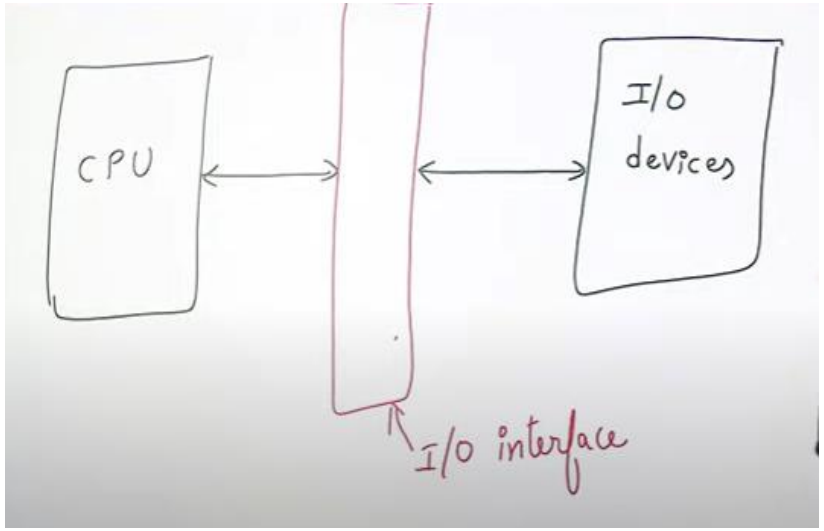
# Unit 5

# IO Organization




- Devices connect to processor externally (except main memory).
- This IO devices are also known as Peripheral devices.
- Generally, of 3 types
  - Input devices-keyboard, mic, joystick, scanner
  - Output devices-screen, printer, speaker
  - Storage devices- hard drive, pen drives.
- This IO devices are not directly accessed by CPU as nature of devices is different.
- Thus, an Interface known as I/O interface is required for accessing CPU and I/O. I/O interface establish communication between CPU and IO.
- CPU is purely electronic devices while IO devices is electromagnetic(hard-drive), electromechanical (printer). Thus, signal generated by each other is not understand by each other.
- Therefore, interface make the communication between CPU and IO.
- Since CPU and IO speed is mismatch ,interface provide the synchronization.
- Data Format conversion (enable different company keyboard and computer to work).
- Operating mode of peripheral devices are different. Thus, with interface they are operated such that no will get hampered.

- Interface is combination of s/w and h/w.
- S/w is generally program (drivers).
- E.g. DMA controller- DMA + interface.
- E.g. I/O processor- Interface + DMA+ IO instruction execution( Graphic processor- display instruction executor).

<u>IO vs memory buses:</u>

- Memory is not peripheral because it doesn't require any interface to communicate with CPU.
- There are 3 ways to connect memory and IO with CPU through buses.

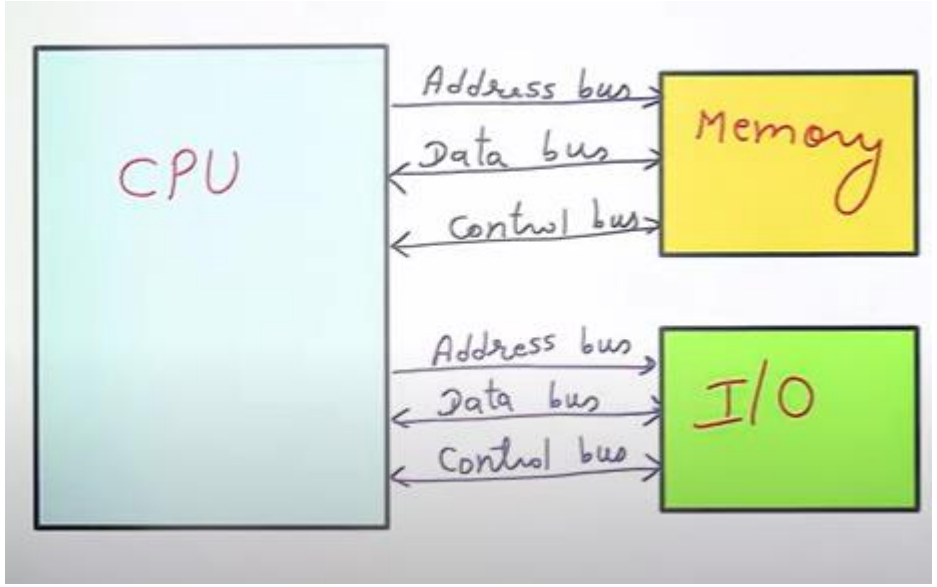
<u>1)Separate buses for both memory and IO.</u>

Disadvantage:

- More costly because of more buses

Advantage:

- Easy to implement.

