VECTORED AND PRIORITY INTERRUPTS

Interrupt

- An interrupt is a signal from a device attached to a computer or from a program within the computer that causes the CPU to stop its normal program execution and perform service related to the event.
- **Maskable Interrupt**: It is a hardware interrupt that may be ignored by set/reset a bit in an interrupt mask register's (IMR) bit-mask.
- **Non-maskable Interrupt**: is a hardware interrupt that does not have a bit-mask associated with it - meaning that it can never be ignored. NMIs are often used for timers.

Interrupt Cycle

The Interrupt enable flip-flop (IEN) can be set or cleared by program instructions.

A programmer can therefore allow interrupts (clear IEN) or disallow interrupts (set IEN)

At the end of each instruction cycle the CPU checks IEN and IST. If either is equal to zero, control continues with the next instruction. If both = 1, the interrupt is handled.

Interrupt micro-operations:

SP←SP – 1	(Decrement stack pointer)
M[SP] ← PC	Push PC onto stack
INTACK ← 1	Enable interrupt acknowledge
PC ←VAD	Transfer vector address to PC
IEN ←0	Disable further interrupts
Go to fetch next instruction	

Interrupt Service Routine

An interrupt handler, also known as an interrupt service routine (ISR), is a callback subroutine in an operating system whose execution is triggered by the reception of an interrupt. Interrupt handlers have a multitude of functions, which vary based on the reason the interrupt was generated.

Interrupt Overhead

The interrupt overhead is caused by context switching (storing and restoring the state of CPU)

On interrupt handler entry, the context of the current process and its thread must be saved. On exit, it must be restored.

On handler entry, memory locations different from the memory locations in the cache are used, and therefore cache updates are required.

Vectored and Non Vectored Interrupts

- Non-vectored interrupt
  - The device has to supply the address of the subroutine to the processor
  - Ex: Call instruction, INTR
- Vectored interrupt
  - The address of the subroutine is already known to the processor
  - RST 5.5 – when this interrupt is received, the processor saves the content of PC into stack and branches to 002CH address.

Priority Interrupt

- Establishes priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously.
- The system may also determine which conditions are permitted to interrupt the computer while another interrupt is being serviced.
- Priority can be established by software or hardware

