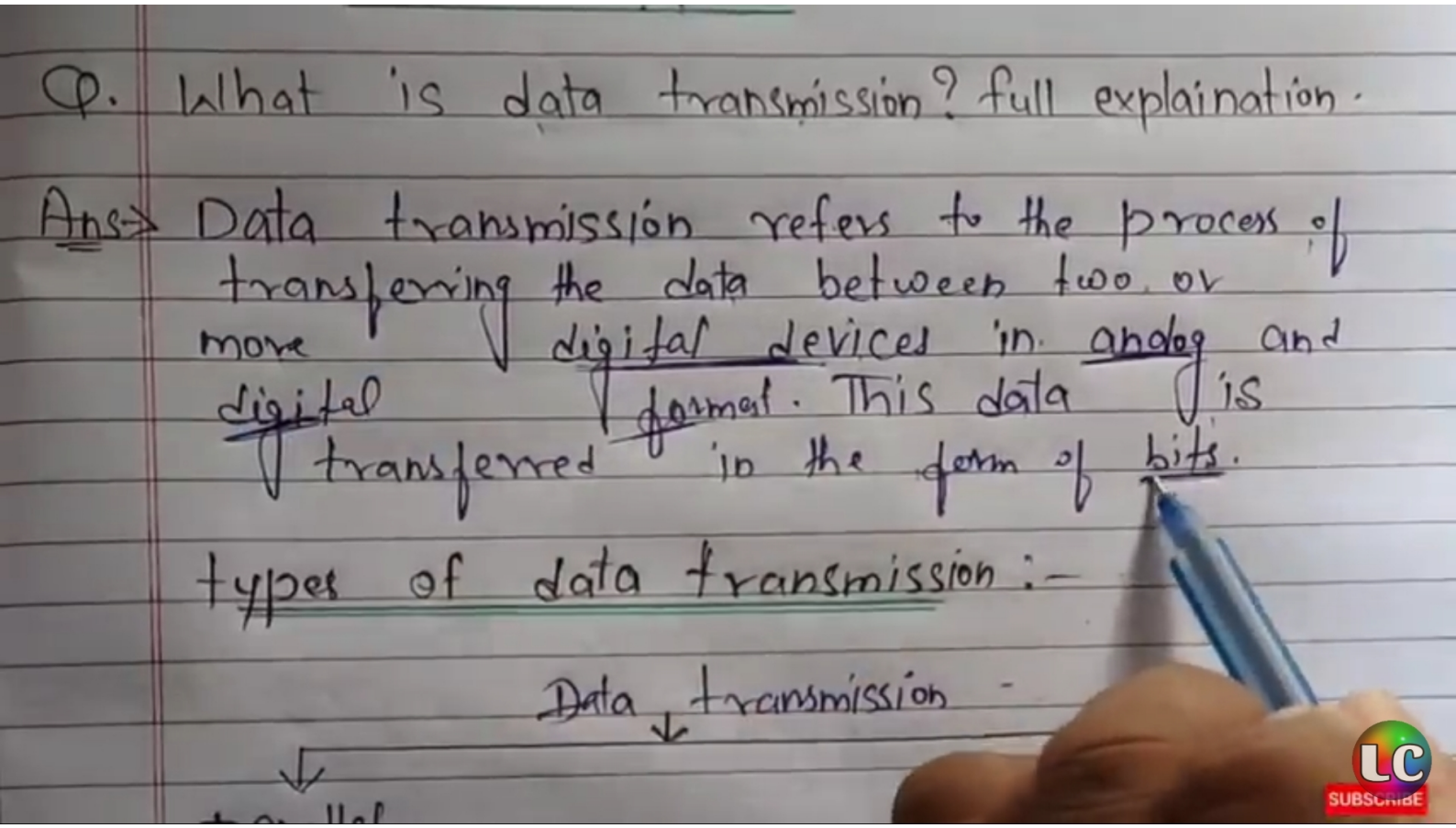

Synchronization
Processes.
int &hared = 5 4
Uniprocessor
Race Condition
P1
P2
1. int x = &hared ;
2. x++ ;
3. Sleep(1);
4. &hared = x;
int y = &hared ;
y--;
Sleep(1);
&hared = y;
y = 5 4
Terminated
SUBSCRIBE

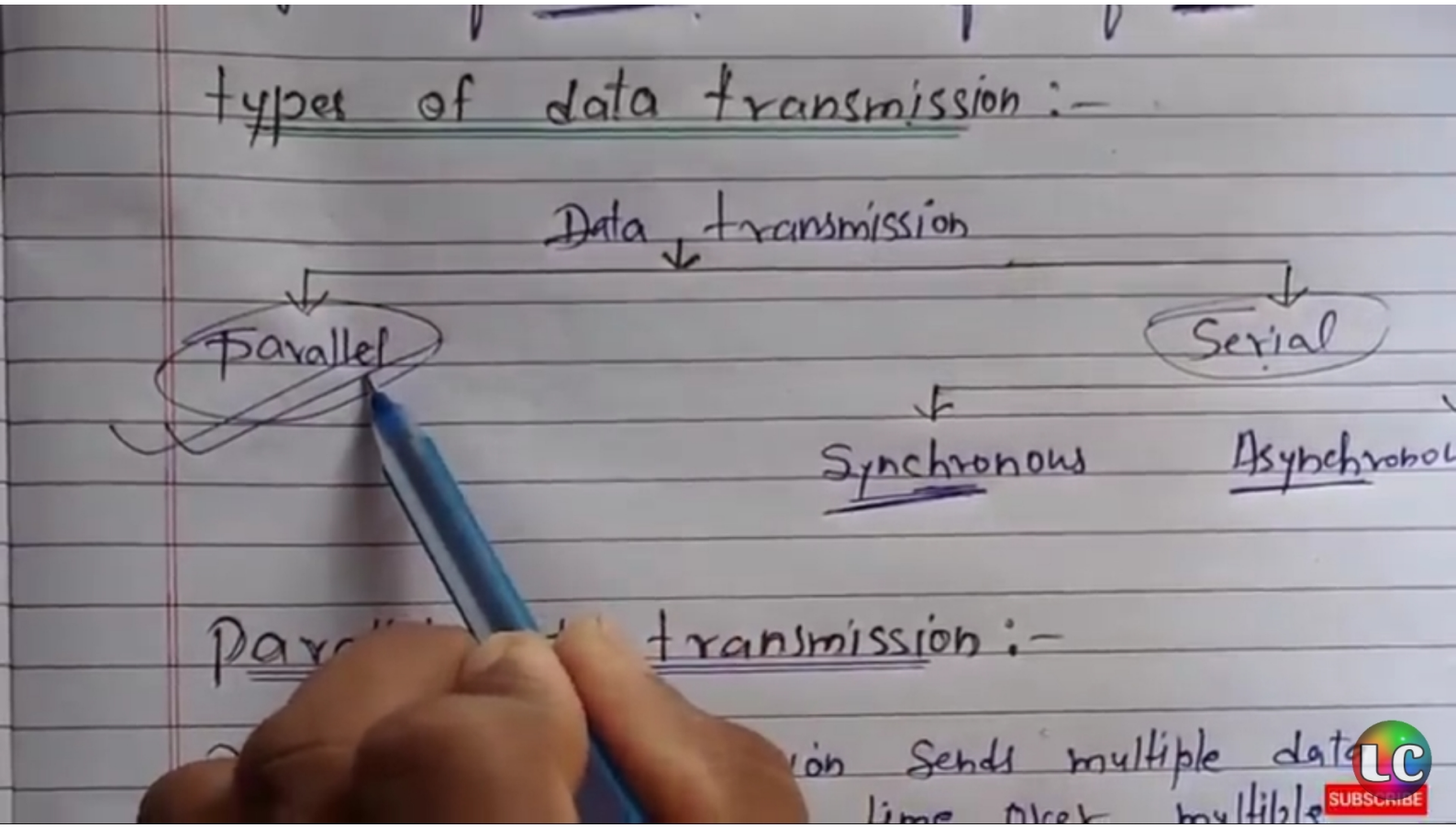
Q. What is data transmission? full explaination.

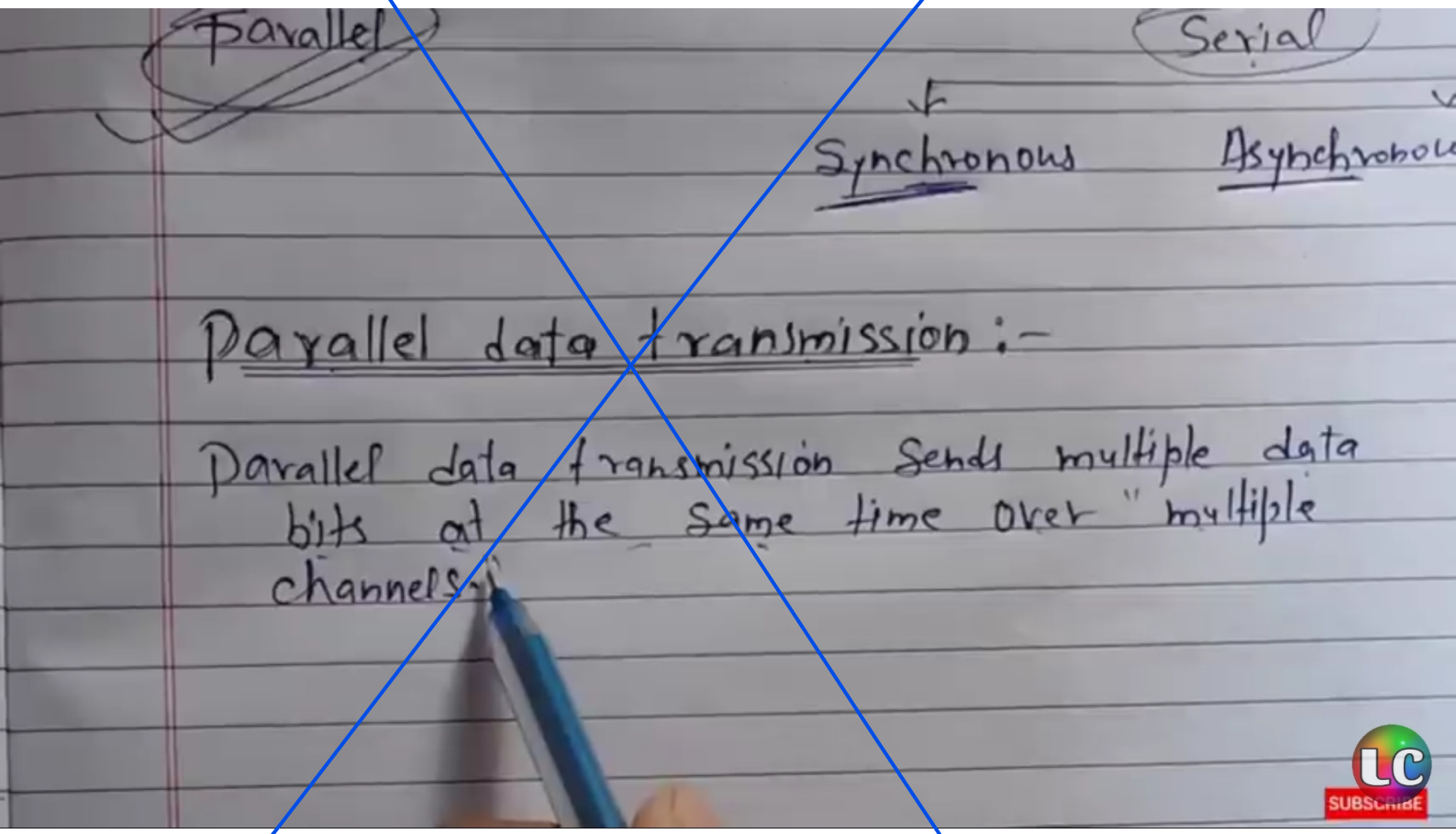
Ans⇒ Data transmission refers to the process of transferring the data between two or more digital devices in analog and digital format. This data is transferred in the form of bits.

types of data transmission :-

Data transmission



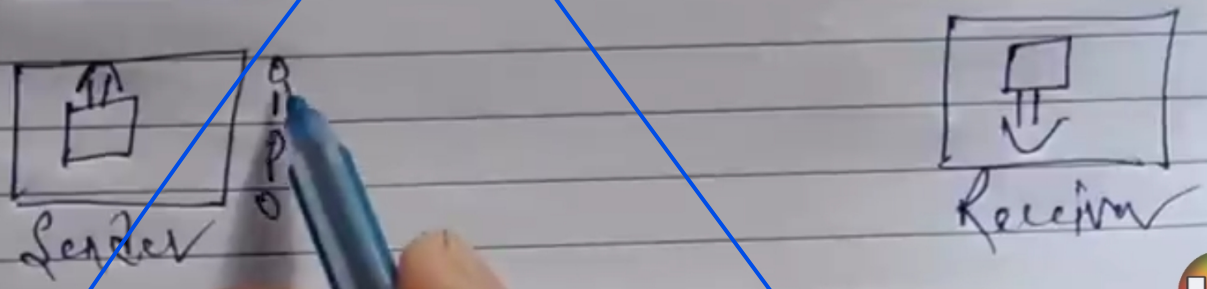

types of data transmission :-
Data transmission
Parallel
Serial
Synchronous
Asynchronou
transmission :-
ion Sends multiple data
SUBSCRIBE


Parallel
Serial
Synchronous
Asynchronous
Parallel data transmission :-
Parallel data transmission sends multiple data bits at the same time over multiple channels.
LC
SUBSCRIBE

Synchronous Asynchronou

Parallel data transmission :-

Parallel data transmission sends multiple data bits at the same time over "multiple channels."