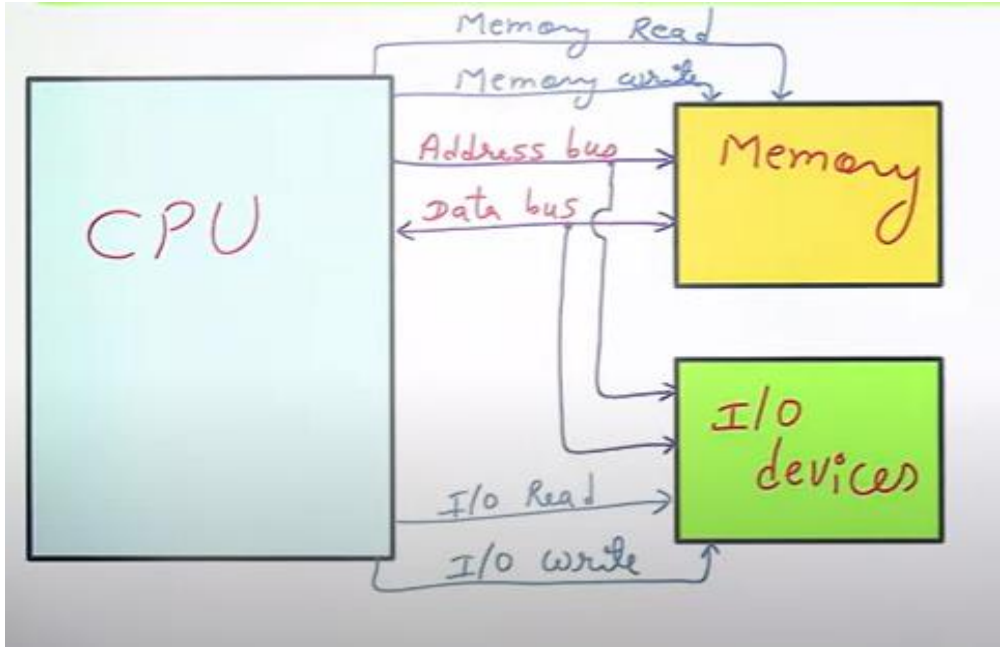


## 2) Common Data and Address Bus

- Same data and address bus but different control bus
- Thus, it is confusion which address is for which(IO or memory).
- Hence with different control signal we can differentiate address or data request.
- Control signal for memory- memory read, memory write.
- Control signal for IO- IO read. IO write.

e.g.- same name person with different job.

- CPU enable control signal along with address to access the respective devices.
- IO and memory have their own addresses.
- Also known as Isolated I/O or I/O-mapped- I/O as IO device have their own addresses.




Advantage:

