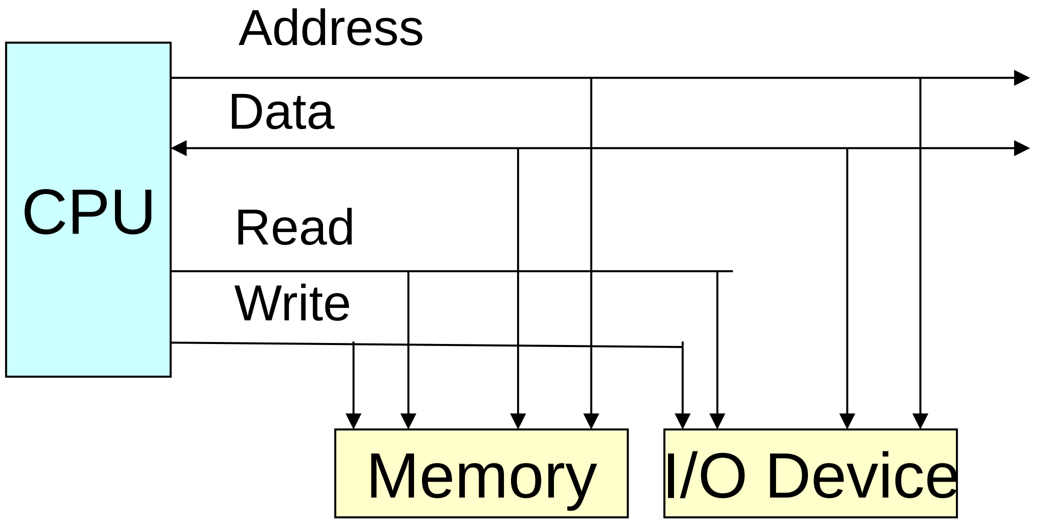
L41- DATA TRANSFER TECHNIQUES IN IO SUBSYSTEMS

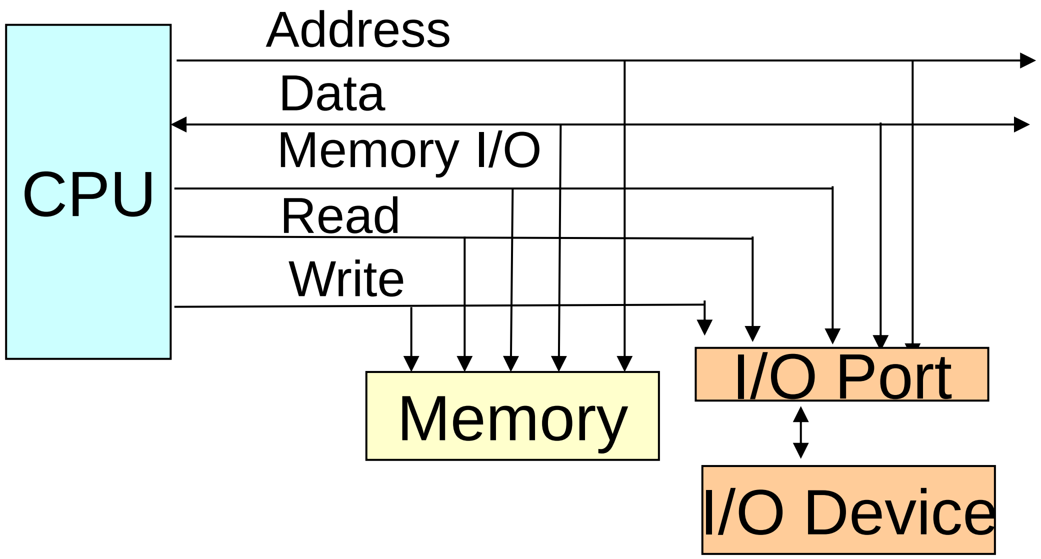
Objectives

- ❖ Explore the structure of an operating system's I/O subsystem
- ❖ Discuss the principles of I/O hardware and its complexity
- ❖ Provide details of the performance aspects of I/O hardware and various data transfer schemes