Synchronous Asynchronou

Parallel data transmission :-

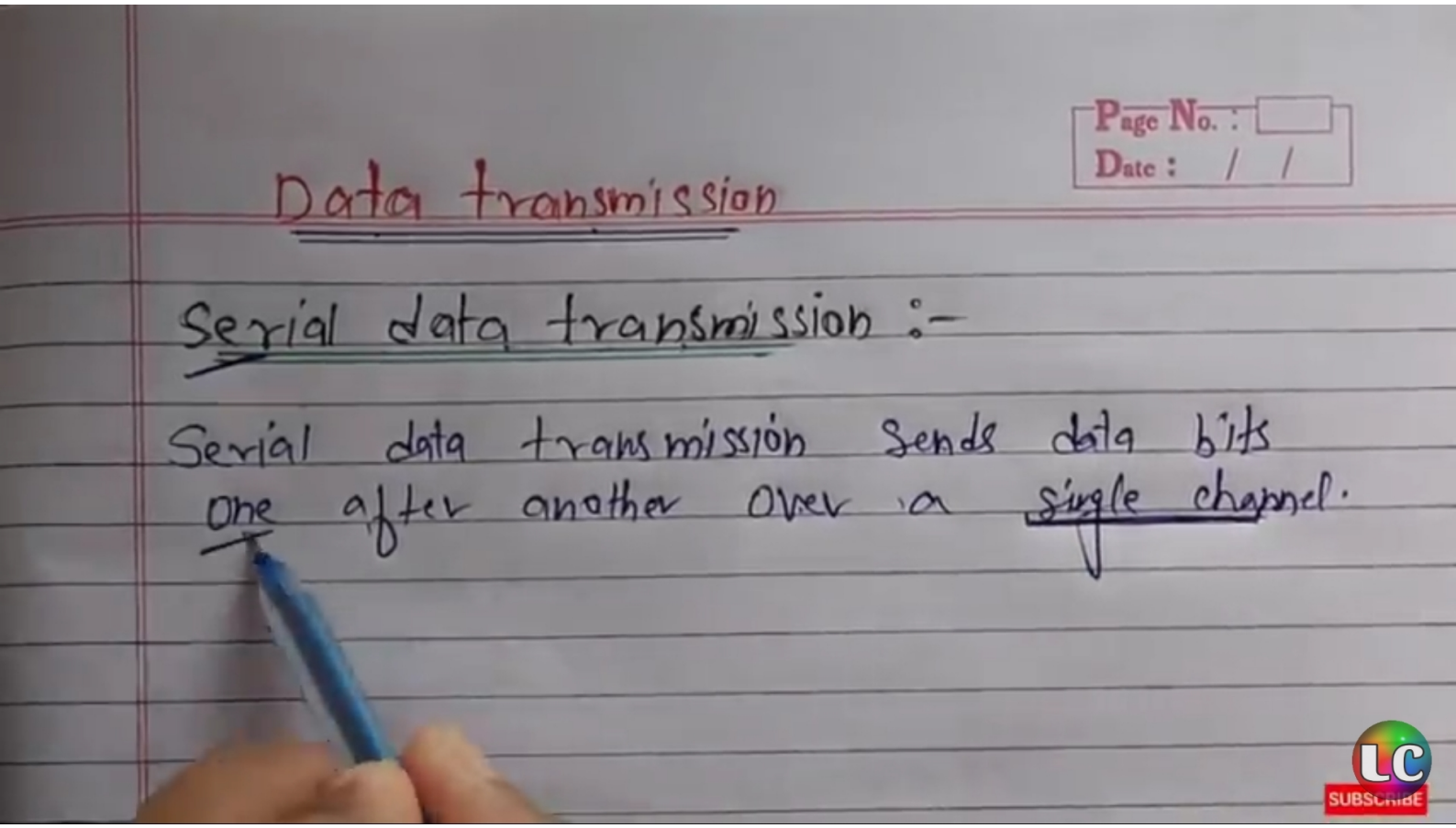
Parallel data transmission sends multiple data bits at the same time over "multiple channels."

Page No. :
Date : / /

Data transmission

Serial data transmission :-

Serial data transmission sends data bits one after another over a single channel.

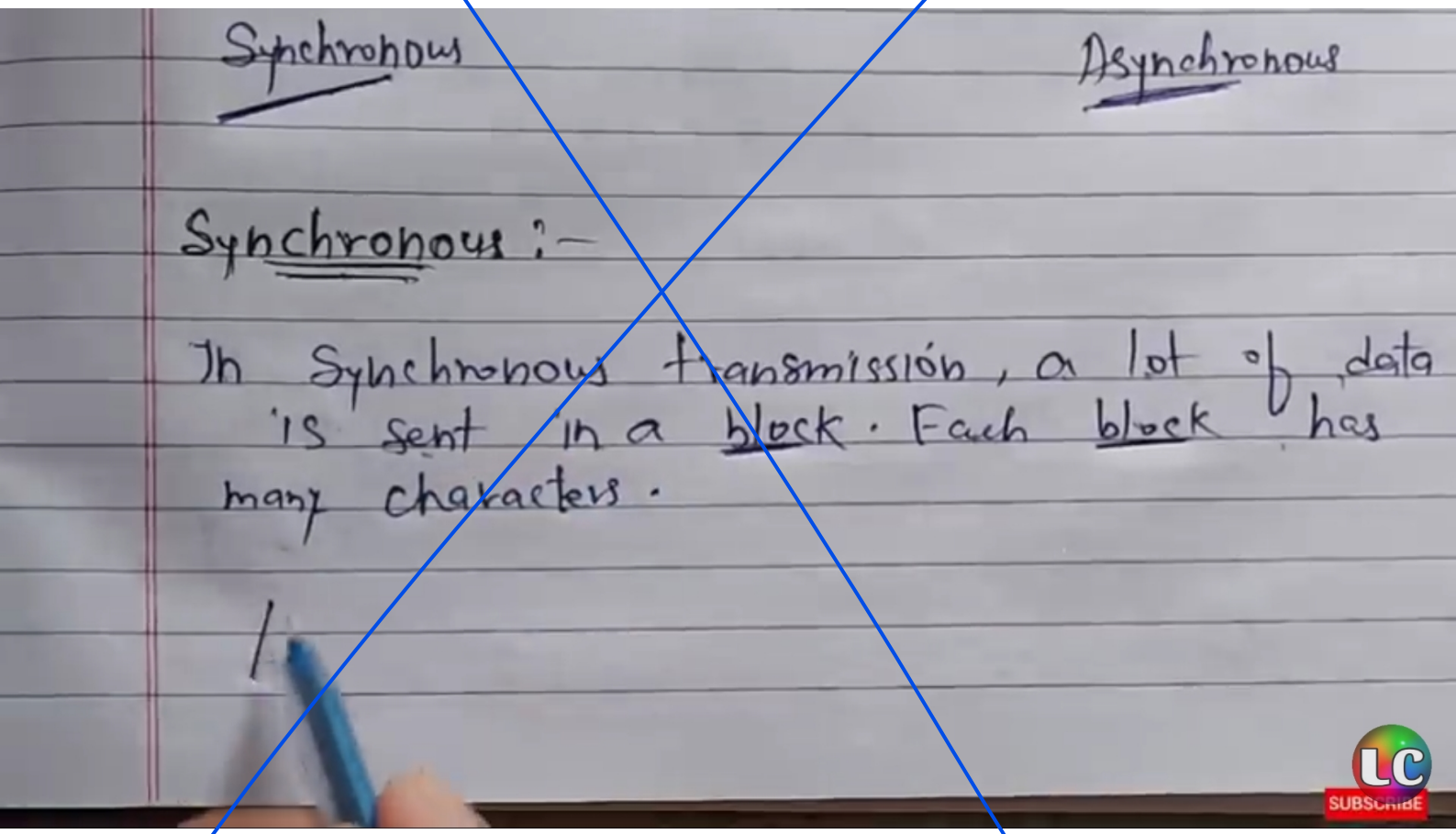


Synchronous

Asynchronous

Synchronous :-

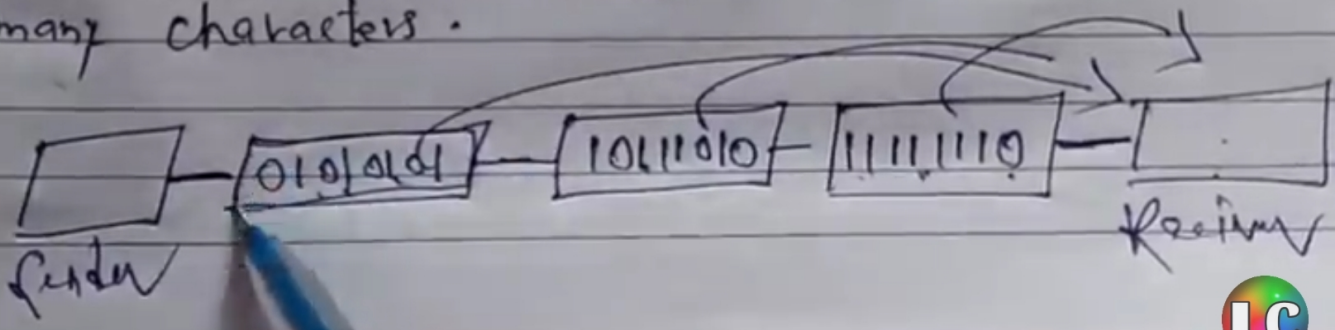
In Synchronous transmission, a lot of data is sent in a block. Each block has many characters.



Synchronous | Asynchronous

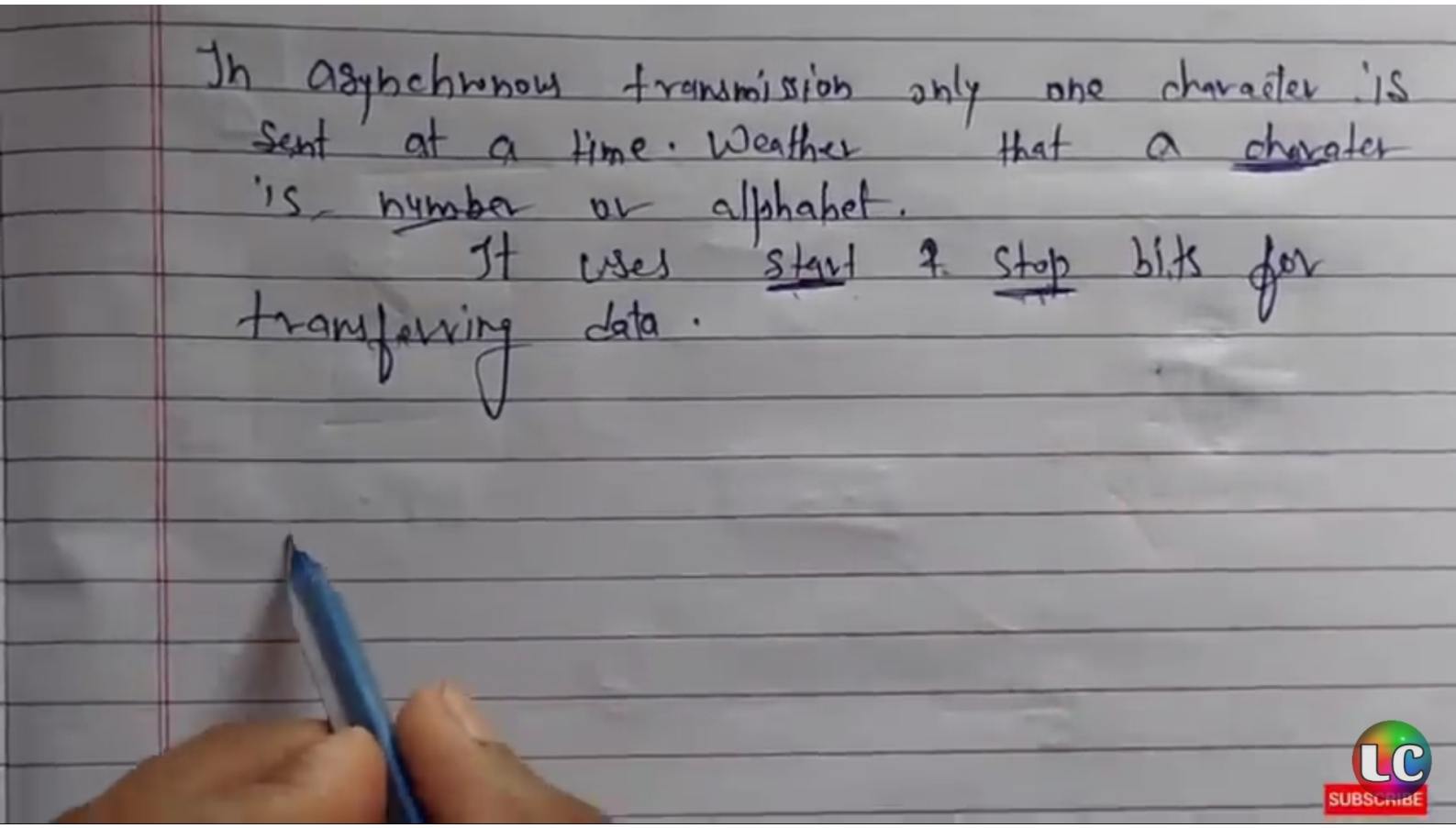
Synchronous :-

In Synchronous transmission, a lot of data is sent in a block. Each block has many characters.