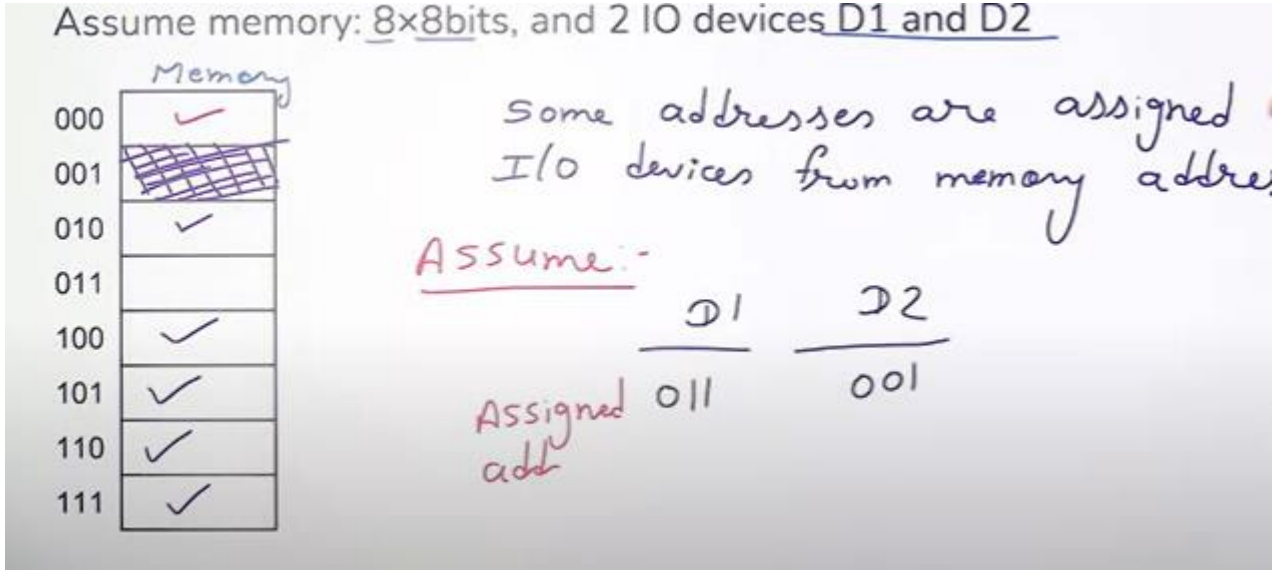
- No memory wastage

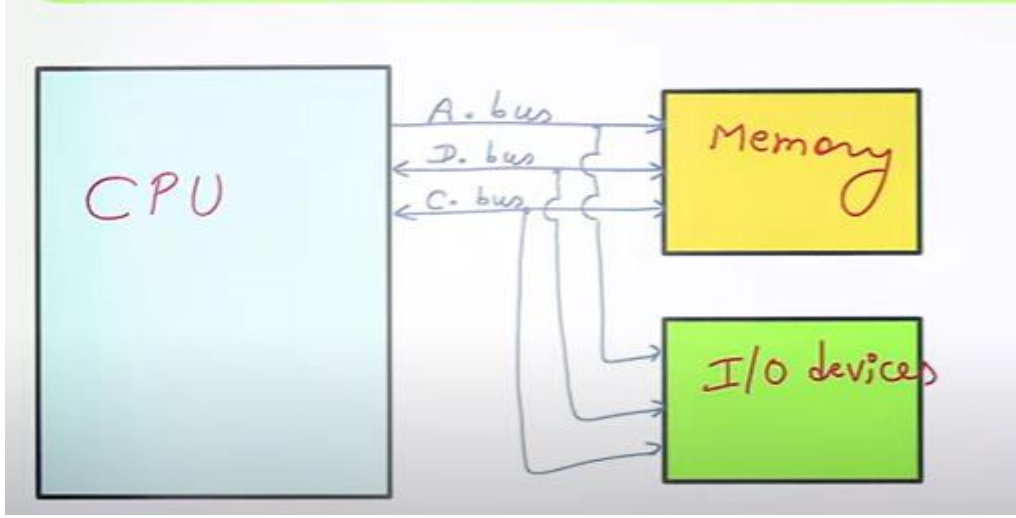
## 3) Common data, address and control bus:

- In this, some memory location will be assigned to the IO devices.

Disadvantage:

- Memory loss as it is assigned to IO.
- Also known as Memory mapped IO.
- Same control signal for both IO and memory.
- E.g. Principal room from classroom.
- Current system work on IO-Mapped-IO.



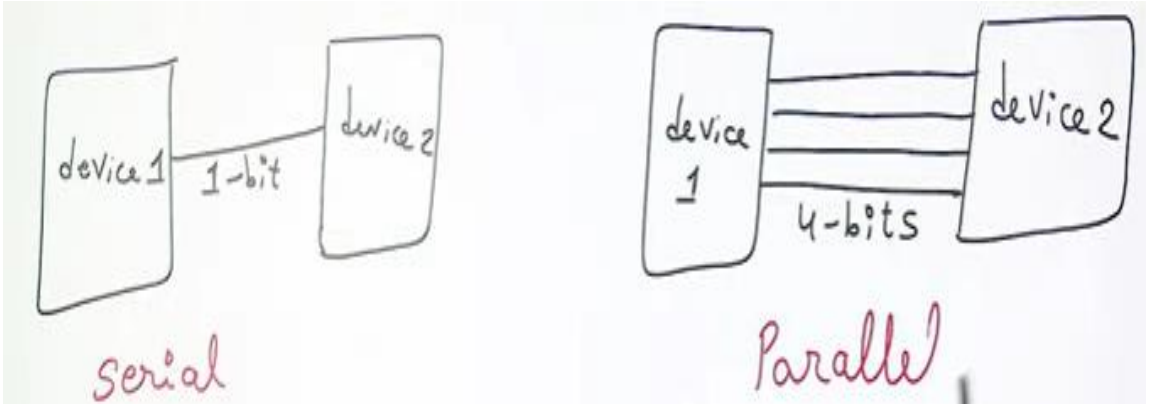

| Memory Mapped IO | IO Mapped IO |
|---|---|
| 1. Memory wastage | 1. No Memory wastage |
| 2. All Memory access instructions used for IO access also | 2. IO access instructions and memory access instructions are different |
| 3. No separate address space for IO | 3. IO have their own separate address space |
| 4. More Instructions for IO access | 4. Less Instructions for IO access |
| 5. More addressing modes for IO access | 5. addressing modes for IO access |

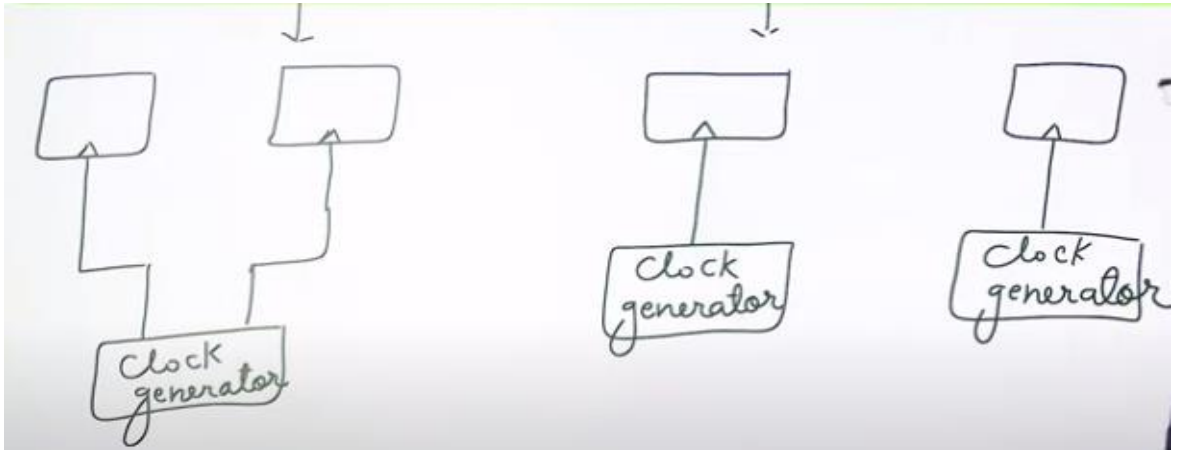
## Serial vs Parallel transfer

- Parallel- faster but costlier
- Serial- simple but slow.

