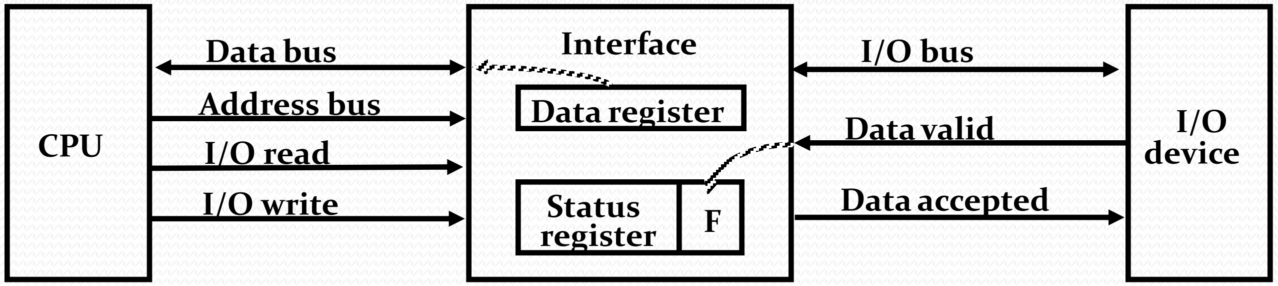
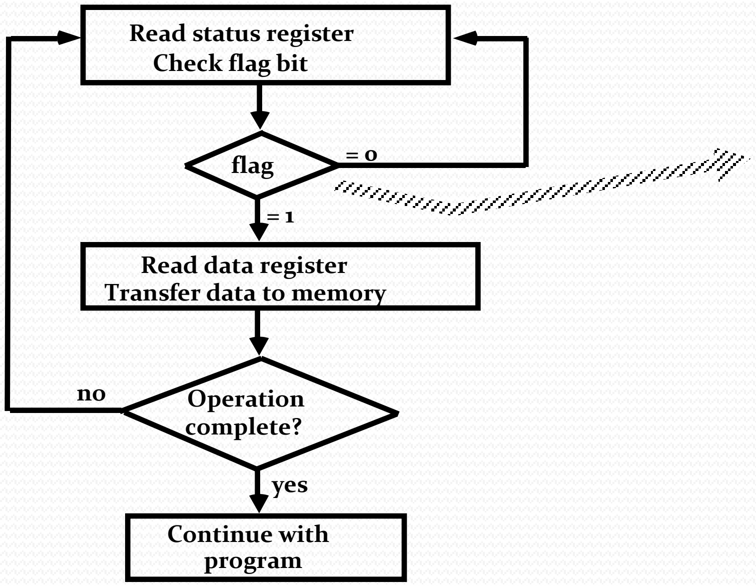
Modes of Transfer – Programmed I/O

3 different Data Transfer Modes between the central computer(CPU or Memory) and peripherals;

- **Program-Controlled I/O**
- **Interrupt-Initiated I/O**
- **Direct Memory Access (DMA)**

Program-Controlled I/O(Input Dev to CPU)





Polling or Status Checking

- Continuous CPU involvement
- CPU slowed down to I/O speed
- Simple
- Least hardware

Modes of Transfer – Interrupted I/O & DMA

Interrupt Initiated I/O

- Polling takes valuable CPU time
- Open communication only when some data has to be passed -> *Interrupt*.
- I/O interface, instead of the CPU, monitors the I/O device
- When the interface determines that the I/O device is ready for data transfer, it generates an *Interrupt Request* to the CPU
- Upon detecting an interrupt, CPU stops momentarily the task it is doing, branches to the service routine to process the data transfer, and then returns to the task it was performing

DMA (Direct Memory Access)

- Large blocks of data transferred at a high speed to or from high speed devices, magnetic drums, disks, tapes, etc.
- DMA controller Interface that provides I/O transfer of data directly to and from the memory and the I/O device
- CPU initializes the DMA controller by sending a memory address and the number of words to be transferred
- Actual transfer of data is done directly between the device and memory through DMA controller

-> Freeing CPU for other tasks

Which of the following is not a Mode of data transfer?

a)Program-Controlled I/O
b)Interrupt-Initiated I/O
c)Direct Memory Access
d)None of the above