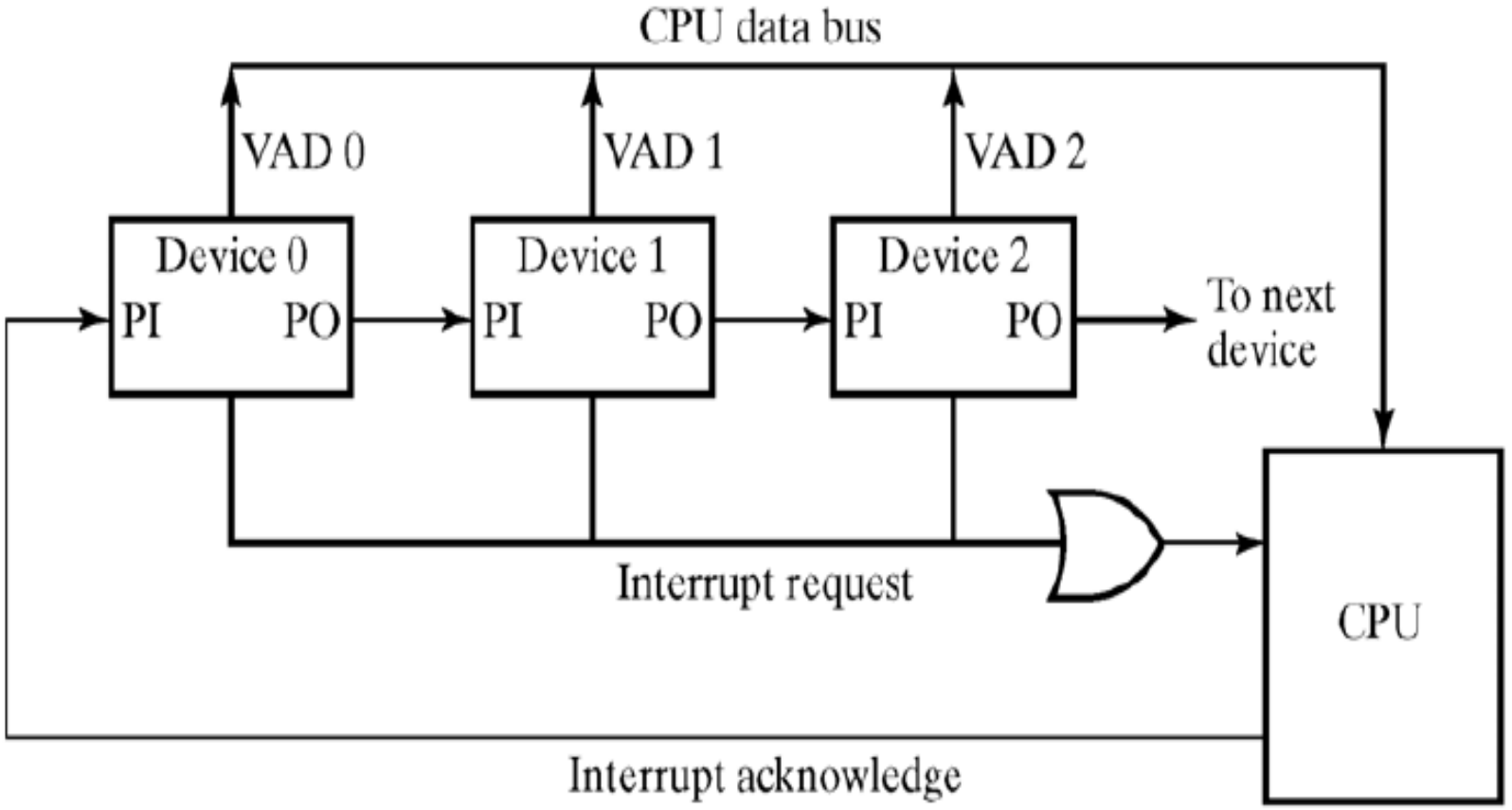
Polling-S/W

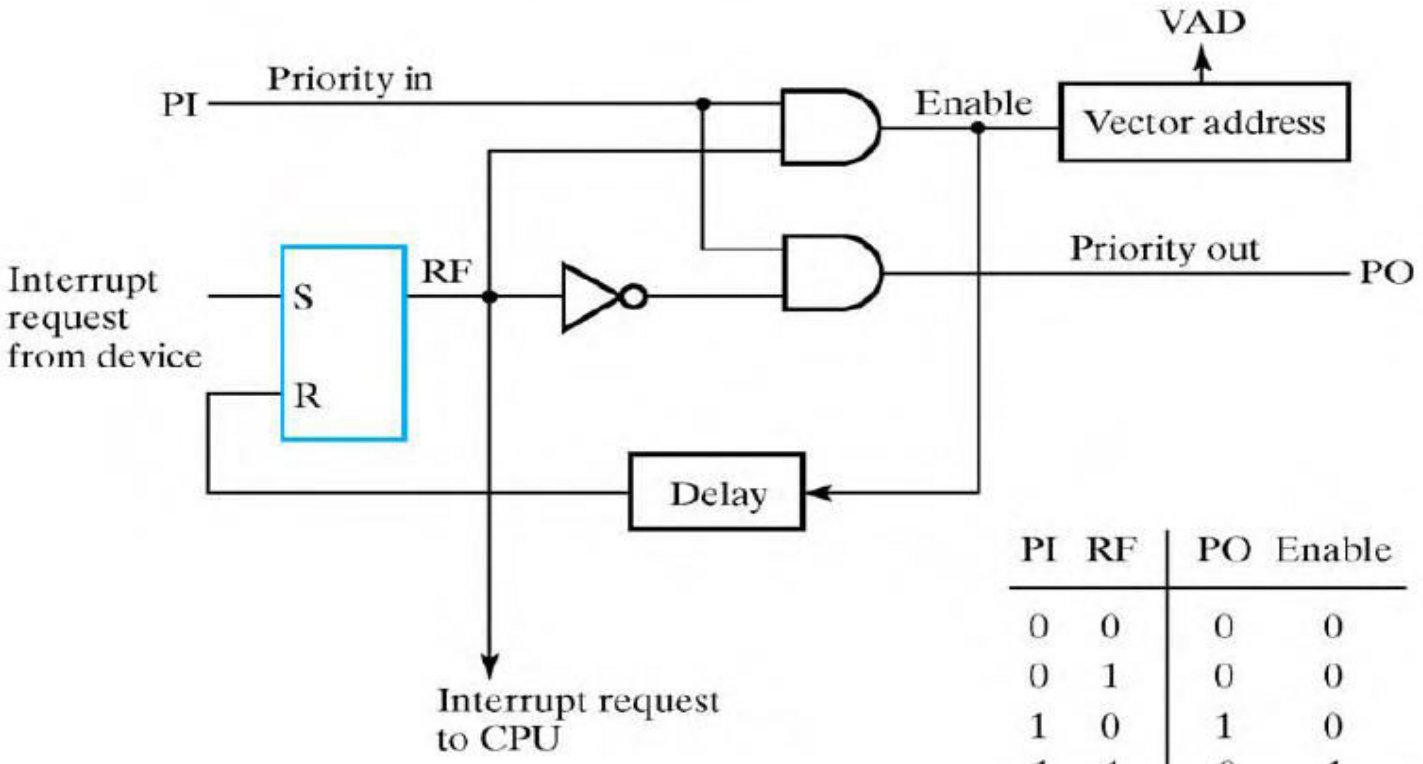
- Software Priority – polling – common branch address for all interrupts. Program to perform the test will be stored in this common branch address.
- Highest priority source is tested first and if its interrupt signal is on, control branches to a service routine for this source.
- Otherwise, the next-lower priority source is tested and so on.
- **Disadvantage – if many interrupts**, time required to poll them may exceed the time available to service the I/O device.