## Synchronous and Asynchronous

- All devices operates at same clock pulse.
- In our system it is asynchronous as different device operates at different speed. If same clock generator, then CPU will work very slowly.
- Asynchronous – all device have individual clock generator. Thus, it required some synchronization to transfer data.
- To synchronize between CPU and IO , IO interface is required.



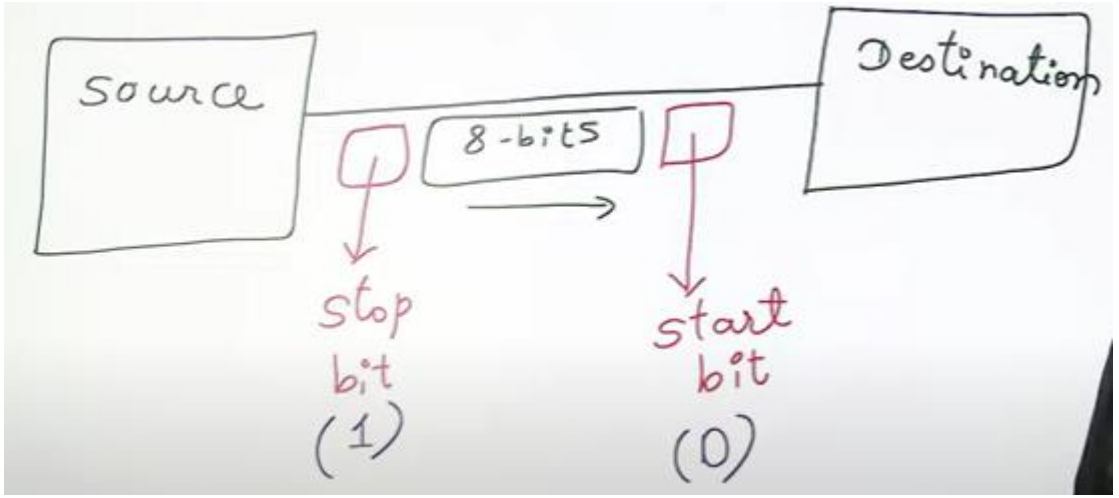



## Serial Asynchronous Transfer:

- To externally provide synchronization
- Send extra bit for initial and end bit.
- Data transfer in form of character.
- Disadvantage- extra bit led to inefficiency.

## Mode of transfer:

- Data between io devices and cpu.
- No ways was there by which IO devices can inform data transfer.
- A flag status is set by IO devices informing the data transfer. Thus, after an interval of time CPU checks each devices flag status to knowing data transfer.
- The setting of flag is done via program.
- Also known as programmed or program-controlled IO
- Disadvantage: Wastage of CPU time( asking every device even though one device has to transfer).




- ✗ No transfer means '1' state
- ✗ '0' (start bit) detected means transmission started
- ✗ Character bits follow start bit
- ✗ After last bit of character '1' (stop bit) will be sent

2) Interrupt initiated IO

- IO devices is given a control signal known as Interrupt.
- Thus when IO device wants to transfer data it sends interrupt signal to CPU.
- On receiving interrupt, CPU enable data transfer at the same time.
- Thus CPU save time.
- If IO wants to transfer direct data to memory it is not possible by both methods. It goes through cpu leading to more time.

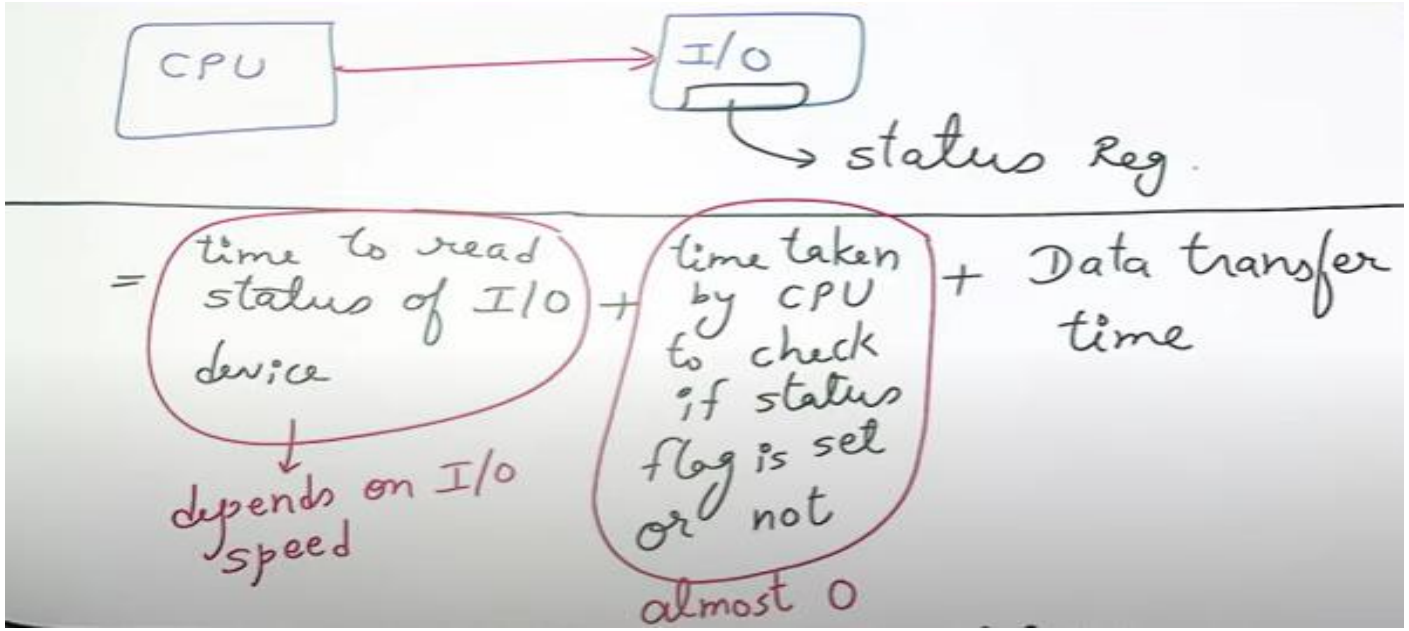
3) Direct Memory access(DMA).- between IO and memory.