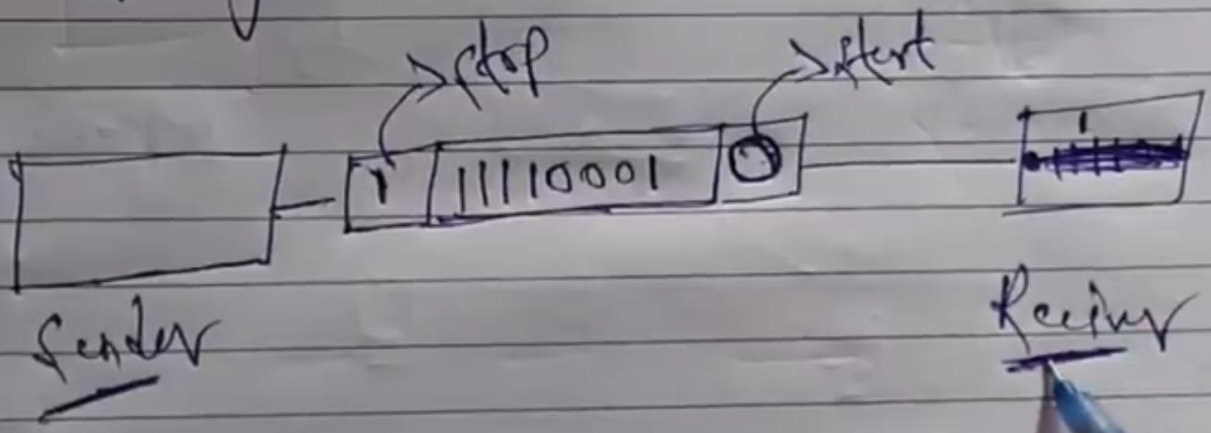
In asynchronous transmission only one character is sent at a time. Weather that a character is number or alphabet.

It uses start & stop bits for transferring data.

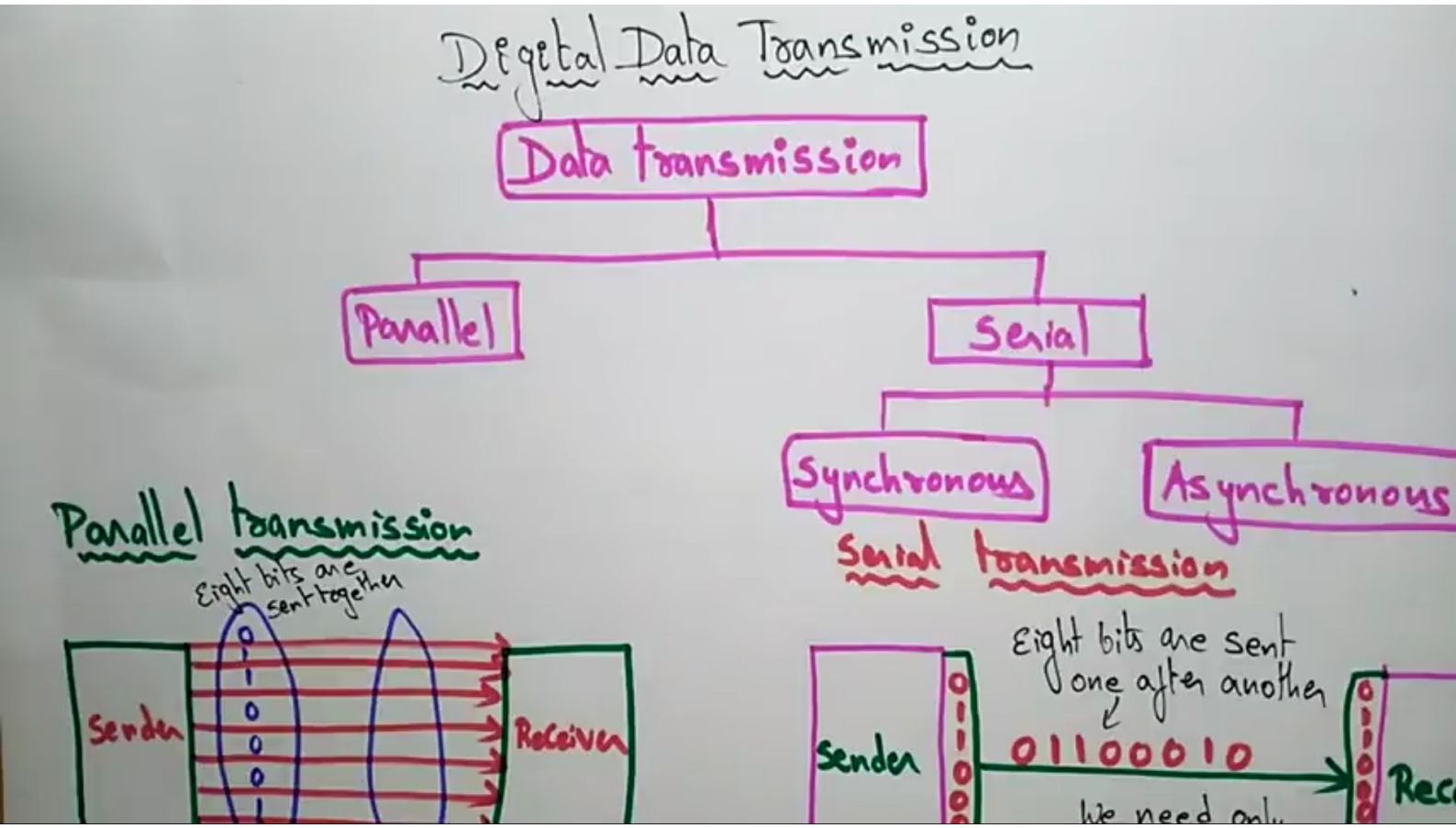


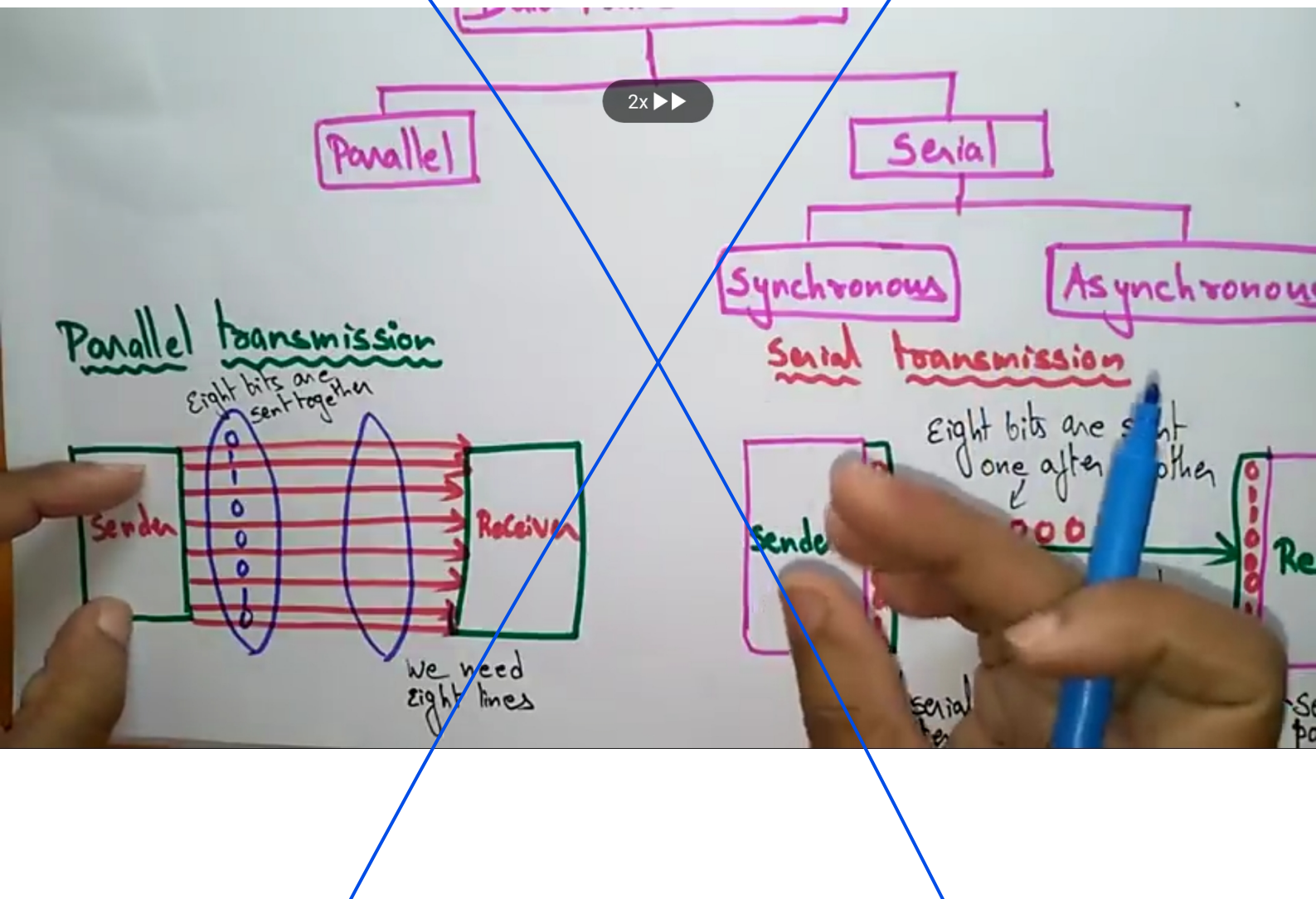
sent at a time. Weather that a character is number or alphabet.

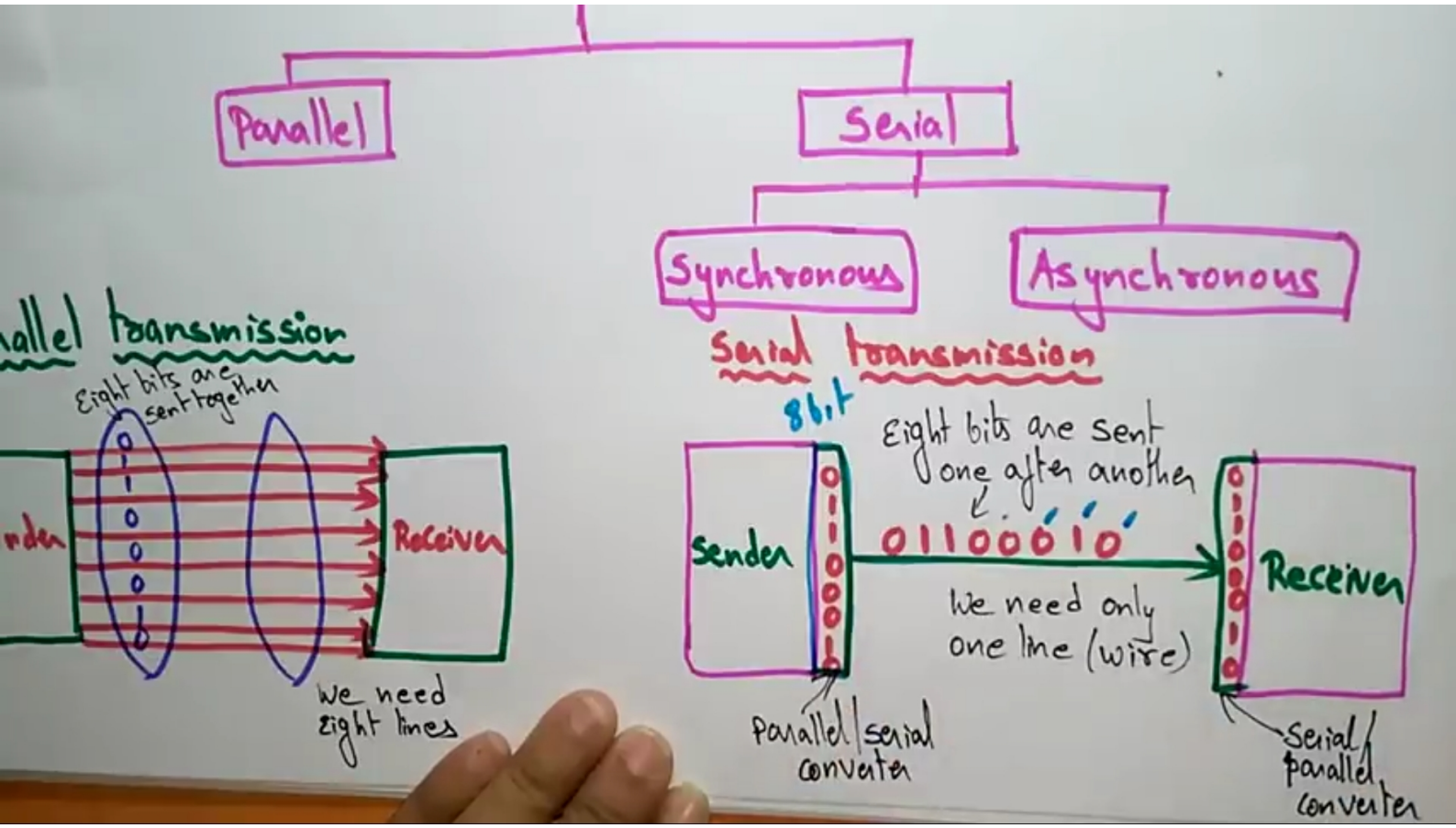
It uses start & stop bits for transferring data. 0 1