Vectored Interrupts-H/W

- Then, hardware priority interrupt unit can be used to speed up the operation.
- Hardware priority interrupt handler – accepts interrupt request from many sources, determines which of the incoming requests has the highest priority and issues an interrupt request to the computer.
- Here, each interrupt source has its own interrupt vector to access its own service routine directly.
- Hardware priority interrupts – Serial or Parallel connection of interrupt lines.

Serial - Daisy chaining





PI	RF	PO	Enable
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

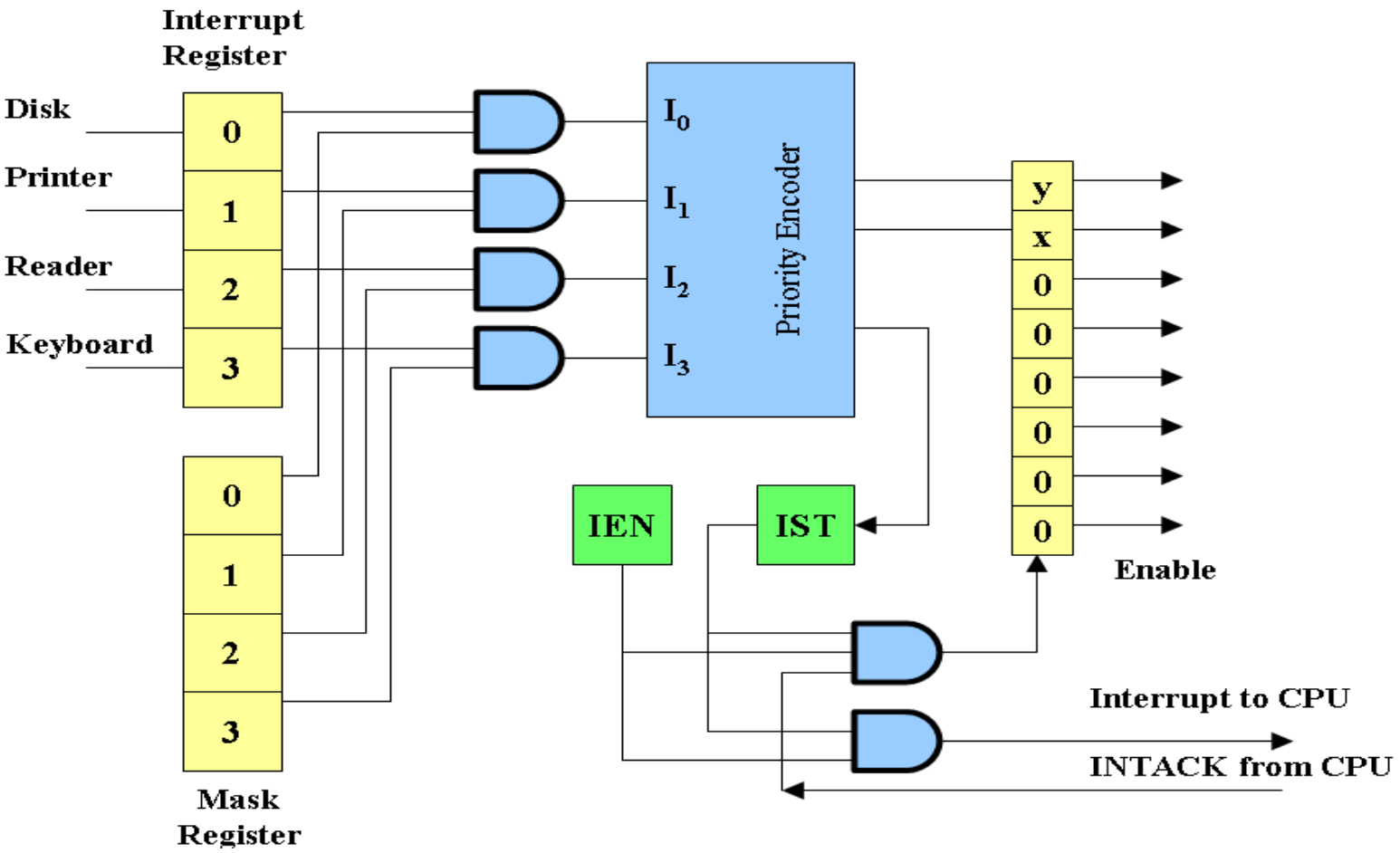
One stage of daisy chain priority interrupt scheme