I/O Mapping

- CPU needs to talk to I/O
- **Memory mapped I/O**
  - Devices mapped to reserved memory locations - like RAM
  - Uses load/store instructions just like accesses to memory
- **I/O mapped I/O**
  - Special bus line
  - Special instructions





I/O Data Transfer techniques

- Polled I/O
- Interrupt-Driven I/O
- Direct Memory Access (DMA)

Polled I/O

- CPU periodically check I/O status (polling)
    - If device ready, do operation
    - If error, take action
- CPU has direct control over I/O
    - Sensing status
    - Read/write commands
    - Transferring data
- CPU waits for I/O module to complete operation
- Wastes CPU time

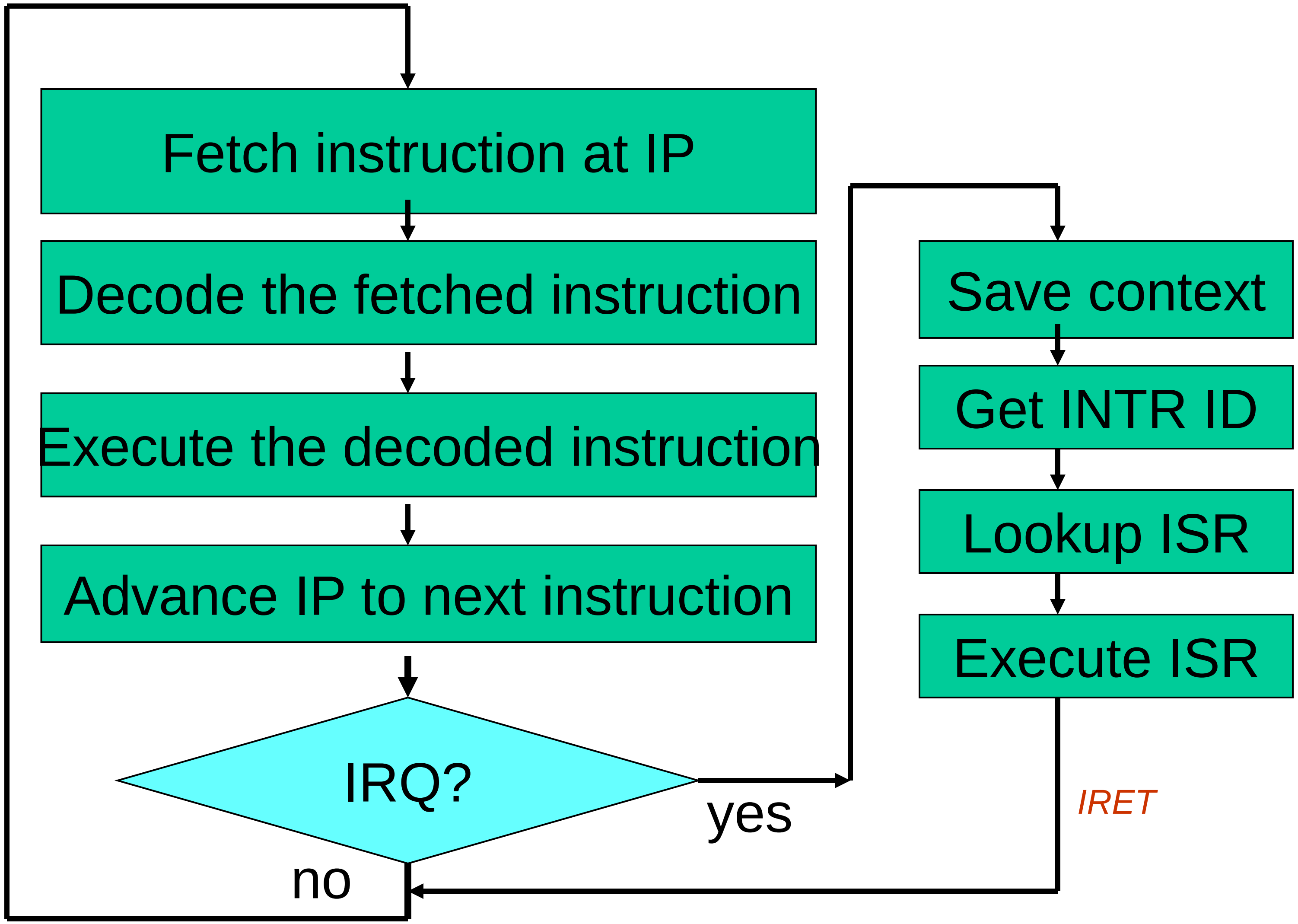
Steps in Polled I/O

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically (polling)
- CPU may wait or come back later
- I/O module does not inform CPU directly
- I/O module does not interrupt CPU

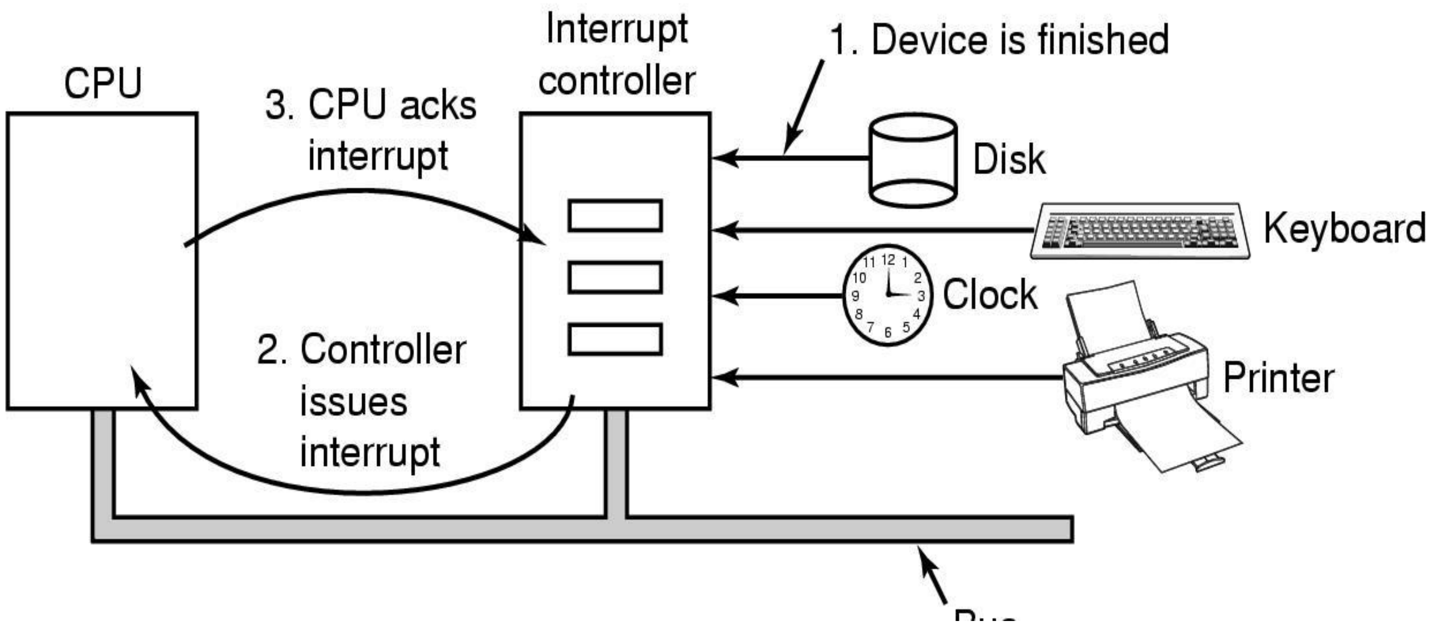
Interrupts

- Polling can happen in 3 instruction cycles
  - Read status, extract status bit, branch if status is shows done.
  - How to be more efficient if status is done infrequently?
- CPU **Interrupt-request line** triggered by I/O device
  - Checked by processor after each instruction
- **Interrupt handler** receives interrupts
  - **Maskable** to ignore or delay some interrupts