- Need- bootstrap, page fault.

## Programmed IO or Program control IO

- Also known as Polling.
- Thus, with help of program, CPU check status of IO devices for data transfer.
- Disadvantage- lots of CPU time wastage.
- Default size of status register is 1 byte.

1. There is no any provision through which IO can inform to CPU about data transfer
2. IO sets its own status and waits
3. CPU runs program periodically and checks the status of each device one-by-one
4. If any device has its status set, then CPU performs data transfer for it.



# Interrupt IO:

- IO device will generate a signal (interrupt) to CPU that he has some data to transfer.
- CPU on receiving interrupt serve the current program and then serve the IO request
- E.g. Halting problem solving while someone call.
- E.g. Printer out of paper
- CPU serve the interrupt using Interrupt routine service(ISR).
- For different Interrupt different ISR.
- Thus a driver is installed for different ISR and are stored in MM.
- Vector address is location / reference / address of ISR in MM.


- IO device has a provision (Interrupt Signal) to inform to CPU about communication
- When CPU receives interrupt:
  1. It completes execution of current instruction
  2. Saves the status (PC, PSW etc.) of current process onto the stack
  3. Branches to service the interrupt
  4. Resumes the previous process by taking out the values from stack

- Interrupt is of 2 types

Vectored- IO Device send interrupt and vector

E.g. – father send u to buy thing along with money and shop address

Non-Vectored (Scalar)- IO Device only send interrupt.

- E.g.- father send to buy thing with money only. So u search the location with google (large database).
- CPU will first execute default service routine. This is will let CPU know the actual ISR location.
- In vectored first CPU accept the interrupt then only vector address is provided.
- Vector is passed through data bus.

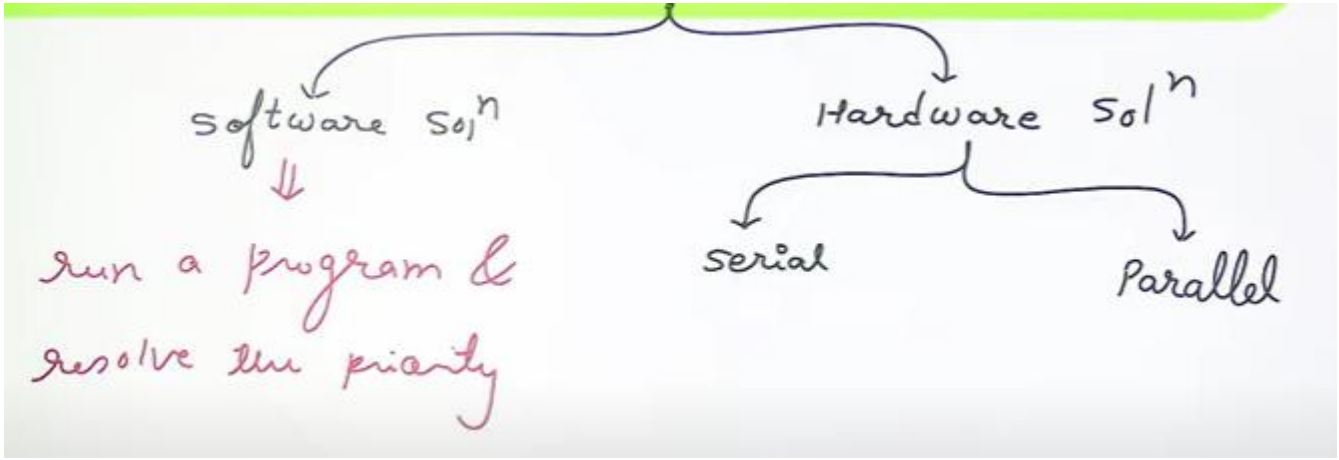
Maskable and non-maskable: maskable is accepted or rejected by CPU while non-maskable is always accepted by CPU.