Digital Data Transmission



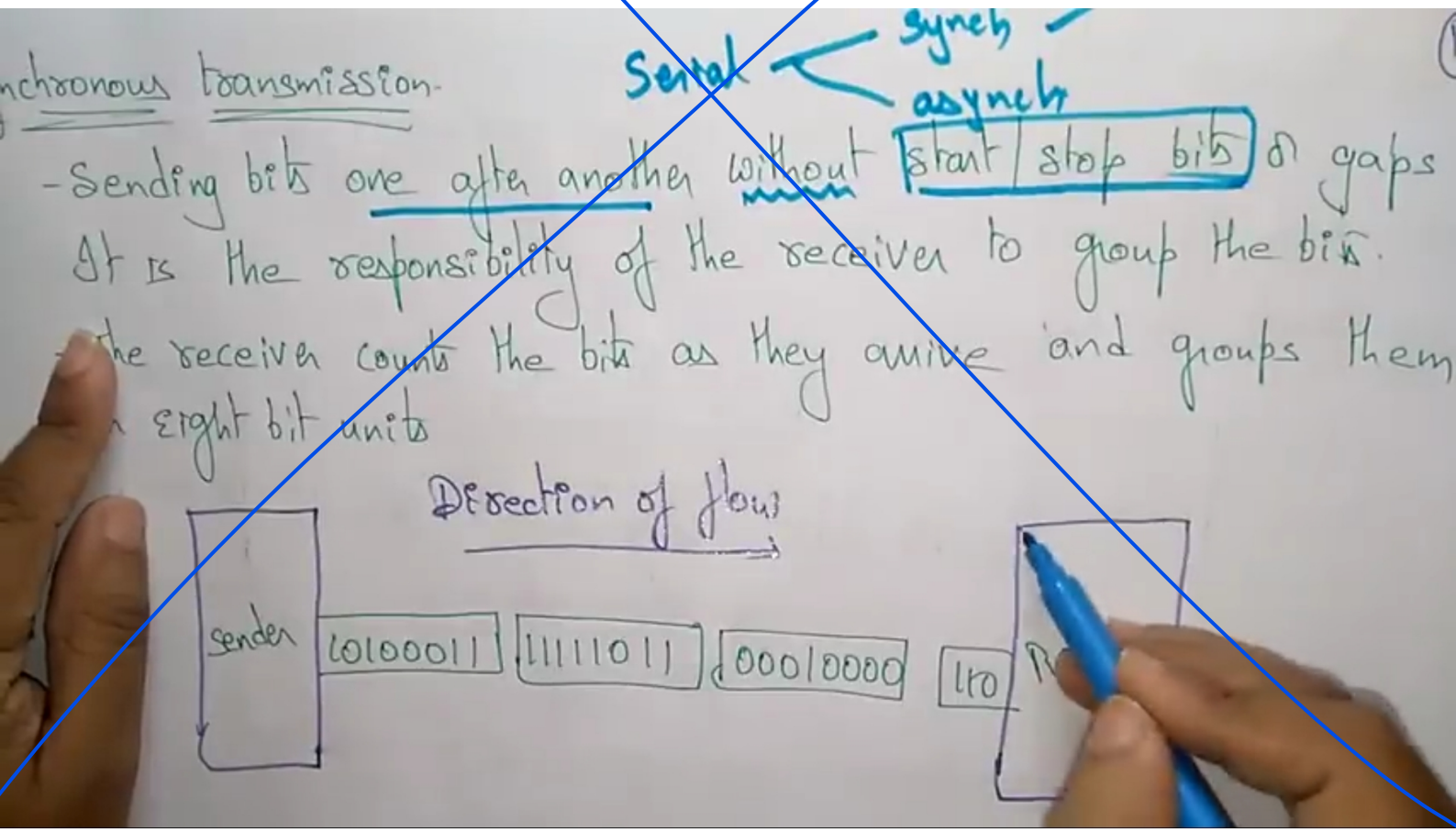

2x
Parallel
Serial
Synchronous
Asynchronous
Parallel transmission
Eight bits are sent together
Sender
Receiver
We need eight lines
Serial transmission
Eight bits are sent one after another
Sende
Re


Parallel
Serial
Synchronous
Asynchronous
allel transmission
Eight bits are sent together
Receiver
we need eight lines
Serial transmission
8bit
Eight bits are sent one after another
Sender
01100010
we need only one line (wire)
Receiver
parallel/serial converter
Serial/parallel converter

nchronous transmission.

Serial < synch / asynch

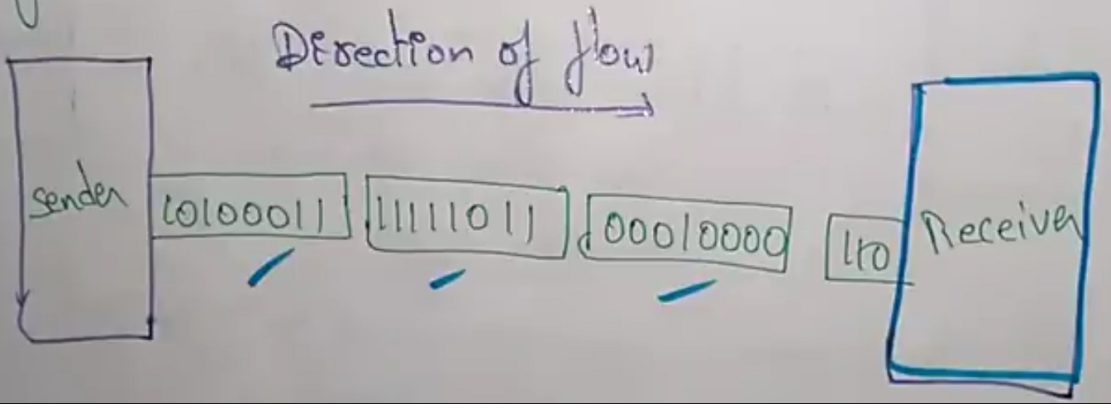
- Sending bits one after another without start/stop bits or gaps
It is the responsibility of the receiver to group the bits.
The receiver counts the bits as they arrive and groups them
eight bit units



chronous transmission.

Serial < synch / asynch

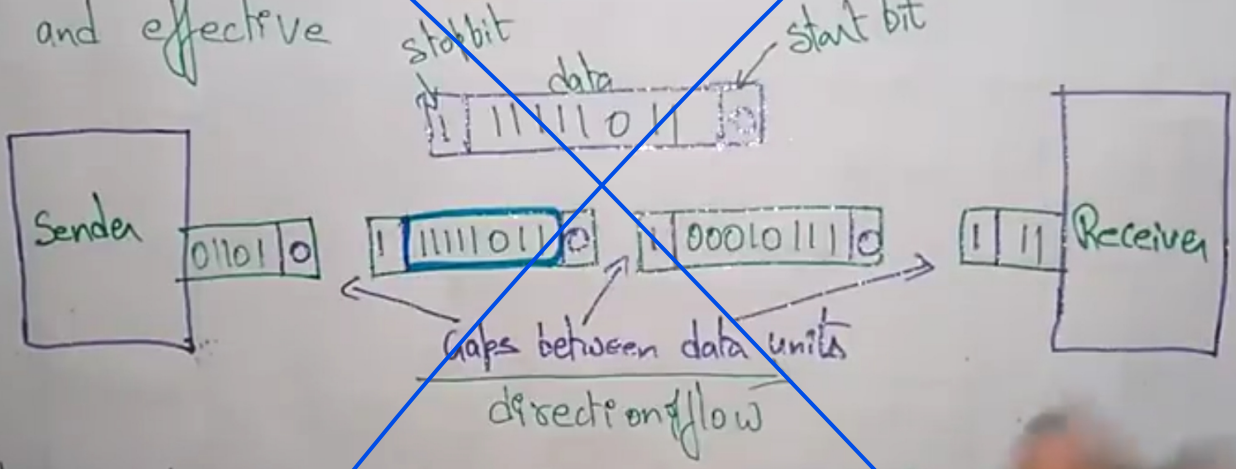
- Sending bits one after another without start/stop bits or gaps.
  It is the responsibility of the receiver to group the bits.
- The receiver counts the bits as they arrive and groups them in eight bit units



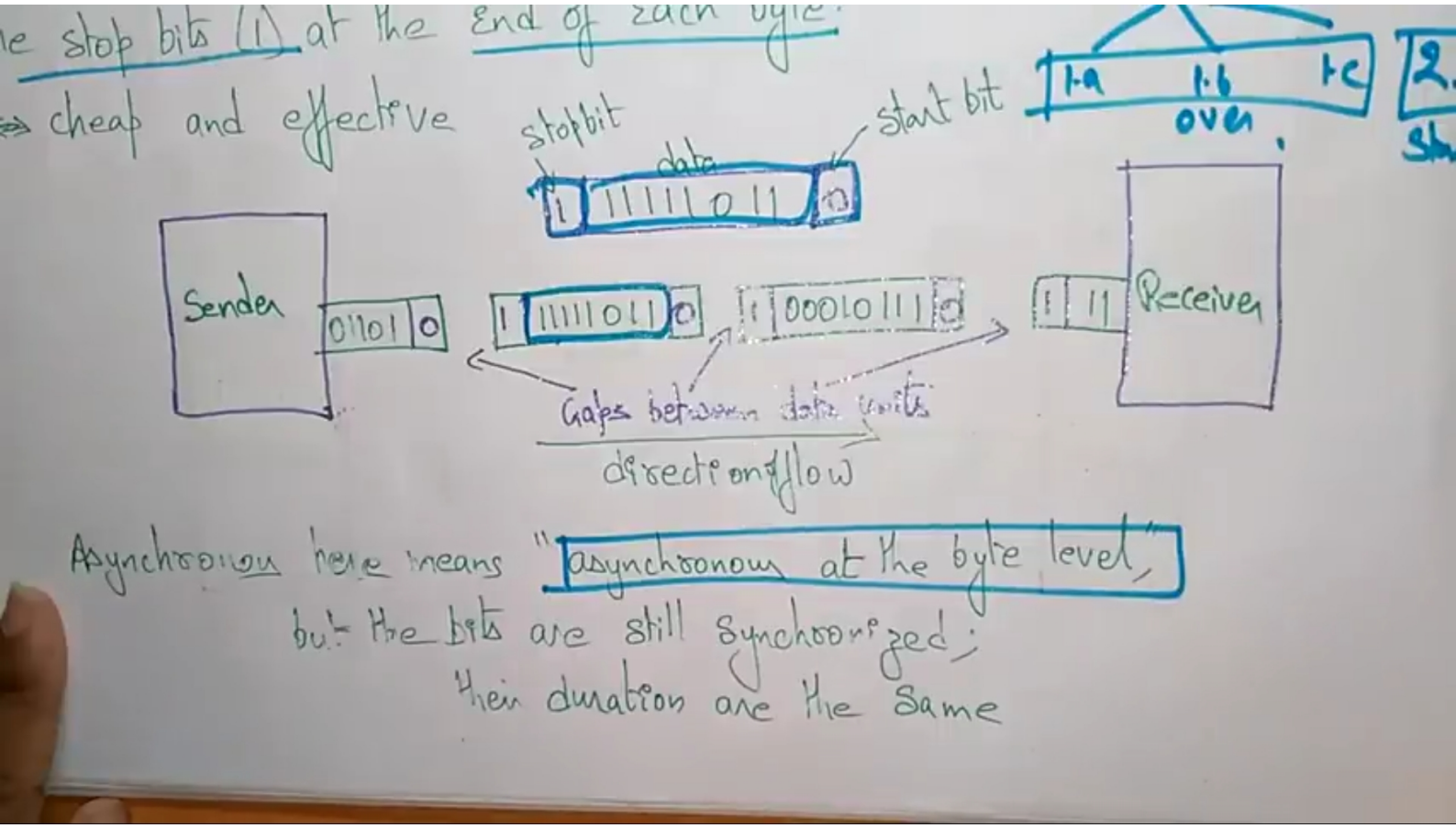
Asynchronous Transmission

Send one start bit (0) at the beginning and one or more stop bits (1) at the end of each byte.