Interrupt Service Routine



Interrupt Driven I/O

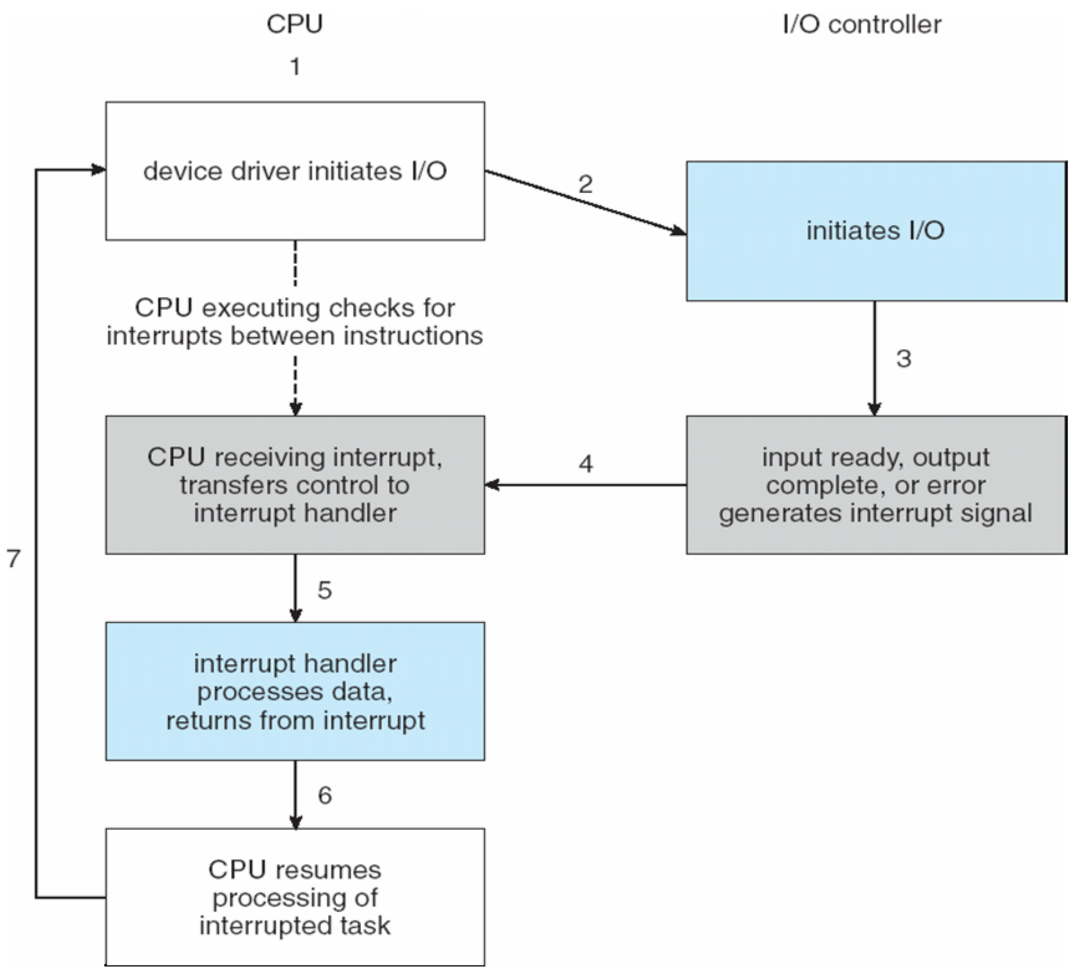


- ❖ I/O module gets data from peripheral while CPU continues other work.
- ❖ I/O module interrupts CPU when data is ready.
- ❖ CPU requests data
- ❖ I/O module transfers data