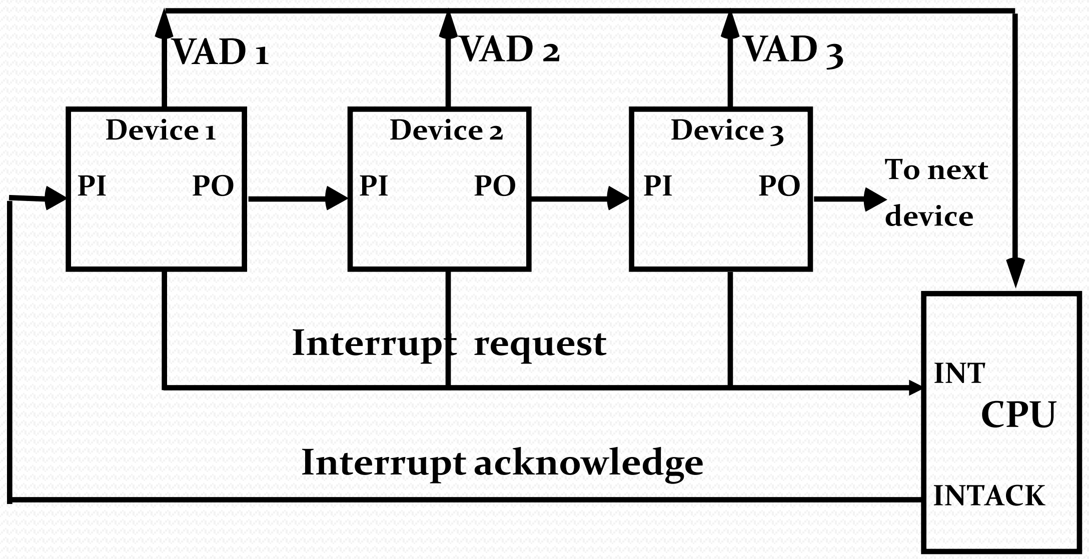
Priority Interrupts

Priority
- Determines which interrupt is to be served first when two or more requests are made simultaneously
- Also determines which interrupts are permitted to interrupt the computer while another is being serviced
- Higher priority interrupts can make requests while servicing a lower priority interrupt

Priority Interrupt by Software(Polling)
- Priority is established by the order of polling the devices(interrupt sources)
- Flexible since it is established by software
- Low cost since it needs a very little hardware
- Very slow

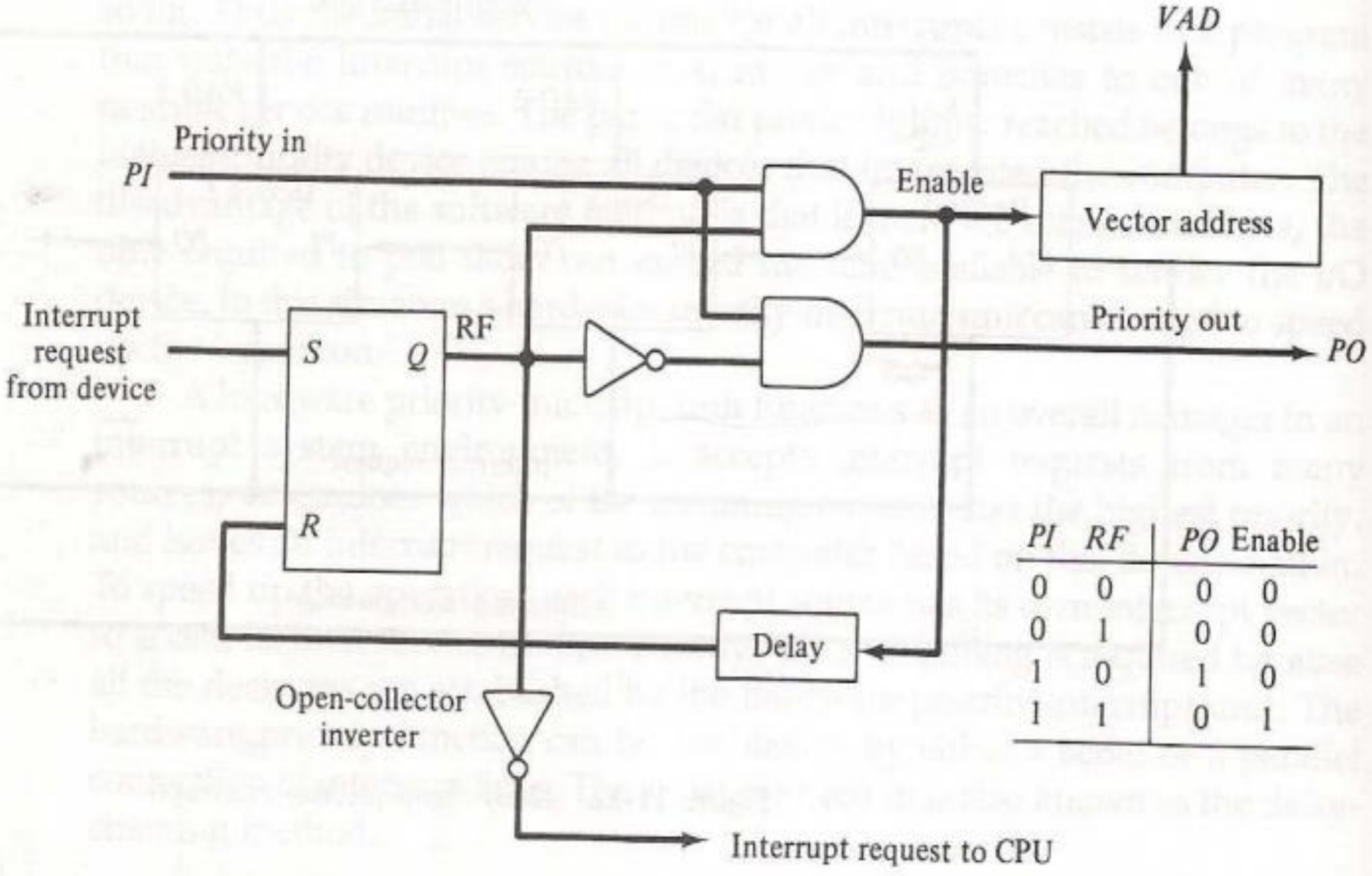
Priority Interrupt by Hardware
- Require a priority interrupt manager which accepts all the interrupt requests to determine the highest priority request
- Fast since identification of the highest priority interrupt request is identified by the hardware
- Fast since each interrupt source has its own interrupt vector to access directly to its own service routine