⇔ cheap and effective



Asynchronous here means "asynchronous at the byte le

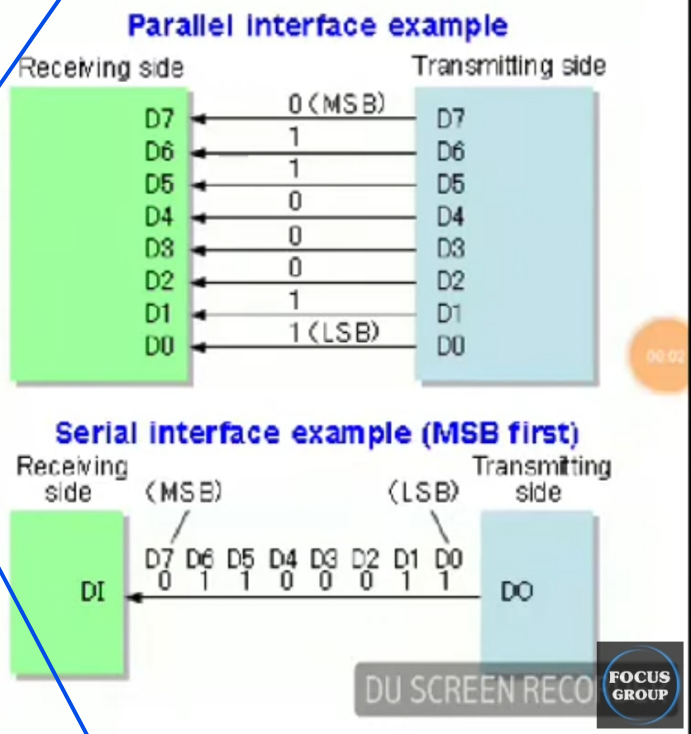


Asynchronous here means "asynchronous at the byte level"
but the bits are still synchronized;
their duration are the same

Parallel Transmission
&
Serial Transmission

Best Lecture in
Urdu/Hindi

SUBSCRIBE
Focus Group

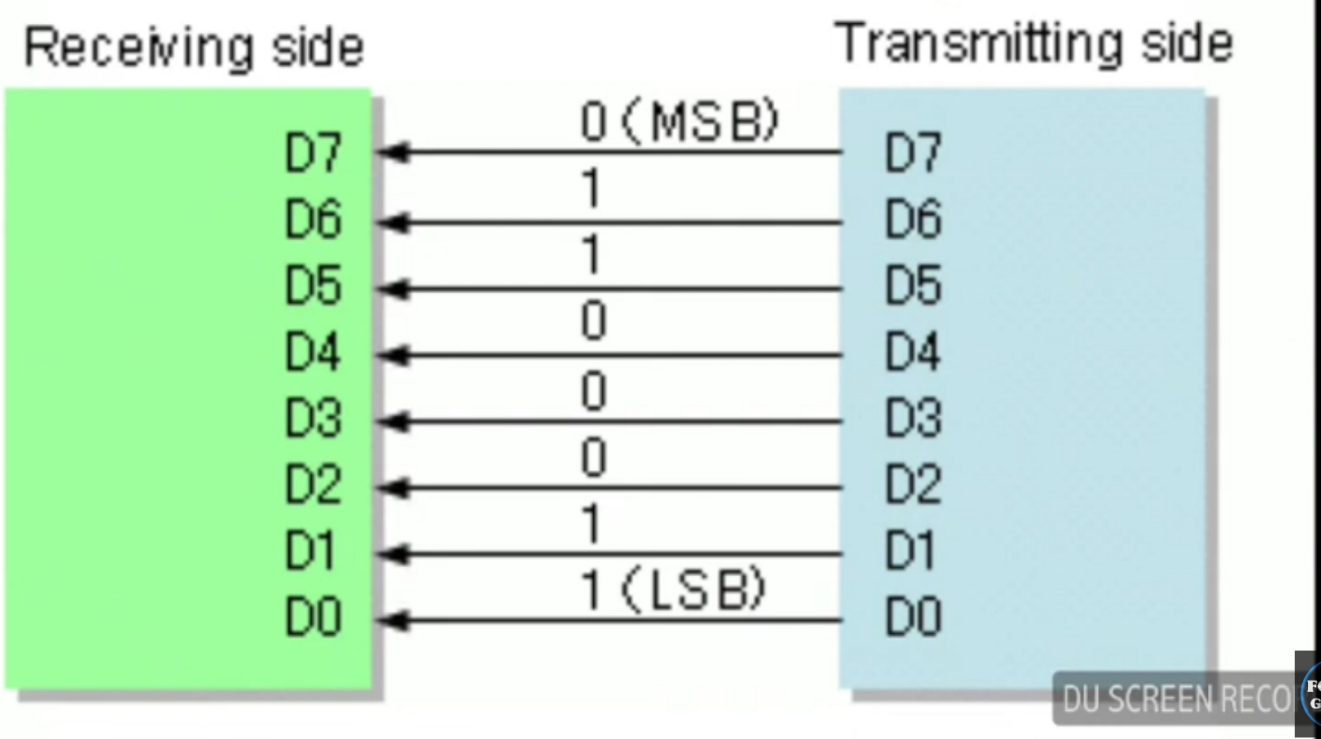


- **Parallel Transmission:**

A method of transmission in which group of bits are sent at the same time over multiple wires is called Parallel Transmission.

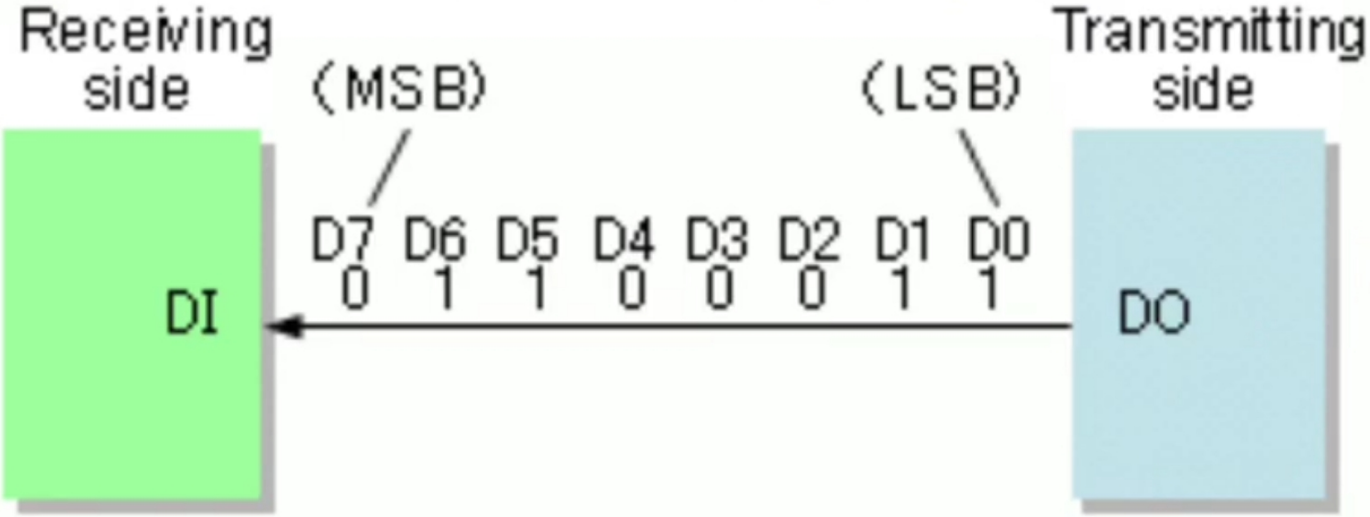
- **Each bit is transmitted over separate line.**
- **The data transmission between computer and printer is done using parallel Transmission.**
- **Parallel Transmission is faster because all bits are sent at the same time.**

Parallel interface example



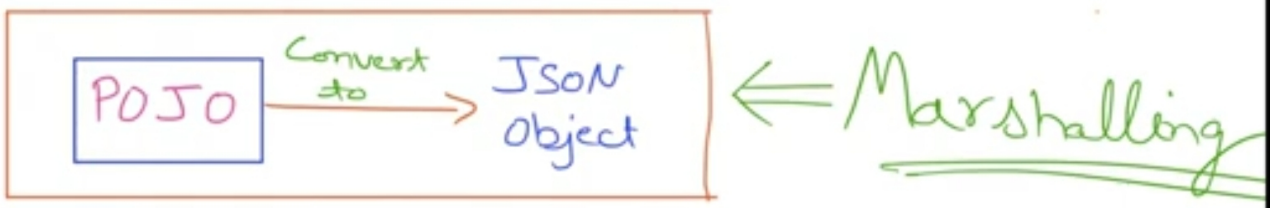
- **Serial Transmission:**
  A method of transmission in which data is sent one bit at a time is called serail transmission.
  - **Serial Transmission is Slowr than parallel Transmission as data is sent sequentially one bit at a time.**
  - **Telephone line use this method of Data Transmission.**

Serial interface example (MSB first)



Question23. What is Marshalling?

The Process in which POJO convert in to the JSON Object is called Marshalling.



Marshalling done with the help of some API like Google JSON, Simple JSON, JSON SOUP etc.

Question24. What is Unmarshalling?

The Process in which JSON OBJECT convert in to the POJO is called Unmarshalling.

Unmarshalling done with the help of some API like Google JSON, Simple JSON, JSON SOUP etc.

4.3. External data representation and marshalling

- At language-level data are stored in data structures
- At TCP/UDP-level data are communicated as 'messages' or streams of bytes – hence, conversion/flattening is needed
  - Converted to a sequence of bytes
- Problem? Different machines have different primitive data reps,
  - Integers: big-endian and little-endian order
  - float-type: representation differs between architectures
  - char codes: ASCII, Unicode
- Either both machines agree on a format type (included in parameter list) or an *intermediate* external standard is used:
  - External data representation: an agreed standard for the representation of data structures and primitive values
  - e.g., CORBA Common Data Rep (CDR) for many languages; Java object serialization for Java code only

4.3. External data representation and marshalling

- Marshalling: process of taking a collection of data items and assembling them into a form suitable for transmission
- Unmarshalling: disassembling (restoring) to original on arrival
- Three alter. approaches to external data representation and marshelling:
  - CORBA's common data representation (CDR)
  - Java's object serialization
  - XML (Extensible Markup Language) : defines a textual format for rep. structured data
- First two: marshalling & unmarshalling carried out by middleware layer
  - XML: software available
- First two: primitive data types are marshalled into a binary form
  - XML: represented texually
- Whether the marshalled data include info concerning type of its contents?
  - CDR: no, just the values of the objects transmitted
  - Java: yes, type info in the serialized form
  - XML: yes, type info refer to externally defined sets of names (with types), *namespaces*

4.3. External data representation and marshalling

- CORBA CDR
  - 15 primitive types: short, long, unsigned short, unsigned long, float, double, char, boolean, octet, any
  - Constructed types: sequence, string, array, struct, enum and union
    - note that it does not deal with objects (*only Java does: and tree of objects*)

Type	*Representation*
sequence	length (unsigned long) followed by elements in order
string	length (unsigned long) followed by characters in order (can also can have wide characters)
array	array elements in order (no length specified because it is fixed)
struct	in the order of declaration of the components
enumerated	unsigned long (the values are specified by the order declared)
union	type tag followed by the selected member

4.3. External data representation and marshalling

- Although we are interested in the use of external data representation for the arguments and results of RMIs and RPCs, it has a more general use for representing data structures, objects, or structured documents in a form suitable for transmission or storing in files

4.3. External data representation and marshalling

index in sequence of bytes	← 4 bytes →	*notes on representation*
0–3	5	*length of string*
4–7	"Smit"	*'Smith'*
8–11	"h "	
12–15	6	*length of string*
16–19	"Lond"	*'London'*
20-23	"on "	
24–27	1934	*unsigned long*

The flattened form represents a *Person*struct with value: {'Smith', 'London', 1934}

The process of transforming the memory representation of an object to a data format suitable for storage or transmission, which is typically used when data must be moved between different parts of a computer program, or from one program to another.

The arrangement of an escutcheon to exhibit the alliances of the owner.