Interrupt Driven I/O

- I/O device issues an interrupt to indicate that it needs attention of CPU
- Interrupts are special signals initialed by I/O devices to catch the attention of the processor.
- Overcomes CPU waiting
- No repeated CPU checking of device
- An I/O interrupt is asynchronous w.r.t. instruction execution
- Is not associated with any instruction so doesn’t prevent any instruction from completing

Interrupt-Driven I/O Cycle



Interrupt Driven I/O

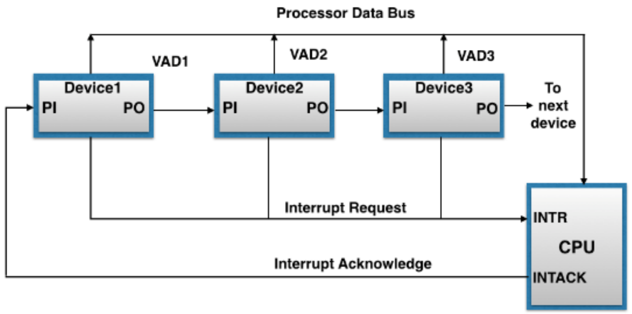
- Advantages
  - Relieves the processor from having to continuously poll for an I/O event; user program progress is only suspended during the actual transfer of I/O data to/from user memory space
- Disadvantages
  - Special hardware is needed to indicate the I/O device causing the interrupt and to save the necessary information prior to servicing the interrupt and to resume normal processing after servicing the interrupt

Challenges in Interrupt Driven I/O

- How do you identify the module issuing the interrupt?
  - Need a way to identify the device generating the interrupt
- How do you deal with multiple interrupts?
  - Can have different urgencies (so need a way to prioritize them)

Identifying Interrupting Module

- CPU asks each module in turn (Slow) Daisy Chain or Hardware poll
  - Interrupt Acknowledge sent down a chain
  - Module responsible places vector on bus
  - CPU uses vector to identify handler routine



- Vectored Interrupt
  - **Interrupt vector** to dispatch interrupt to correct handler

Ex: Intel Pentium Processor Event-Vector Table

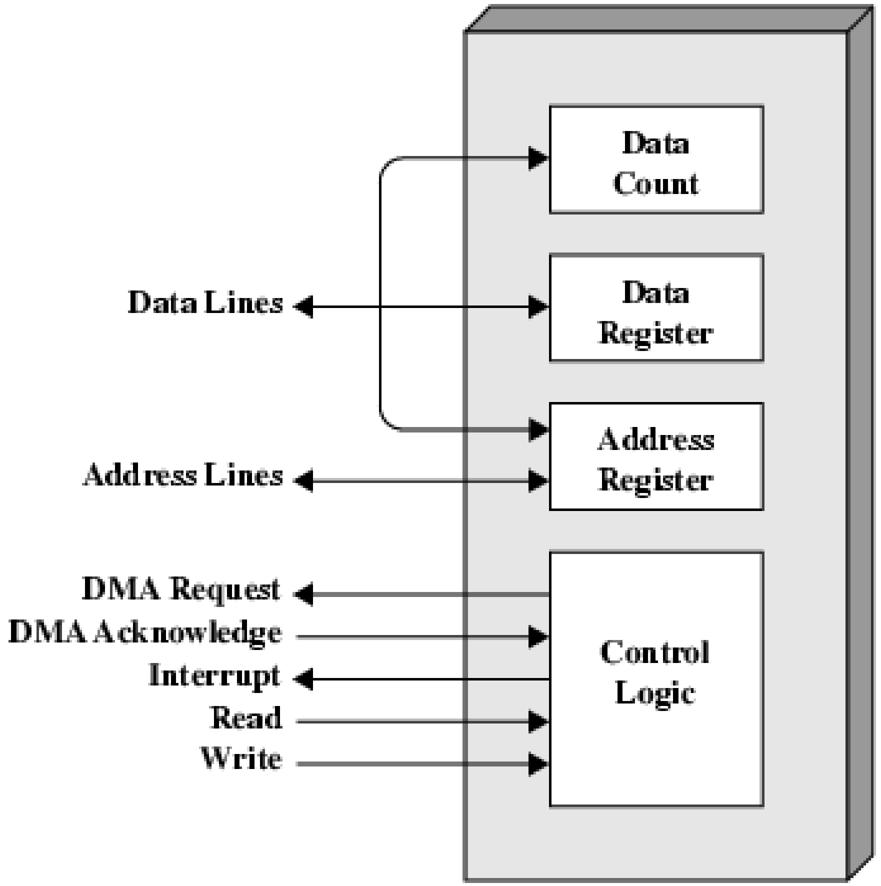
vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Direct Memory Access

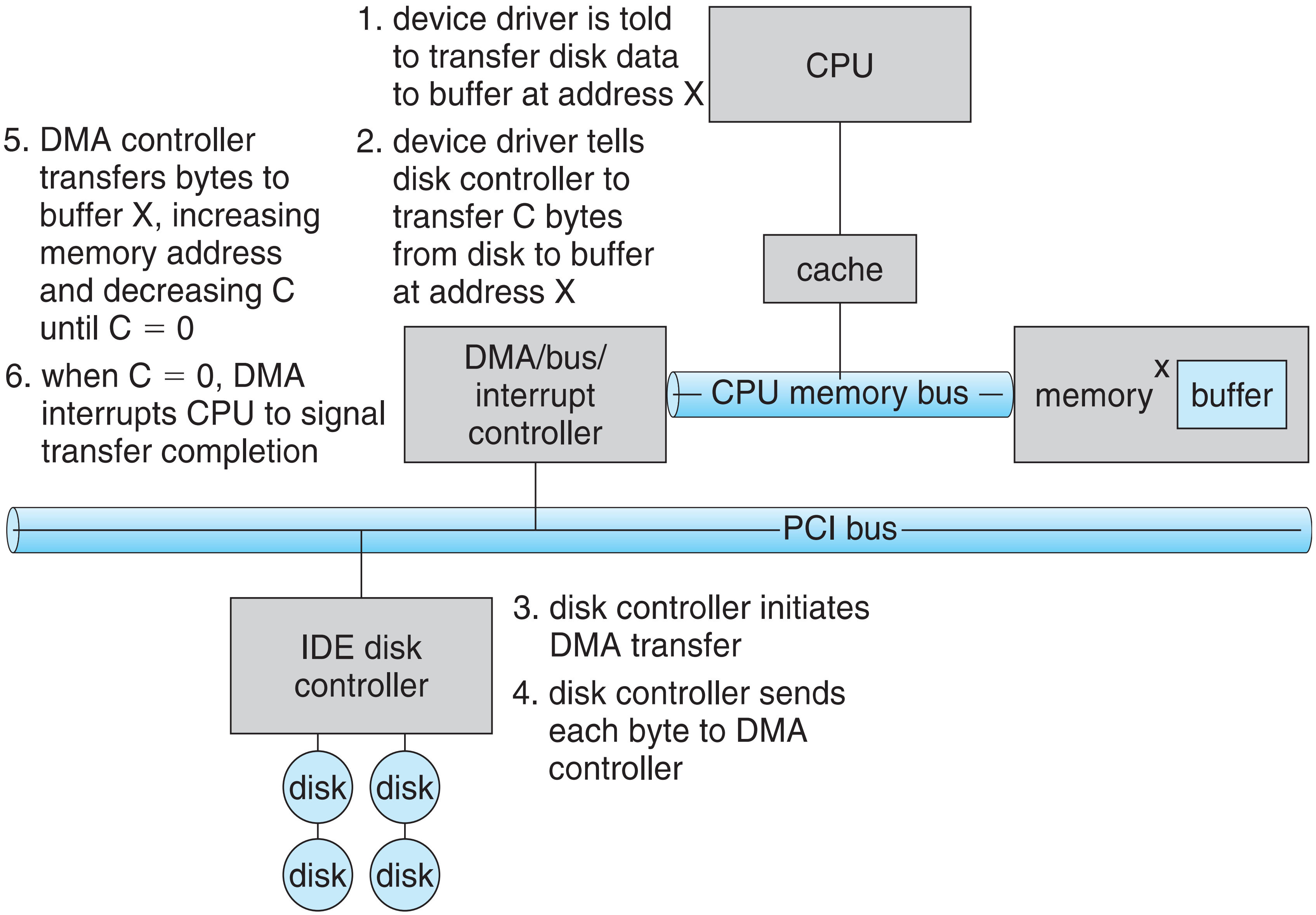
- Interrupt driven and programmed I/O require active CPU intervention
- For high-bandwidth devices (like disks) interrupt-driven I/O would consume a lot of processor cycles
- Bypasses CPU to transfer data directly between I/O device and memory
- OS writes DMA command block into memory
  - Source and destination addresses, Read or write mode
  - Count of bytes, Writes location of command block to DMA controller
- DMA is an additional module (hardware) on bus
- DMA controller takes over from CPU for I/O operations