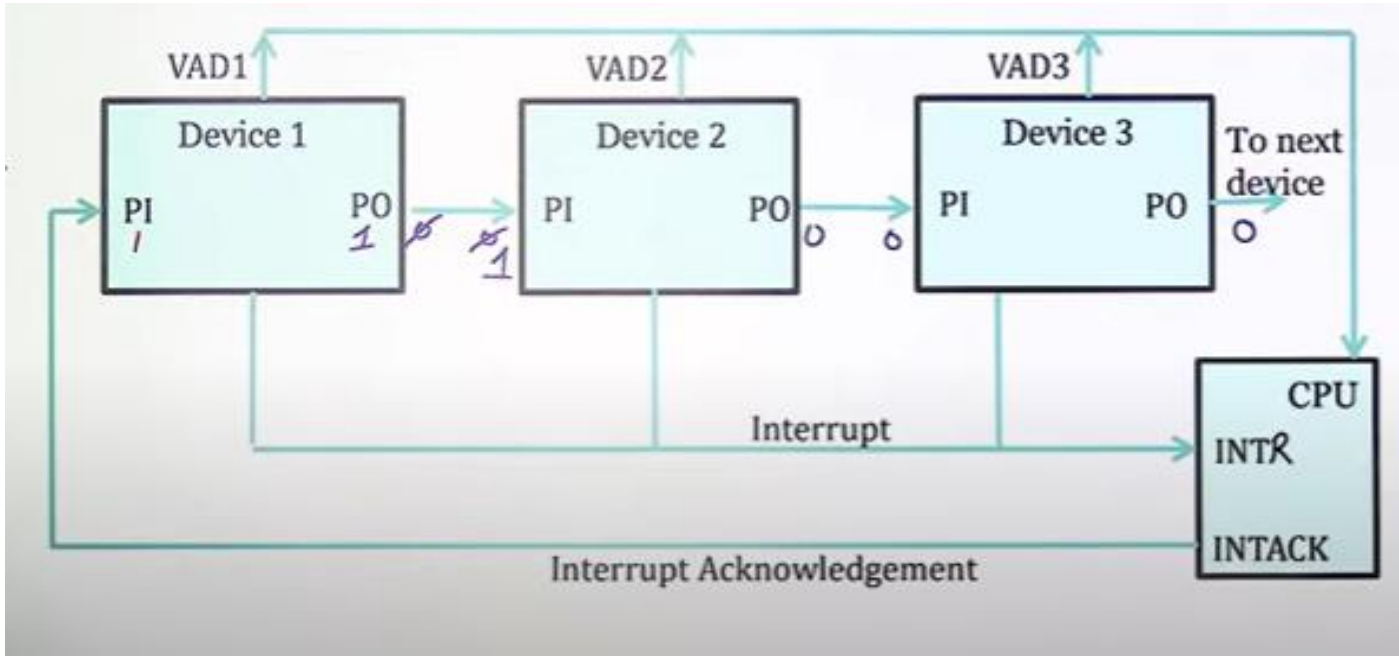
Internal vs external interrupt: External interrupt is generated by IO devices, while internal interrupt is generated due to unexpected error.(Page fault, system call).

- When two or more interrupted generated simultaneously, then highest priority device interrupt will be served first.