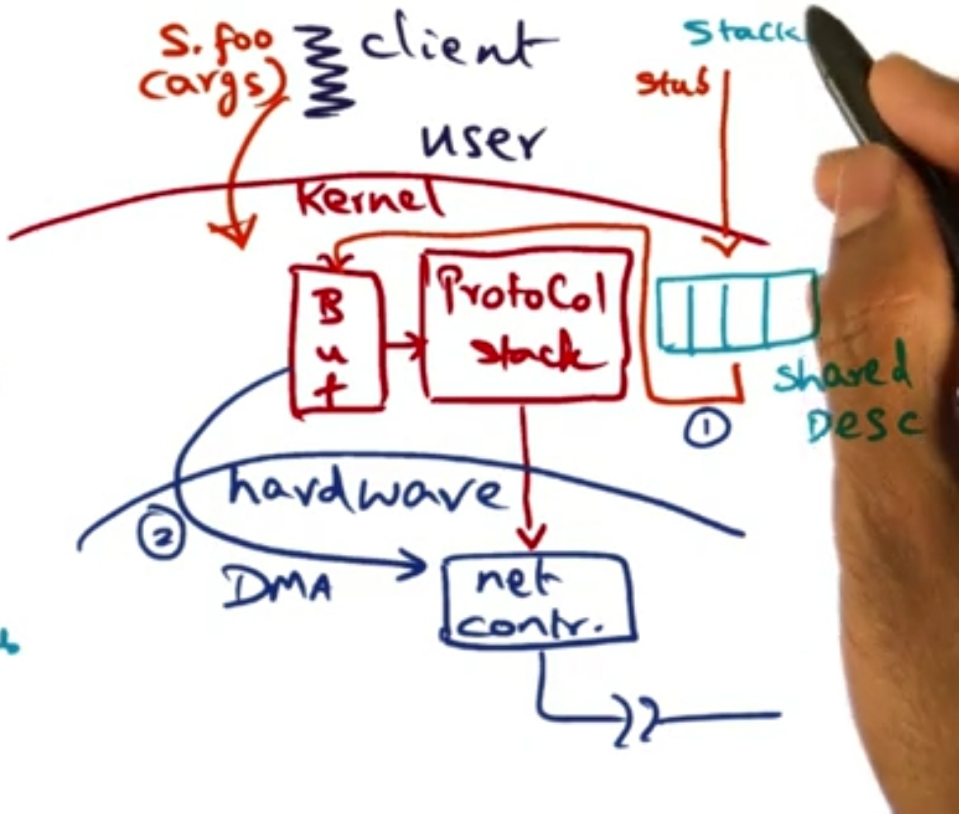
marshaling and data copying

Three copies
- client stub
- kernel buffer
- DMA to controller

Reducing copies?
- marshal into kernel buffer directly

OR

- shared descriptors between client stub and kernel



- Topic
  - External data representation
  - Marshalling
  - Multicast Communication

- **External data representation**

  - The decided data format for representation of data structure during the process of interprocess communication is External data representation.

- **Marshalling**

  - collecting groups of data & arranging them in a suitable data format for data transmission in the form of msg during interprocess communication is Marshalling

- **Substitutes External data representation & Marshalling**
  1. CDR
  2. Java's object serilization
  3. Extensible markup language

- **Substitutes External data representation & Marshalling**

  1. CORBA's CDR

  - Defined with CORBA 2.0
  - Middleware is responsible
  - Original data type are organized into binary form
  - All data types can be represented with the help of CDR in CORBA 2.0.

- **Substitutes External data representation & Marshalling**

  2. Java's object serilization

  - Middleware layer is responsible

  - Original data types are organized into binary form

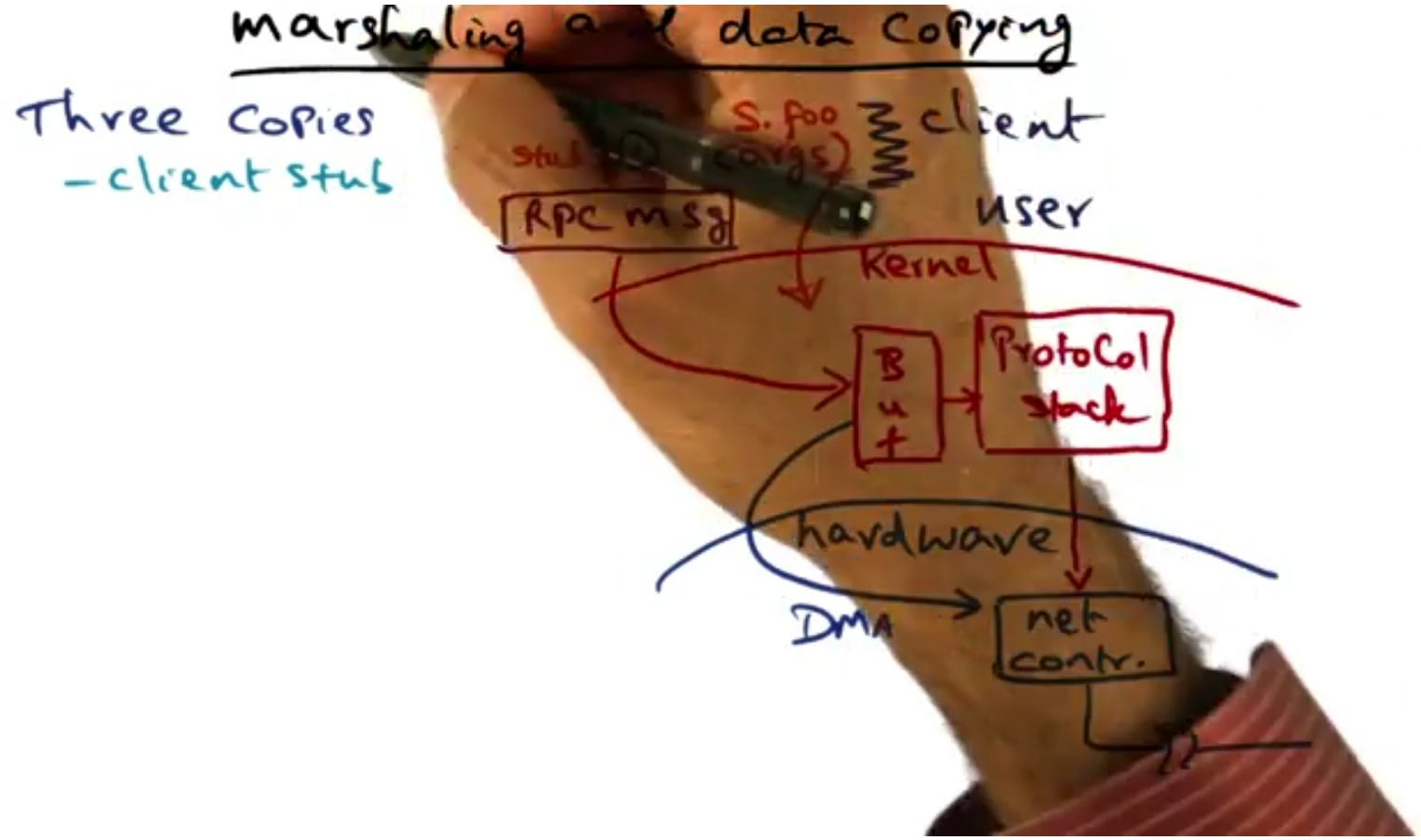
  - This type information is in serialized form

- **Substitutes External data representation & Marshalling**

  3. Extensible markup language

  - Defined by W3C
  - Primitives data types are textually represented
  - Namespaces

- **Multicast Communication**
  - Reliable multicasting
  - Characteristics to build DS
  1. fault tolerance
  2. Better performance
  3. broadcasting of event notification


marshaling & data copying
Three copies
- client stub
S. foo (args)
client
RPC msg
user
Kernel
Buf
ProtoCol stack
hardware
DMA
net contr.

Agenda

- Interprocess Communication
- IPC – Unicast and Multicast
- Interprocess Communication in Distributed Computing
- Interprocess Communication in basic HTTP
- Event Synchronization
- Synchronous vs. Asynchronous Communication
- Blocking, deadlock, and timeouts
- Using threads for asynchronous IPC
- Deadlocks and Timeouts
- Indefinite blocking due to a deadlock
- Data Representation

Interprocess Communications

- **Exchange of data** between two or more separate, independent processes/threads.
- Operating systems provide facilities/resources for inter-process communications (**IPC**), such as message queues, semaphores, and shared memory.
- Distributed computing systems make use of these facilities/resources to provide **application programming interface (API)** which allows IPC to be programmed at a higher level of abstraction. (e.g., send and receive)
- Distributed computing requires information to be exchanged among independent processes.

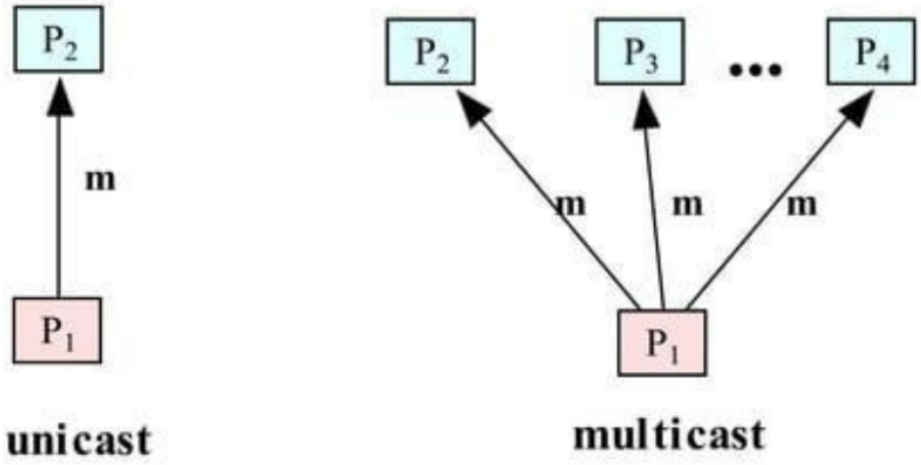
IPC – unicast and multicast

- In distributed computing, two or more processes engage in **IPC** using a **protocol** agreed upon by the **processes**. A process may be a sender at some points during a protocol, a receiver at other points.
- When communication is from one process to a single other process, the IPC is

IPC – unicast and multicast

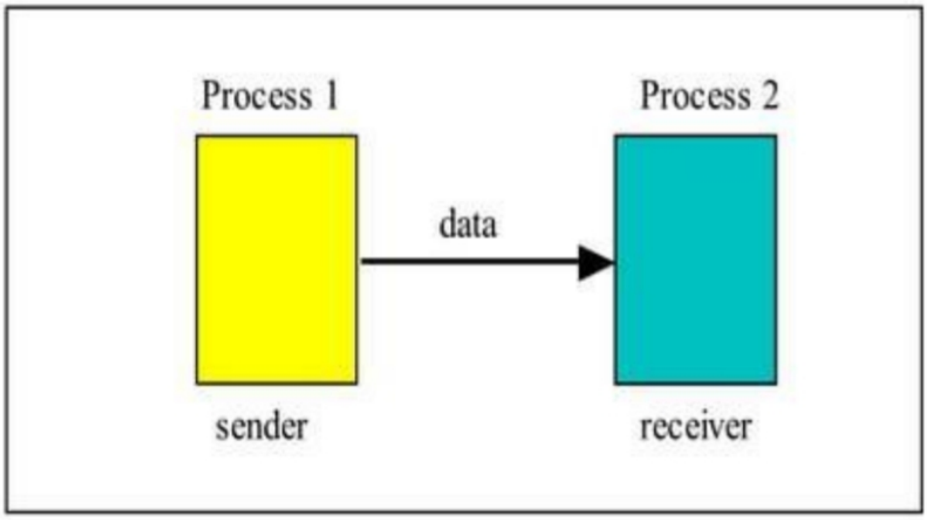
- In distributed computing, two or more processes engage in IPC using a protocol agreed upon by the processes. A process may be a sender at some points during a protocol, a receiver at other points.
- When communication is from one process to a single other process, the IPC is said to be a *unicast*, e.g., Socket communication. When communication is from one process to a group of processes, the IPC is said to be a *multicast*, e.g., Publish/Subscribe Message model.