DMA Module Diagram

- ❖ CPU tells DMA controller:-
  - ❖ Read/Write
  - ❖ Device address
  - ❖ Starting address of memory block for data
  - ❖ Amount of data to be transferred
- ❖ CPU carries on with other work
- ❖ DMA controller deals with transfer
- ❖ DMA controller sends interrupt when finished



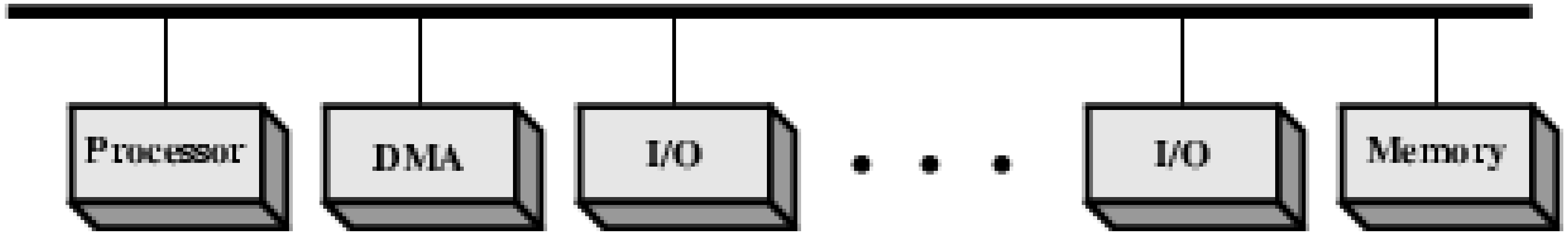
Disk to Memory Copy via DMA



Modes of DMA operation

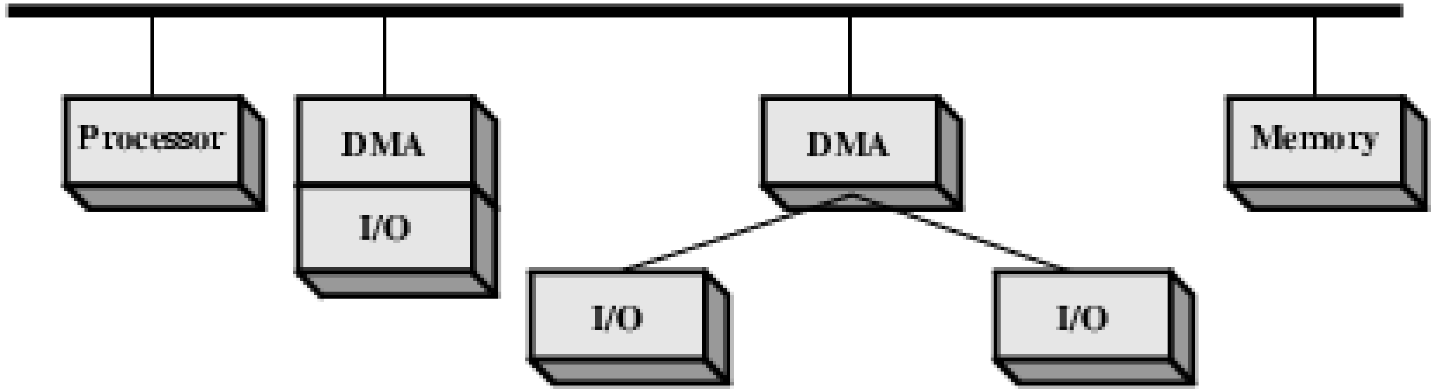
- **Cycle Stealing**
  - DMA controller acquires control of bus
  - Transfers a single word and releases the bus
  - The CPU is slowed down due to bus contention
  - Responsive but not very efficient
- **Burst Mode**
  - DMA Controller acquires control of bus