If PI = 0, both PO and the enable line to VAD (vector address) are equal to 0, irrespective of RF.

- If PI = 1 and RF = 0, then PO = 1 and the vector address is disabled => passes the acknowledge signal to the next device through PO.
- The device is active when PI = 1 and RF = 1. This condition places a 0 in PO and enables the vector address for the data bus.
- The RF flip-flop is reset after a sufficient delay to ensure that the CPU has received the vector address.

Parallel Priority Interrupt

- Uses a register whose bits are set separately by the interrupt signal from each device.
- Priority is established according to the position of the bits in the register.
- Mask register is used to disable lower priority interrupts while a higher priority device is being serviced.
- It can also provide a facility that allows a high-priority device to interrupt the CPU while a lower priority device is being serviced



- Mask register has a same number of bits as the interrupt register.
- By means of program instructions, it is possible to set or reset any bit in the mask register.

- The priority encoder generates two bits of the vector address, which is transferred to the CPU.
- Another output from the encoder sets an interrupt status flip-flop IST when an interrupt that is not masked occurs.
- The interrupt enable flip-flop IEN can be set or cleared by the program to provide an overall control over the interrupt system.
- IST ANDed with IEN provide a common interrupt signal for the CPU.
- The INTACK signal from the CPU enables the bus buffers in the output register and a vector address VAD is placed into the data bus.

Priority encoder

Circuit that implements the priority function.

Logic – if two or more inputs arrive at the same time, the input having the highest priority will take precedence.

Boolean functions

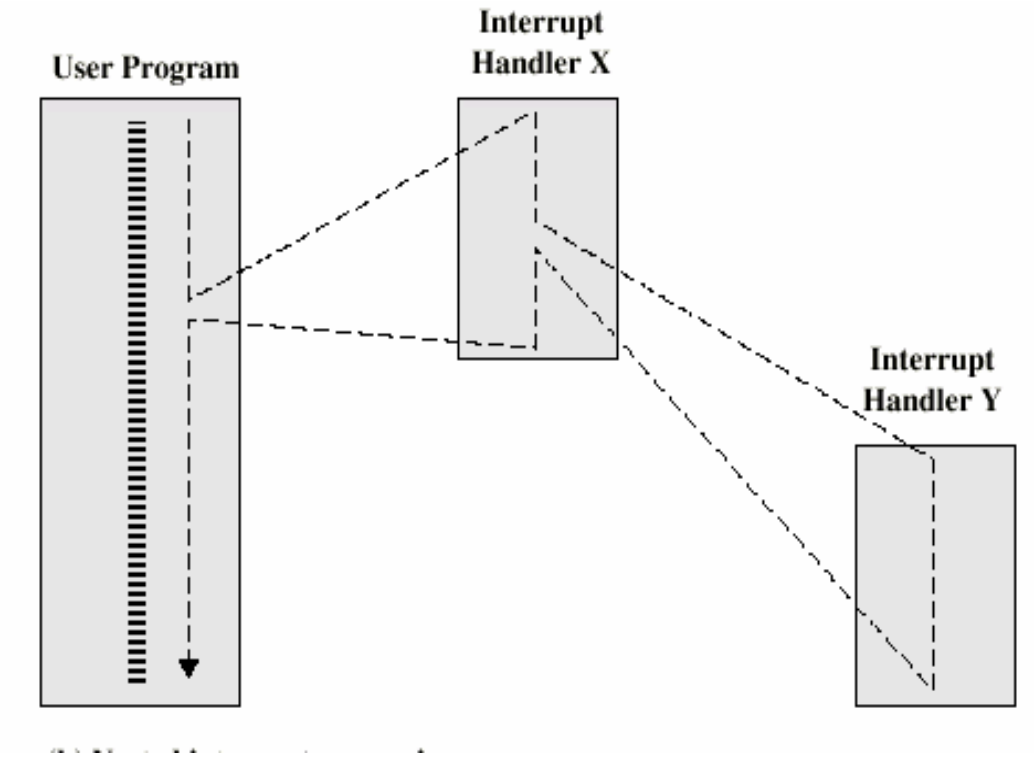
$X = I'_0 I'_1$ $\quad Y = I'_0 I_1 + I'_0 I'_2$ $\quad IST = I_0 + I_1 + I_2 + I_3$