Unicast vs. Multicast



Interprocess Communications in Distributed Computing

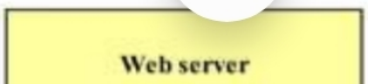
Interprocess Communications in Distributed Computing



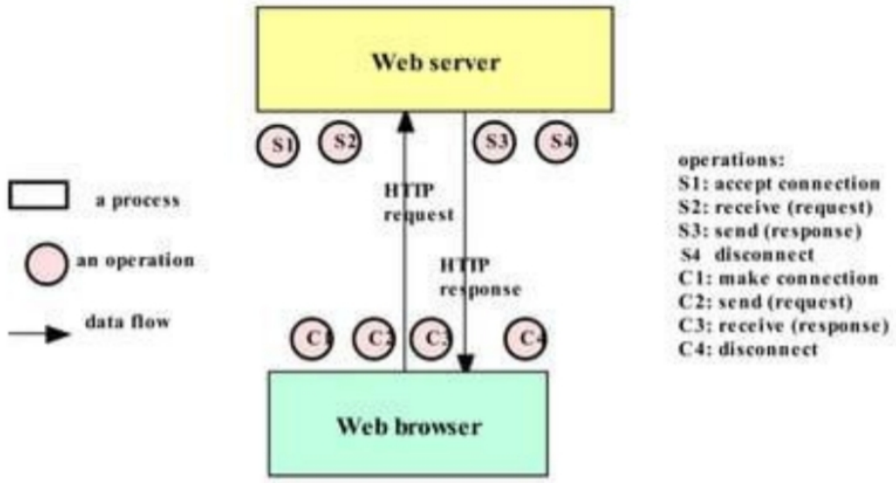
Operations provided in an archetypal Interprocess Communications API

- **Receive** ([sender], message storage object)
- **Connect** (sender address, receiver address), for connection-oriented communication.
- **Send** ([receiver], message)
- **Disconnect** (connection identifier), for connection-oriented communication.

erprocess Communication in basic HTTP



Interprocess Communication in basic HTTP



Processing order: C1, S1, C2, S2, S3, C3, C4, S4

Event Synchronization

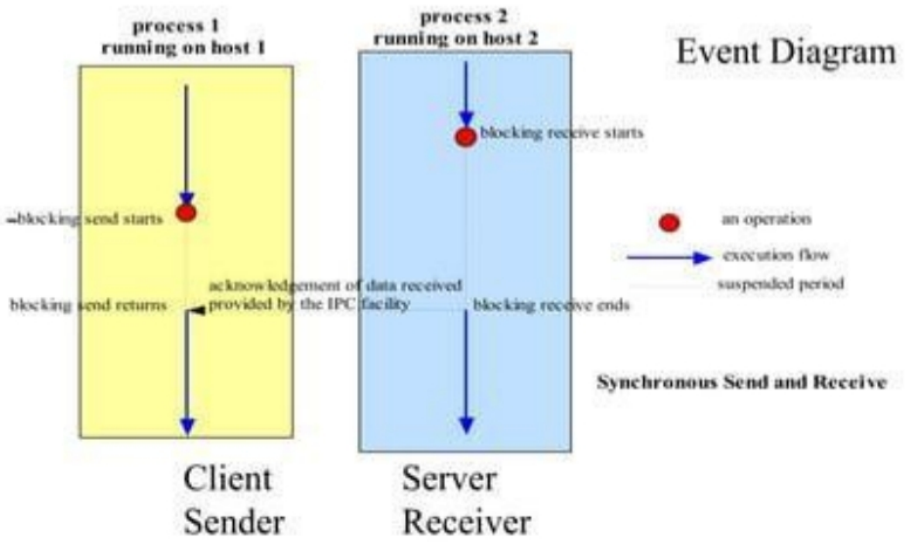
- Interprocess communication may require that the two processes synchronize their operations: one side sends, then the other receives until all data has been sent and received.
- Ideally, the send operation starts before the receive operation commences.
- In practice, the synchronization requires system support.

Synchronous vs. Asynchronous Communication

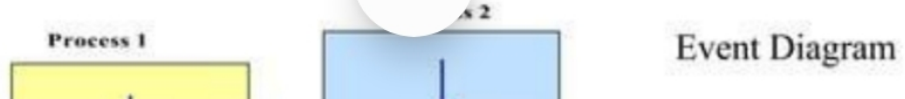
Synchronous vs. Asynchronous Communication

- The IPC operations may provide the synchronization necessary using *blocking*. A blocking operation issued by a process will block further processing of the process until the operation is fulfilled.
- Alternatively, IPC operations may be asynchronous or nonblocking. An asynchronous operation issued by a process will not block further processing of the process. Instead, the process is free to proceed with its processing, and may optionally be notified by the system when the operation is fulfilled.

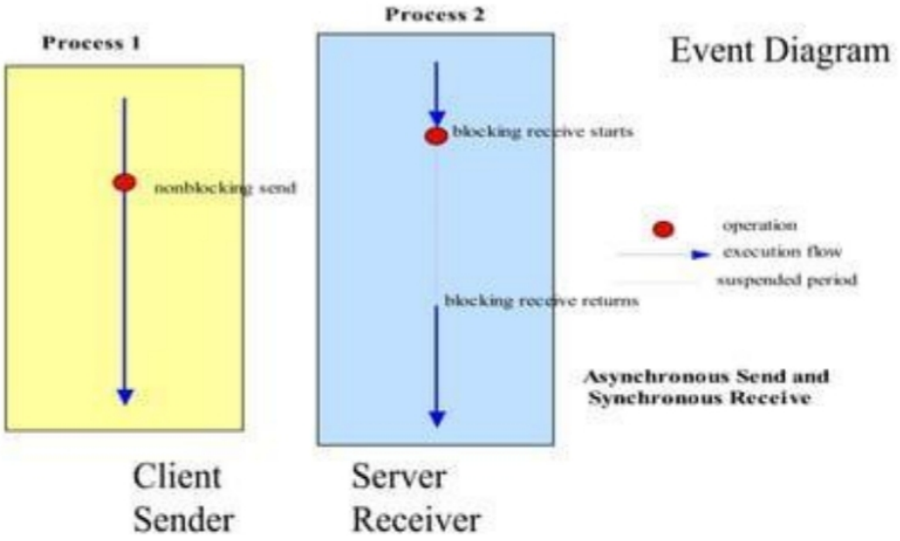
Synchronous send and receive



Asynchronous send and synchronous receive

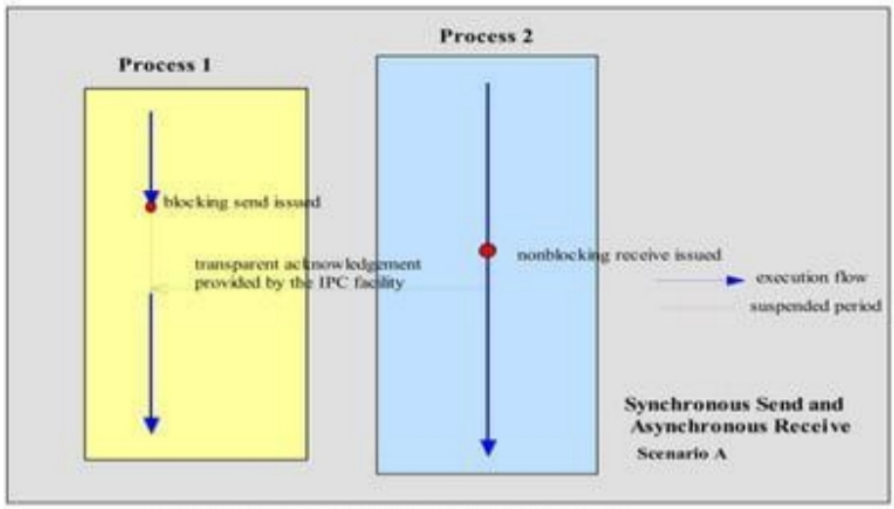


Asynchronous send and synchronous receive



12/

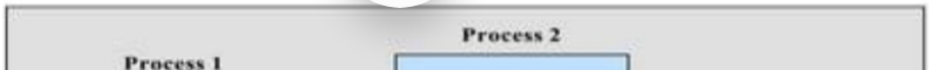
Synchronous send and Async. Receive - 1



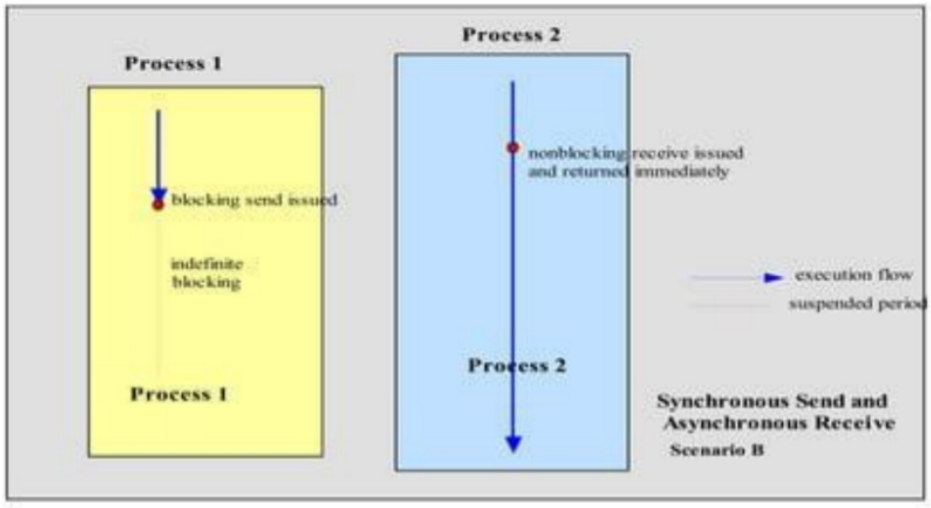
Data from P1 was received by P2
before issuing a non-blocking receive op in P2

13/

Synchronous send and Asyn[illegible]eive - 2



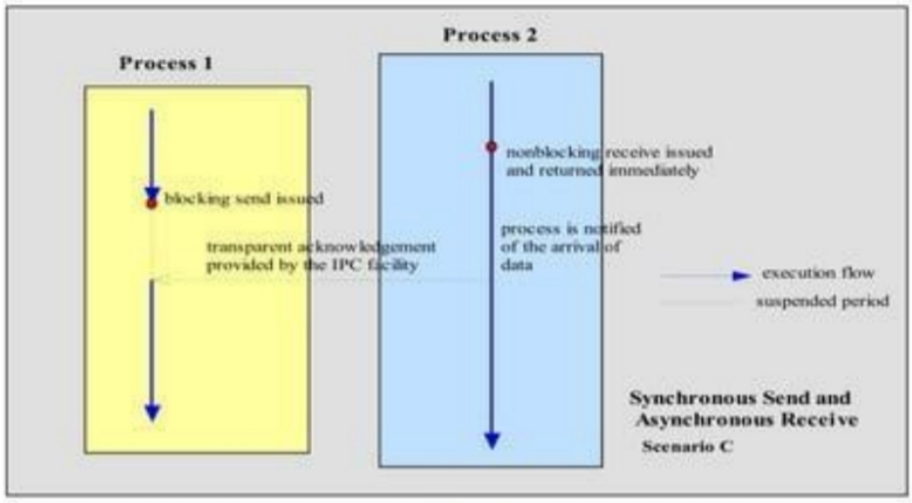
Synchronous send and Async. Receive - 2



Data from P1 arrived to P2 **after** P2 issued a non-blocking receive op

14/2

Synchronous send and Async. Receive - 3

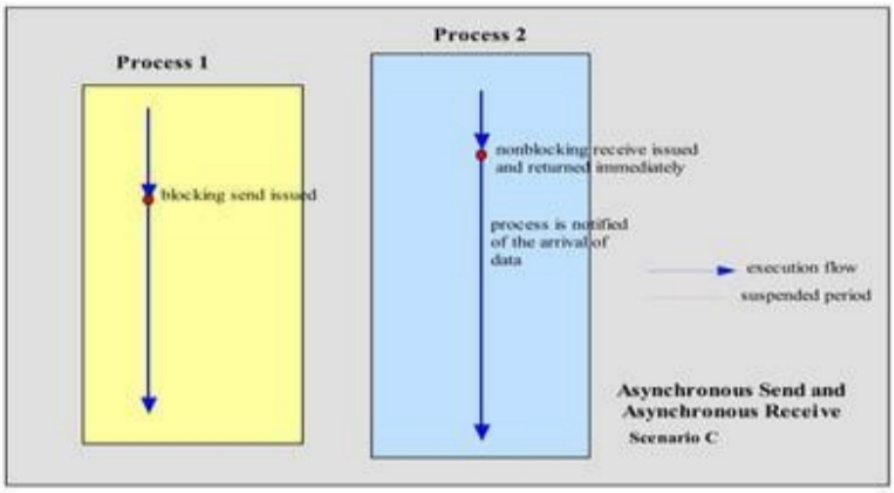


Data from P1 arrived to P2 **before** P2 issues a non-blocking receive op. P2 is notified of the arrival of data

15/2

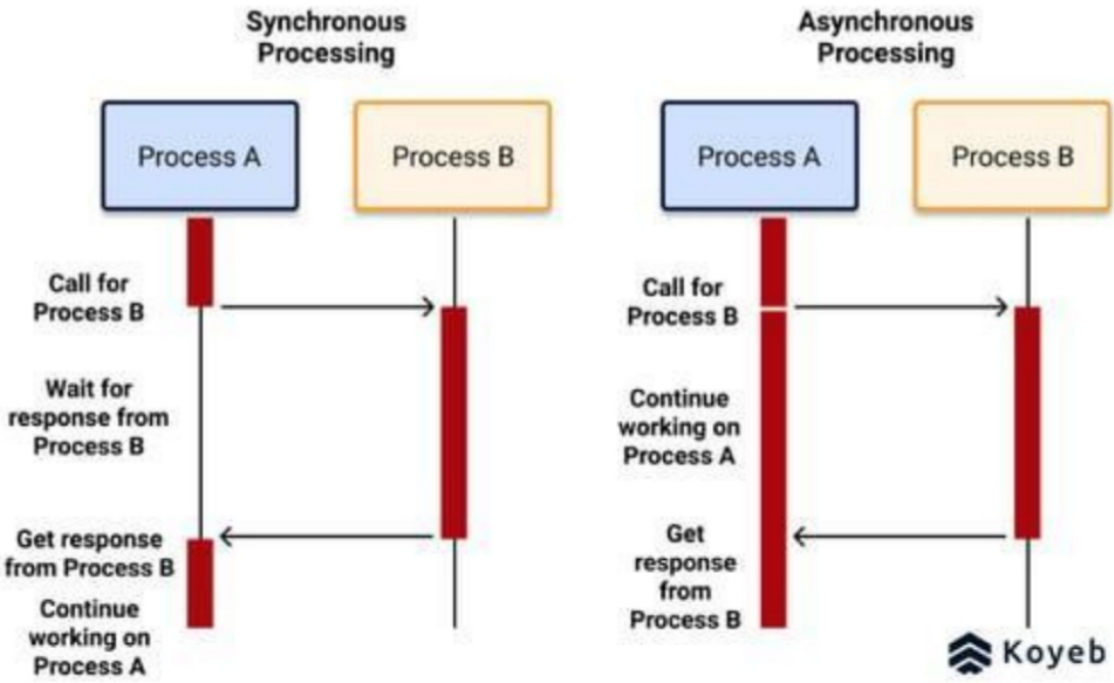
Asynchronous s and Asynchronous receive

Asynchronous send and Asynchronous receive



Does P1 need an acknowledgement from P2?