  - Transfers all the data and then only releases the bus
  - The CPU is suspended or works with cache
  - Efficient but interrupts may not be serviced in a timely way

DMA Configurations



- Single Bus, Detached DMA controller
- Each transfer uses bus twice
  - I/O to DMA then DMA to memory
- CPU is suspended twice

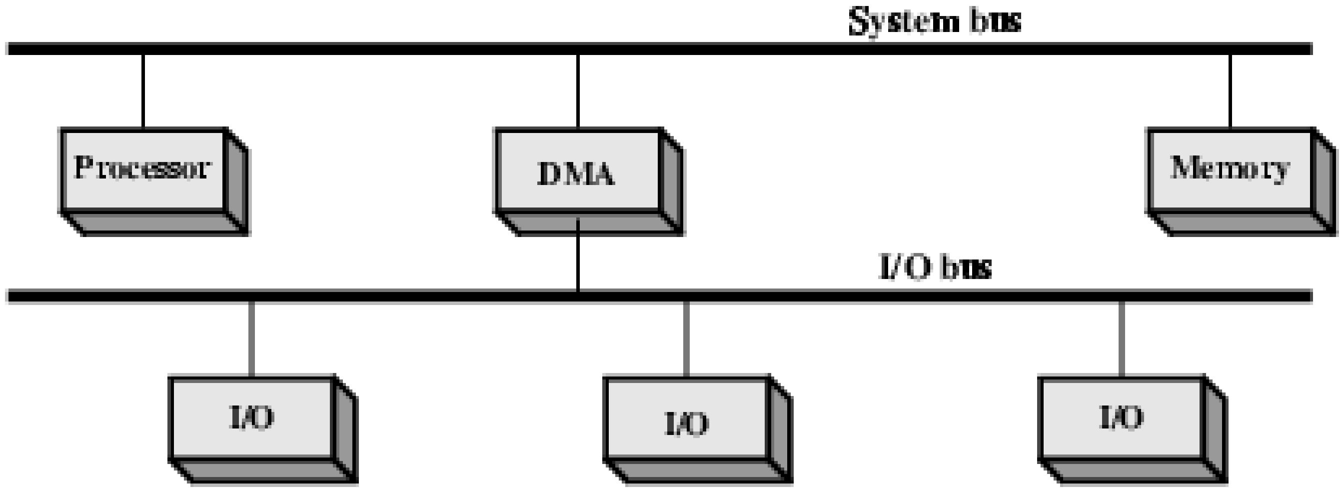
DMA Configurations



(b) Single-bus, Integrated DMA-I/O

- ❖ Single Bus, Integrated DMA controller
- ❖ Controller may support >1 device
- ❖ Each transfer uses bus once
  - ❖ DMA to memory
- ❖ CPU is suspended once

DMA Configurations



(c) I/O bus

- Separate I/O Bus
- Bus supports all DMA enabled devices
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

Thank You