Hardware Priority Interrupts – Daisy Chain



* Serial hardware priority function
* Interrupt Request Line
- Single common line
* Interrupt Acknowledge Line
- Daisy-Chain

Interrupt Request from any device(>=1)
-> CPU responds by INTACK <- 1
-> Any device receives signal(INTACK) 1 at PI puts the VAD on the bus
Among interrupt requesting devices the only device which is physically closest to CPU gets INTACK=1, and it blocks INTACK to propagate to the next device



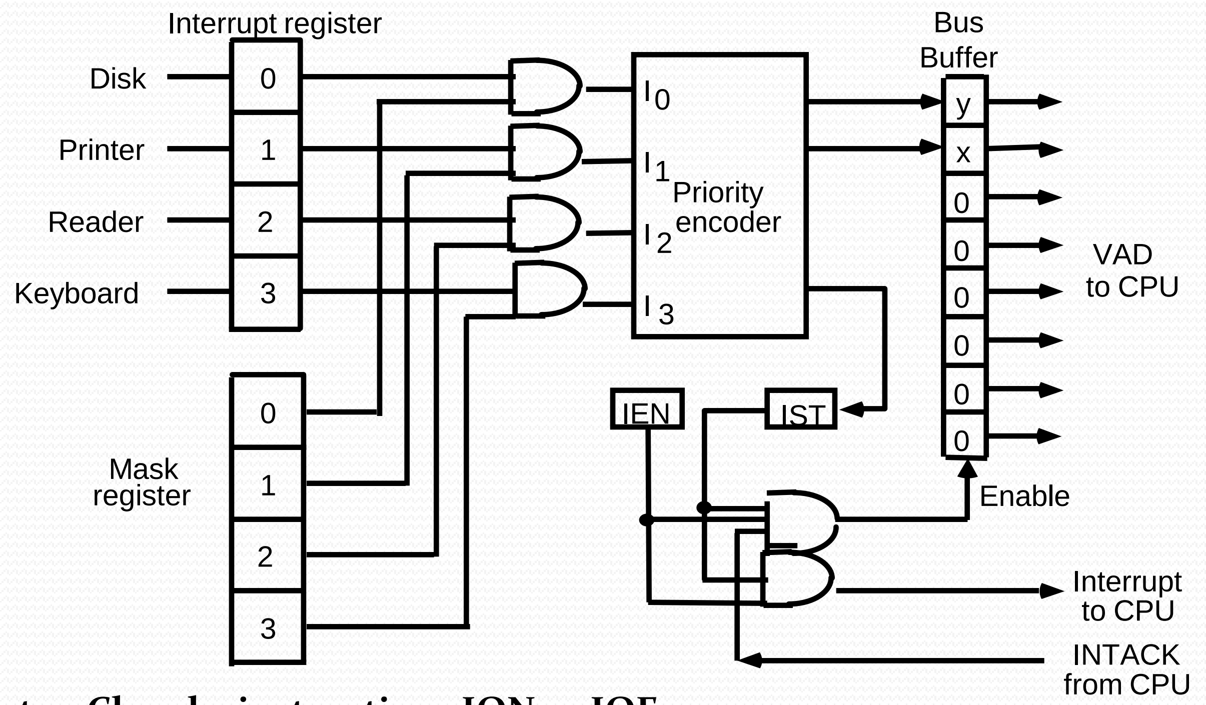
PI	RF	PO	Enable
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Figure 11-13 One stage of the daisy-chain priority arrangement.

What does VAD stands for?

a) Vectored Address Device
b) Vector Address Development
c) Vector Address
d) None of the above

Parallel Priority Interrupts



IEN: Set or Clear by instructions ION or IOF
IST: Represents an unmasked interrupt has occurred. INTACK enables tristate Bus Buffer to load VAD generated by the Priority Logic

Interrupt Register:
- Each bit is associated with an Interrupt Request from different Interrupt Source - different priority level
- Each bit can be cleared by a program instruction

Mask Register:
- Mask Register is associated with Interrupt Register
- Each bit can be set or cleared by an Instruction

Priority Encoder

Determines the highest priority interrupt when more than one interrupts take place

Priority Encoder Truth table

Inputs				Outputs			
I_0	I_1	I_2	I_3	x	y	IST	Boolean functions
1	d	d	d	0	0	1	
0	1	d	d	0	1	1	
0	0	1	d	1	0	1	$x = I_0' \; I_1'$
0	0	0	1	1	1	1	$y = I_0' I_1 + I_0' I_2'$
0	0	0	0	d	d	0	$(IST) = I_0 + I_1 + I_2 + I_3$