16/28

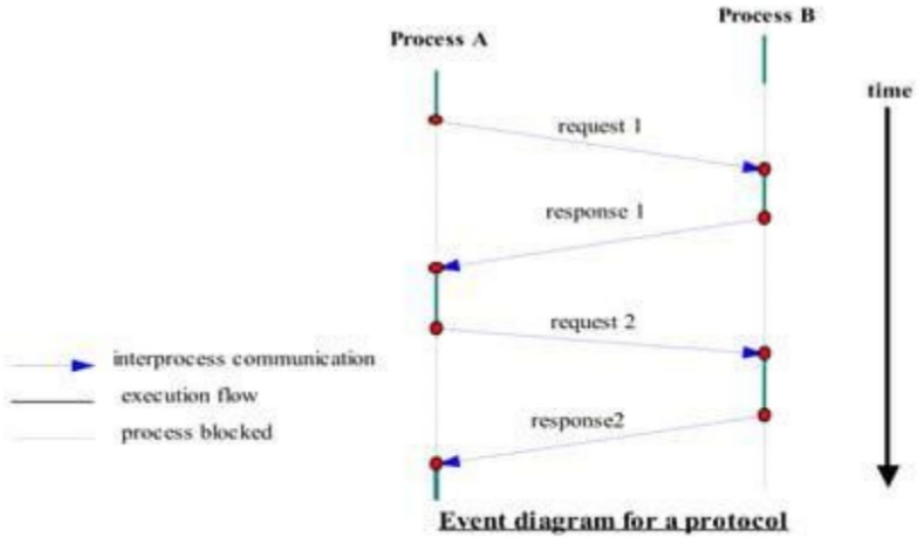


17/28

Event diagram

17

Event diagram

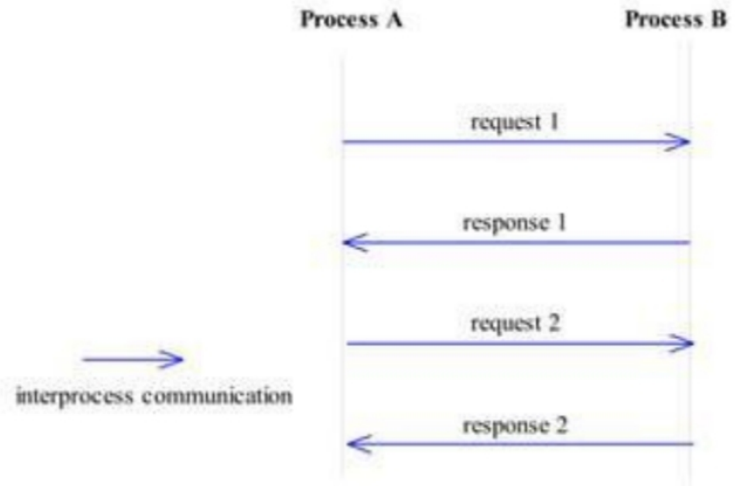


Event diagram for a protocol

Synchronous send and receive

18

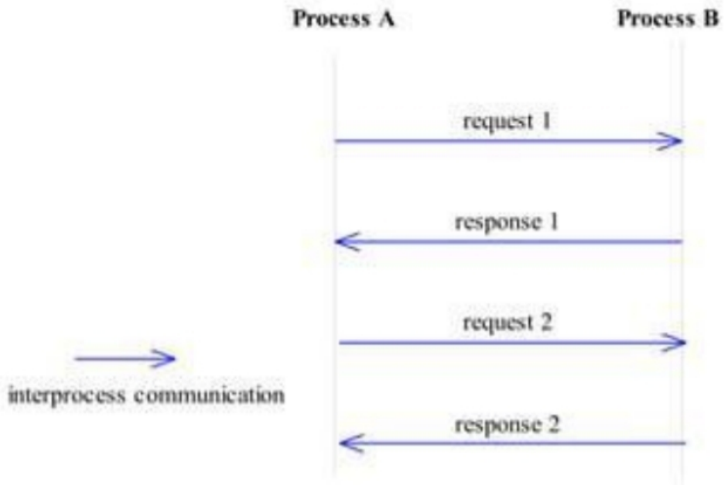
Sequence Diagram



19

Sequence diagram for a HTTP session

Sequence diagram for a HTTP session



Blocking, deadlock, and timeouts

- **Blocking operations issued in the wrong sequence can cause *deadlocks*.**
- **Deadlocks should be avoided. Alternatively, *timeout* can be used to detect deadlocks.**

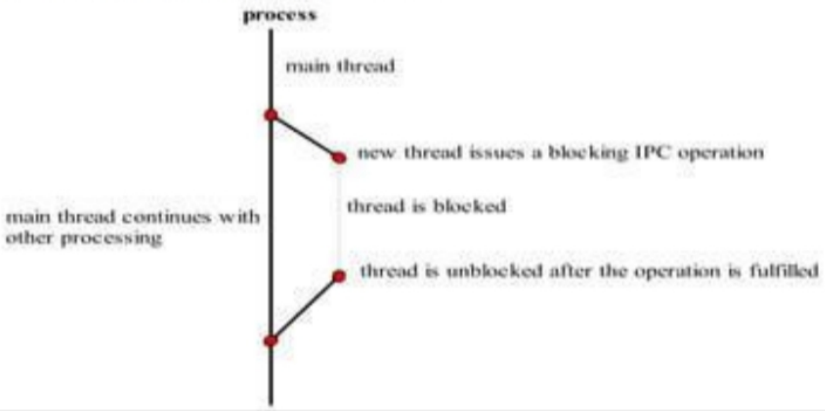
P1 is waiting for P2's data; P2 is waiting for P1's data.

Using threads fc chronous IPC

- **When using an IPC programming interface, it is important to note whether**

Using threads for asynchronous IPC

- **When using an IPC programming interface, it is important to note whether the operations are synchronous or asynchronous.**
- **If only blocking operation is provided for send and/or receive, then it is the programmer's responsibility to using child processes or threads if asynchronous operations are desired.**



2

Deadlocks and Timeouts

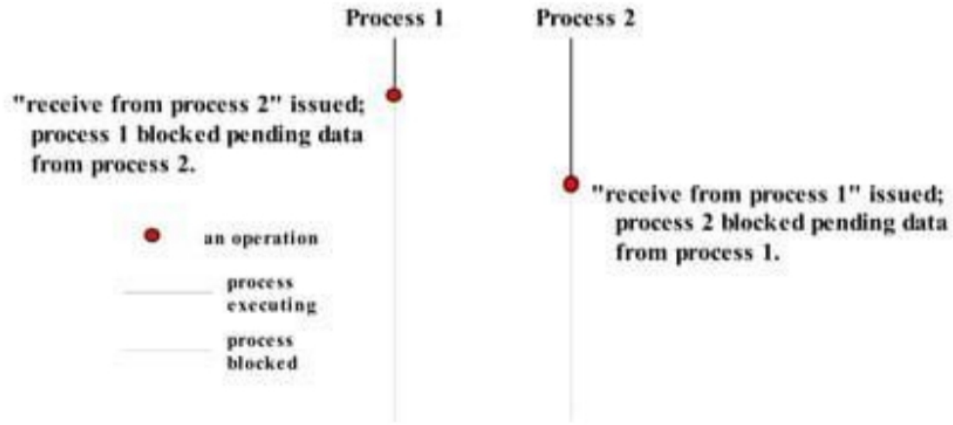
- Connect and receive operations can result in indefinite blocking
- For example, a blocking *connect request* can result in the requesting process to suspended indefinitely if the connection is unfulfilled or cannot be fulfilled, perhaps as a result of a breakdown in the network .
- It is generally unacceptable for a requesting process to "hang" indefinitely. Indefinite blocking can be avoided by using timeout.
- Indefinite blocking may also be caused by a *deadlock*

2

Indefinite blocking to a deadlock

2

Indefinite blocking due to a deadlock



P1 is waiting for P2's data; P2 is waiting for P1's data.

2

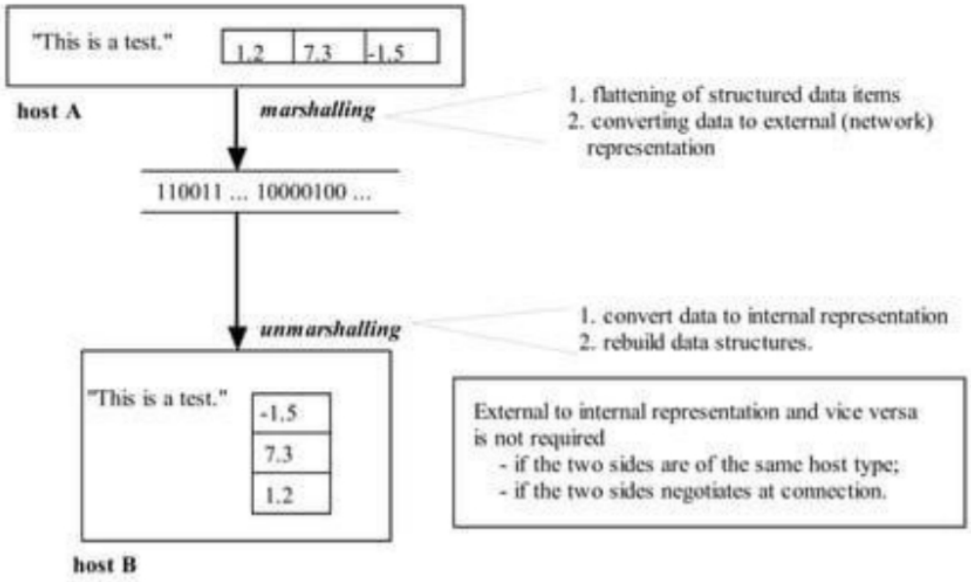
Data Representation

- Data transmitted on the network is a binary stream.
- An interprocess communication system may provide the capability to allow data representation to be imposed on the raw data.
- Because different computers may have different internal storage format for the same data type, an external representation of data may be necessary—standard format.
- *Data marshalling* is the process of (I) flattening a data structure, and (ii) converting the data to an external representation.
- Some well known external data representation schemes are:
  - Sun XDR (External Data Representation)
  - ASN.1 (Abstract Syntax Notation One)
  - XML (Extensible Markup Language)

2

Data Marshalling

Data Marshalling

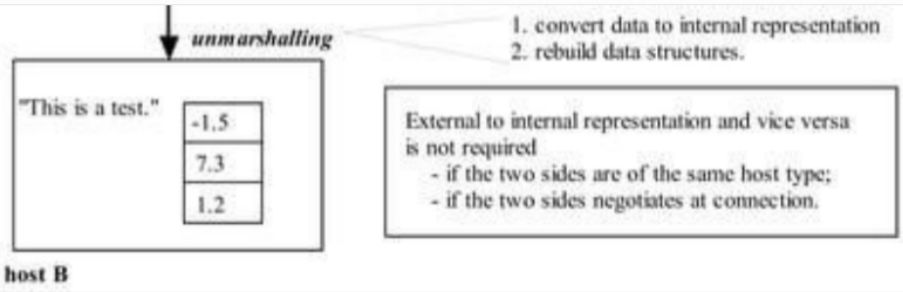


Text-based protocols

- Data marshalling is at its simplest when the data exchanged is a stream of characters, or text.
- Exchanging data in text has the additional advantage that the data can be easily parsed in a program and displayed for human perusal. Hence it is a popular practice for protocols to exchange requests and responses in the form of character-strings. Such protocols are said to be text-based.
- Many popular network protocols, including FTP (File Transfer Protocol), HTTP, and SMTP (Simple Mail Transfer Protocol), are text-based.

Protocol

- In a distributed application, two processes perform interprocess



26/

Text-based protocols

- Data marshalling is at its simplest when the data exchanged is a stream of characters, or text.
- Exchanging data in text has the additional advantage that the data can be easily parsed in a program and displayed for human perusal. Hence it is a popular practice for protocols to exchange requests and responses in the form of character-strings. Such protocols are said to be text-based.
- Many popular network protocols, including FTP (File Transfer Protocol), HTTP, and SMTP (Simple Mail Transfer Protocol), are text-based.

27/

Protocol

- In a distributed application, two processes perform interprocess communication in a mutually agreed upon **protocol.**
- The **specification** of a **protocol** should include

(i) the sequence of data exchange, which can be described using a time event diagram.

(ii) the format of the data exchange at each step.

28/