Inputs			
I_0	I_1	I_2	I_3
1	d	d	d
0	1	d	d
0	0	1	d
0	0	0	1
0	0	0	0

Outputs		
d	Y	IST
0	0	1
0	1	1
1	0	1
1	1	1
d	d	0

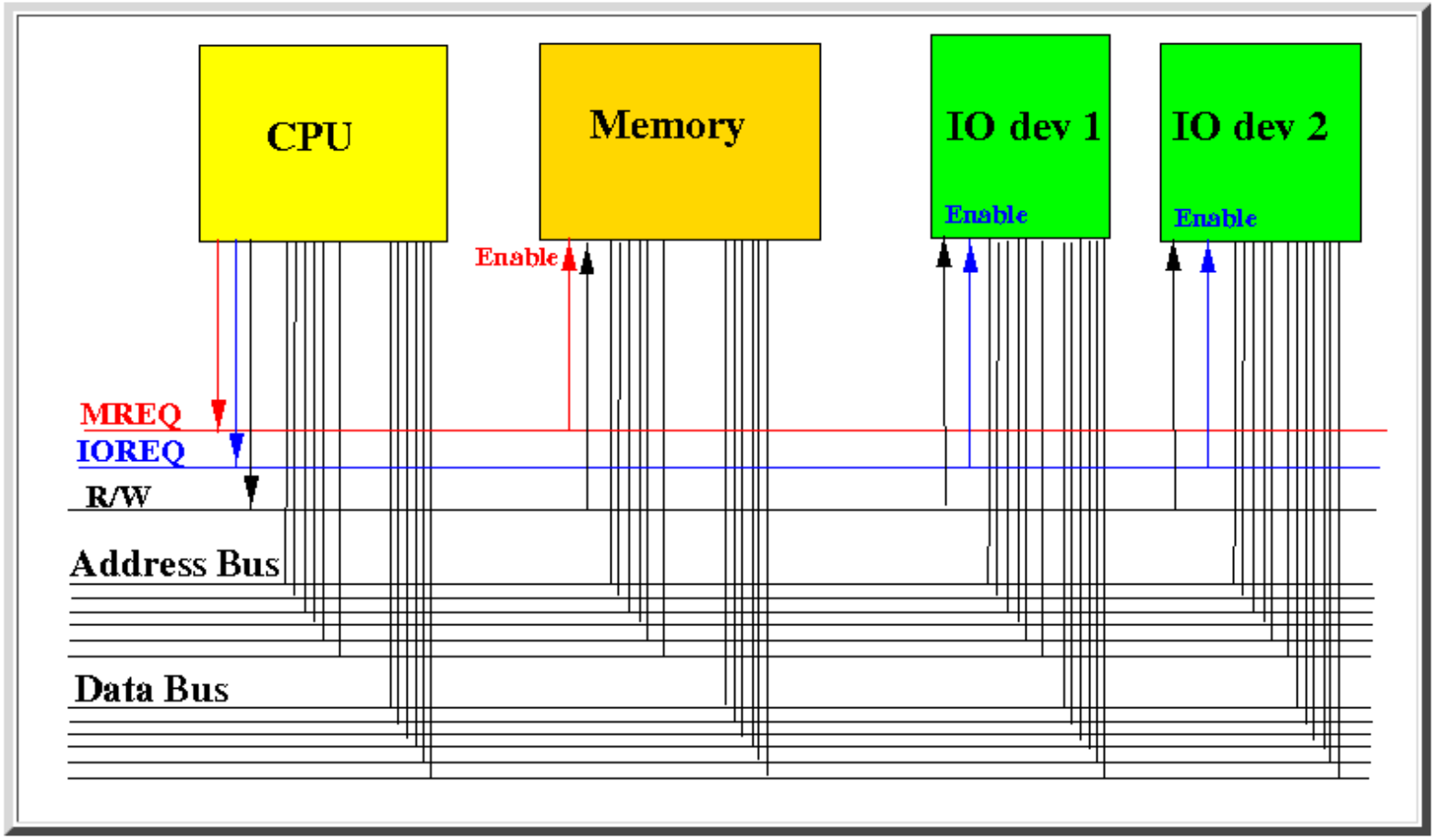
Interrupt nesting

- An interrupt can happen while executing an ISR. This is called in*terrupt* nesting.



Bus System

- How devices are connected inside a compute:



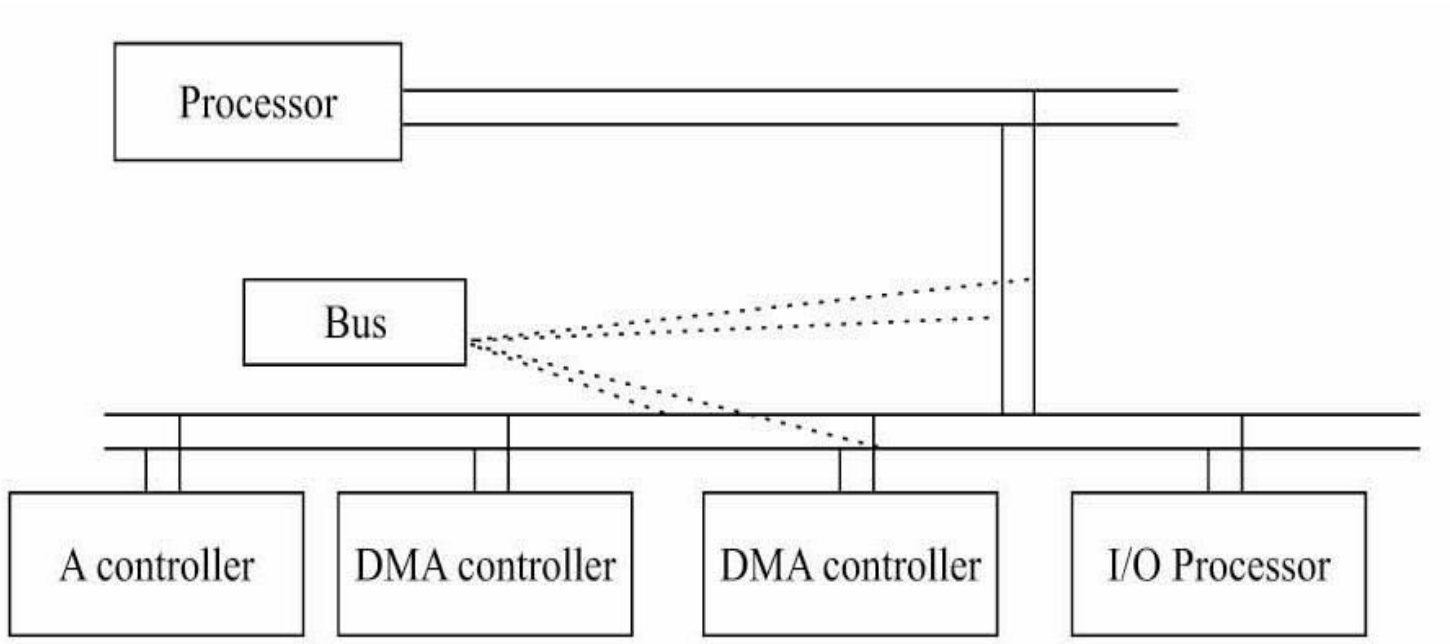
Synchronous Bus

- Synchronous bus (e.g., processor-memory buses)
  - Includes a clock in the control lines and has a fixed protocol for communication that is relative to the clock
  - Advantage: involves very little logic and can run very fast
  - Disadvantages:
    - Every device communicating on the bus must use same clock rate

Asynchronous Bus

- Asynchronous bus (e.g., I/O buses)
  - It is not clocked, so requires a handshaking protocol and additional control lines (ReadReq, Ack, DataRdy)
  - Advantages:
    - Can accommodate a wide range of devices and device speeds
  - Disadvantage: slow(er)

Bus Arbitration



A bus (any bus) **cannot** be used by **more than one device** at one time due to electrical properties of the devices.

Bus Arbitration

- Before any device is allowed to perform a read/write operation using the system bus, it must first **obtain permission.**
- A device that **starts** a **read/write operation** is called a **master device**
- The device that it "talks" to is called the **slave device**

Bus Arbitration

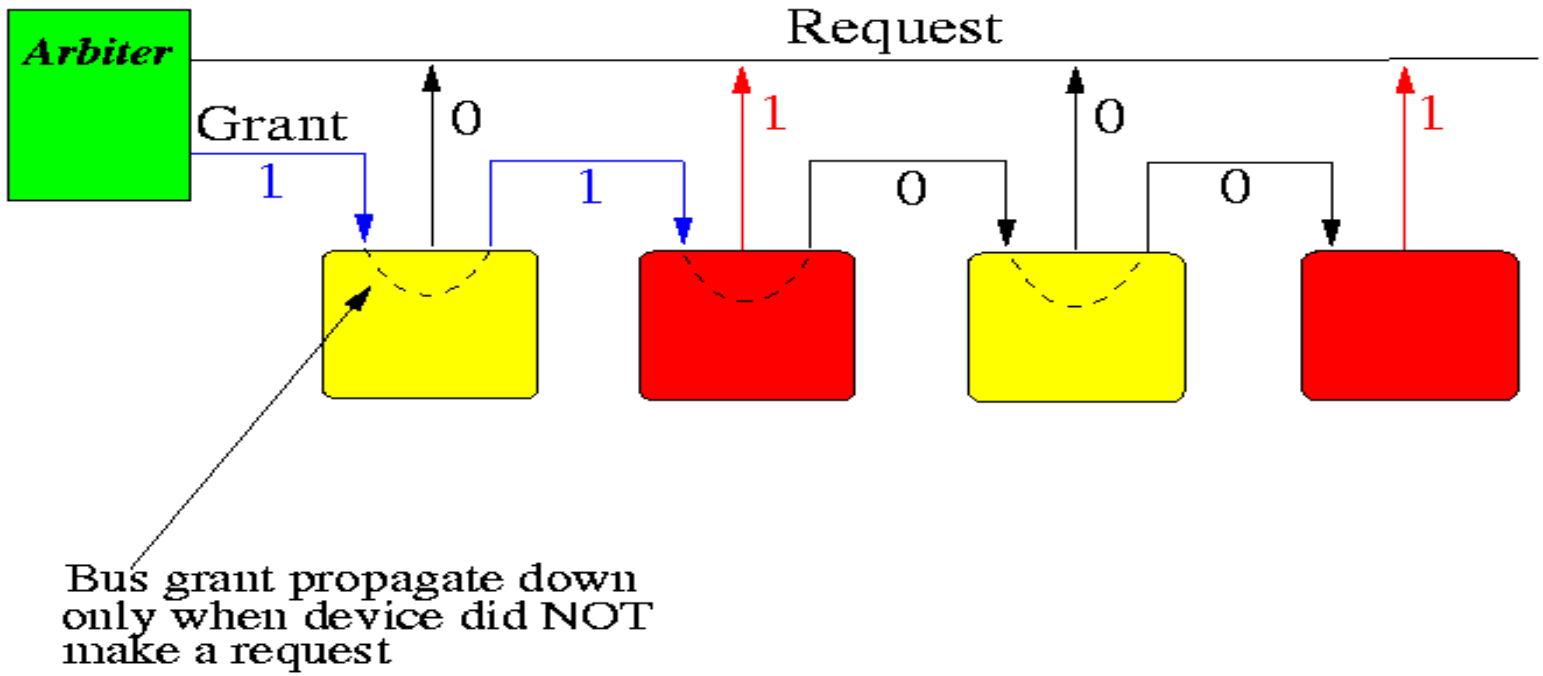
- Only one processor or controller can be **bus master**
- The bus master– the controller that has access to a bus at an instance.
- Any one controller or processor can be the bus master at the given instance (s).

3 BUS ARBITRATION METHODS

1. Daisy Chain
2. Independent Bus Requests and Grant
3. Polling

Daisy Chained

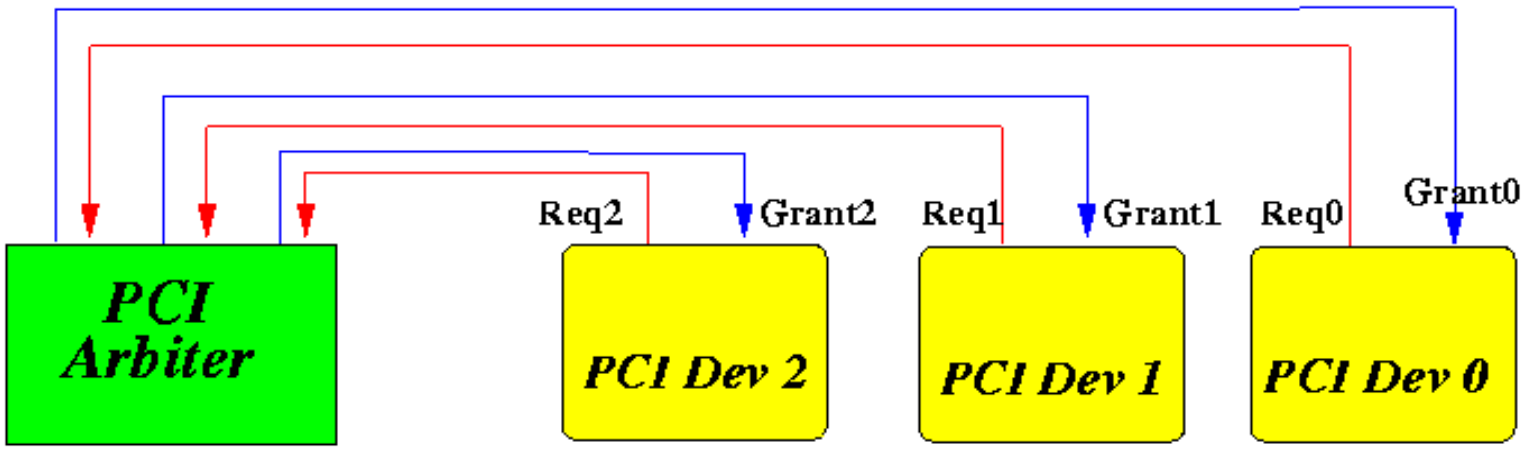
- Daisy-chained (cheap - requires no special hardware)



- The daisy chain consists of wires connecting the devices in a pre-defined order
- This ordering defines the "pecking order" of the devices.

Independent Bus Requests and Grant

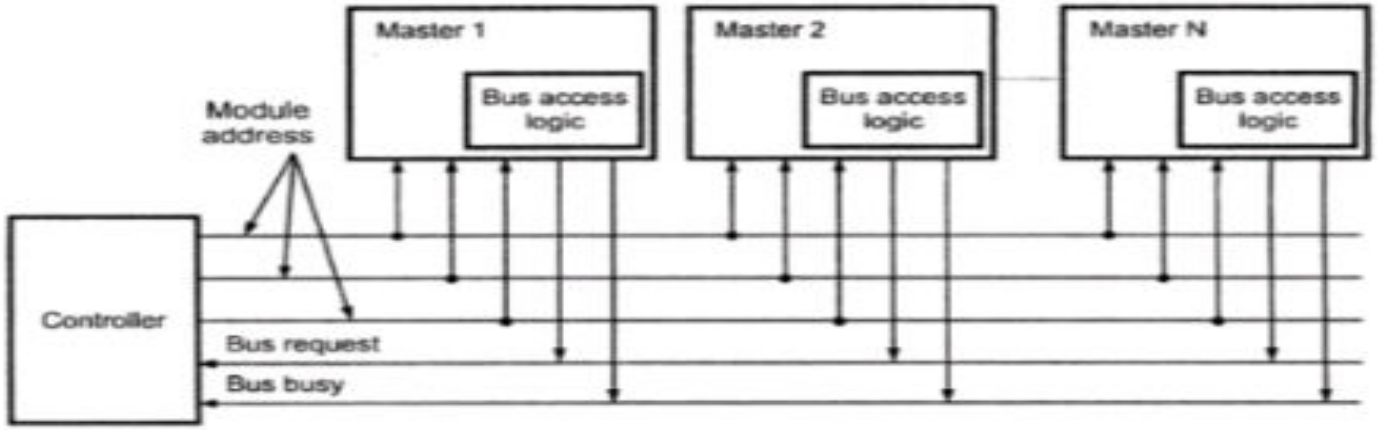
Centralized Arbiter



- The logic of the bus arbiter is simple and and depends on how the priorities are assigned.

Req2	Req1	Req0	Grant2	Grant2	Grant0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

Polling



•In this the controller is used to generate the addresses for the master. Number of address line required depends on the number of master connected in the system.

•In response to the bus request controller generates a sequence of master address. When the requesting master recognizes its address, it activated the busy line and begins to use the bus.

Daisy chaining, the centralized controller always sending bus control to highest priority, which passes it to next if bus access not required.

Independent request method, the centralized controller listens to requests of each device individually and grant access to the bus on resolving its priority.

Polling methods, the centralized controller does the polling of the devices and grant access to that bus which requests it on receiving the poll address.

References

- M. M. Mano, Computer System Architecture, Prentice-Hall