## Priority Based Interrupt Handling

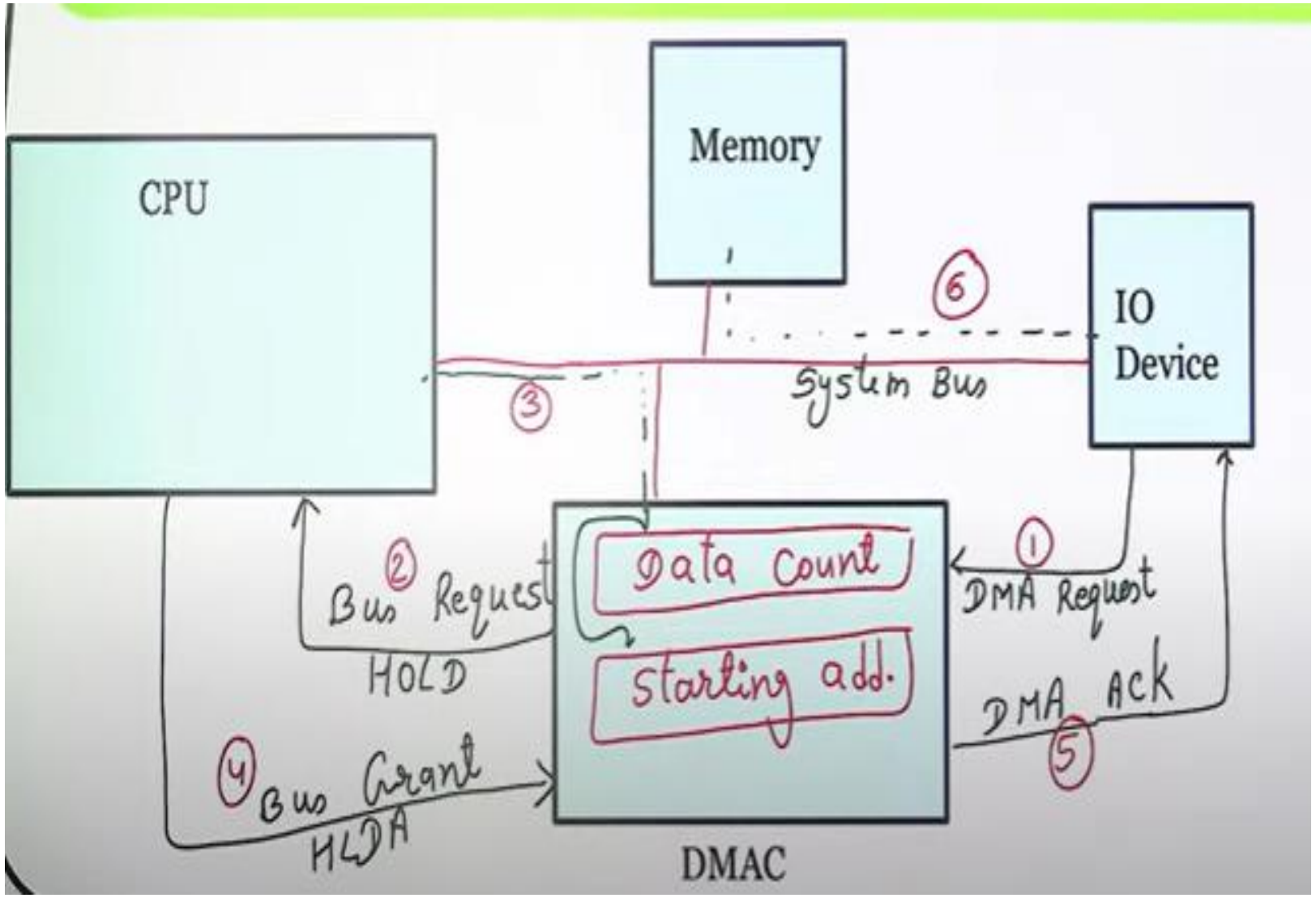
- In serial one by one device priority is checked while in parallel priority of all devices are checked once and highest priority is served first.
- Serial based solution is also known as Daisy chaining.
- All devices are connected serially (with decreasing order of priority)
- INTR- interrupt request INTACK- interrupt, acknowledgment, PI-priority in, PO-priority out, VAD-vector address.
- First ACK will be sent to device 1.
- Device 1 on receiving ACK send VAD to CPU. CPU then serve interrupt of Device 1.
- If still CPU is receiving INTR, then more interrupt from other devices are there to serve.
- Device closest to CPU will get highest priority in case of Daisy Chaining.



# Direct Memory Access:

- Enable data transfer between I/O and memory without intervention of CPU.
- Need a hardware to implement DMA transfer: DMA controller(DMAC).
- DMAC is responsible for enabling data transfer between IO and memory.
- DMAC need control of system bus to transfer data but control of bus is under CPU.
- Bus controlled is given to DMAC using bus grant or HLDA signal.



- DMAC doesn’t know how much data to transfer. It is decided informed by CPU to DMAC to transfer how much data/time or where to transfer data in memory IO can transfer.
- Starting address- memory address from where data transfer should be started.
- Data count- number of words or bytes to be transferred (depending upon memory addressable).
- E.g. playing movie of pendrive.
- During transfer of bus control, CPU does thing which doesn’t require bus. This means CPU will be blocked.
- DMAC will give address and control signal for operation between IO and memory.
- To avoid CPU context switching, we need DMAC. During blocked state, CPU doesn’t change its state but remain idle. Thus lot of times save in context switching.
- DMAC is special type processor which generates address and control signal for data transfer between IO and memory.
- Entire data transfer is not between IO and Memory as we cannot keep it idle for long time.
- Thus system bus is shared between CPU and DMAC.

# Mode of DMA transfer

- It defines for how time and when the DMAC will get control of system bus.
- Generally three types
- Burst mode
- Cycle stealing mode
- Interleaving mode

Burst mode:

- A burst of data (generally 1-2 KB data) in one time before CPU take control over the buses.
- e.g. borrowing of calculator during exam.

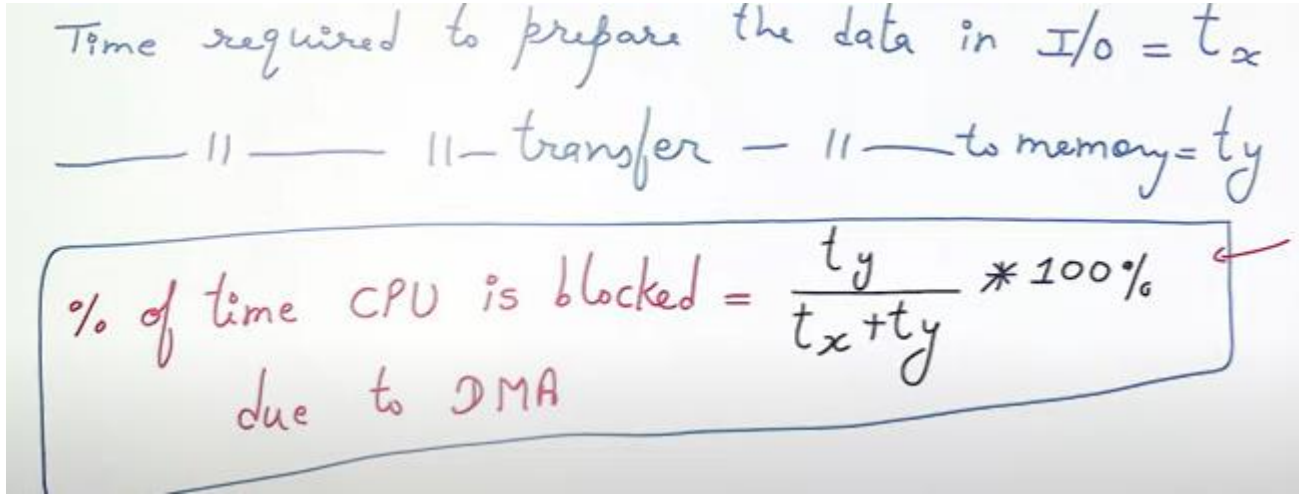
Cycle stealing mode:

- Since IO devices is very slow compared to CPU, lot of time is consumed in preparing data. Thus it is not good for CPU to transfer control to DMAC and sit idle.
- Thus after knowing that IO device has prepared the data, then CPU will assign the system bus to DMAC for one memory cycle. During the cycle prepared word or byte is transferred to memory.
- After one memory cycle, again CPU take backs the control of system bus and so on.

Interleaving mode:

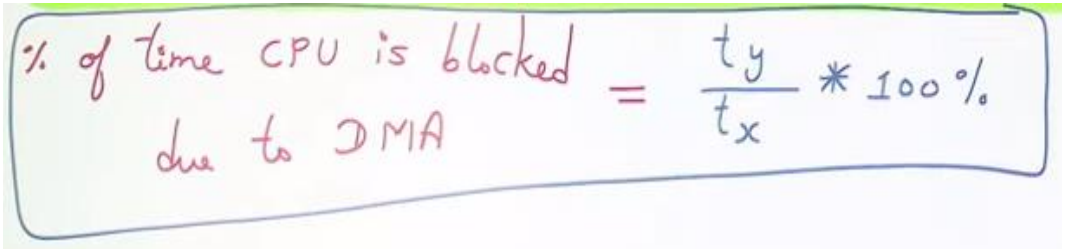
- Whenever CPU perform internal operations and does not require the buses then only DMAC will be given the system bus.
- Hence CPU will never be blocked due to DMA.
- The CPU will inform the DMAC that the number of memory cycle.

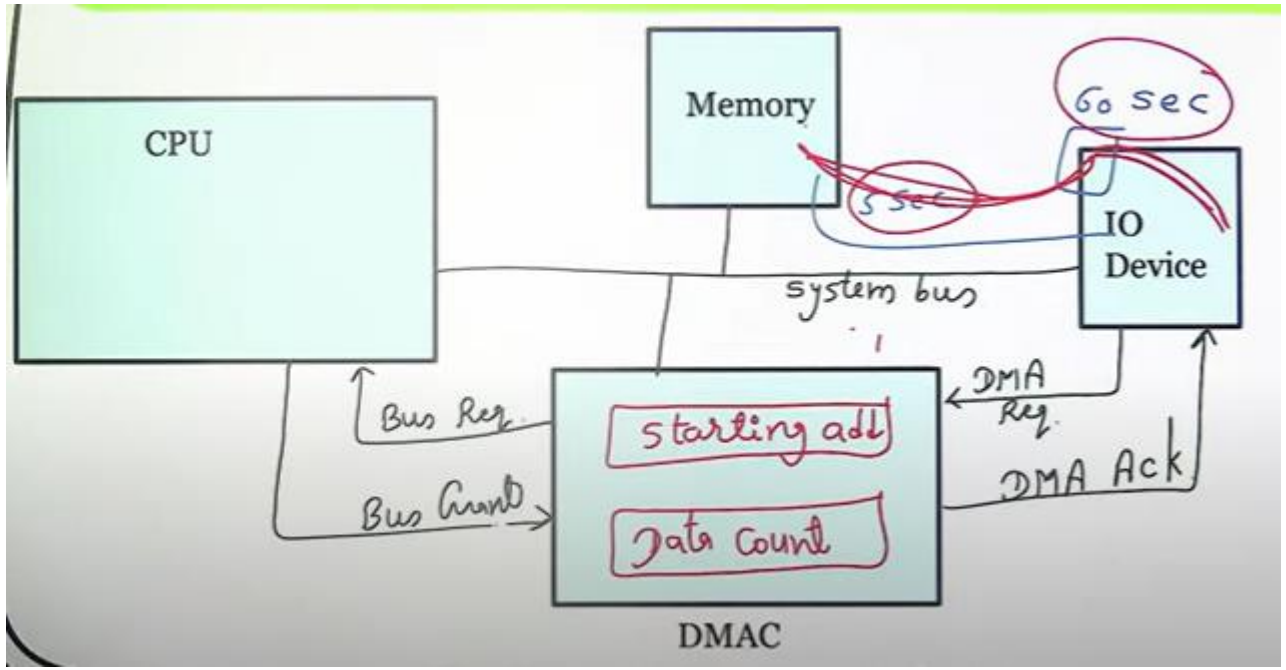
Burst Mode:



## Cycle stealing mode:

In this mode, only first time whole time is taken to prepare. After that data preparation is parallelly with data transfer time.




## Interleaving mode: