

**CPS 104**  
**Computer Organization and Programming**  
**Lecture 22: I/O**

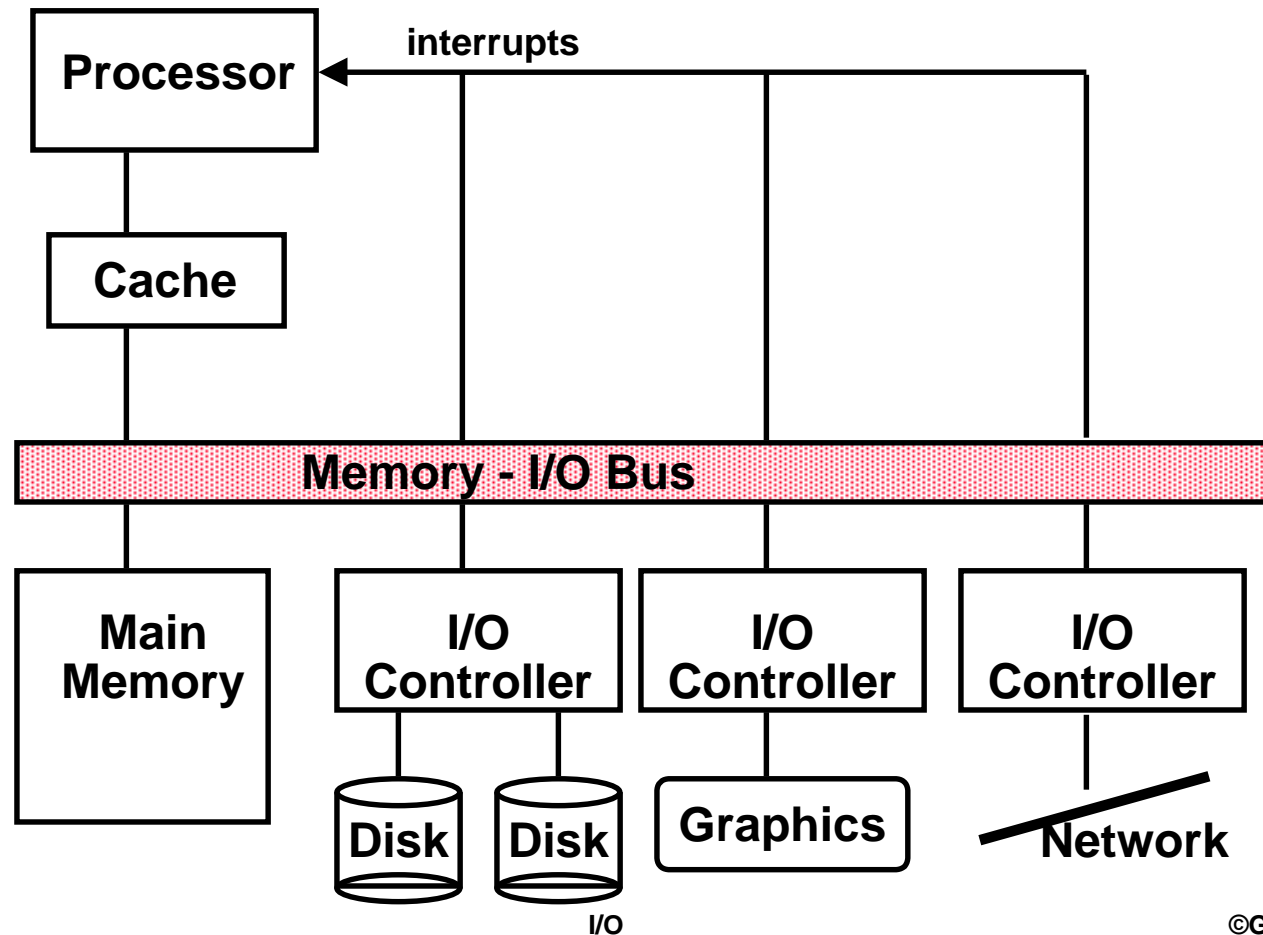
**Nov. 10, 1999**

**Dietolf (Dee) Ramm**

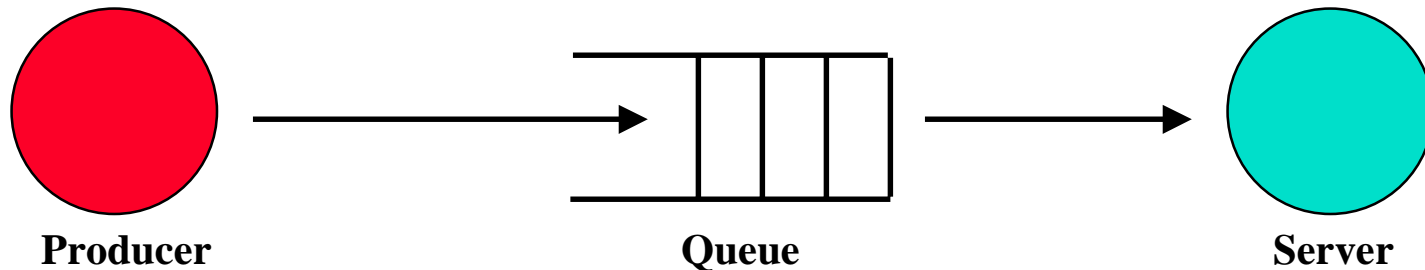
**<http://www.cs.duke.edu/~dr/cps104.html>**

# I/O System Design Issues

- \* Performance
- \* Expandability
- \* Resilience in the face of failure



# Producer-Server Model



- **Throughput:**

- \* The number of tasks completed by the server in unit time
- \* In order to get the highest possible throughput:
  - The server should never be idle
  - The queue should never be empty

- **Response time:**

- \* Begins when a task is placed in the queue
- \* Ends when it is completed by the server
- \* In order to minimize the response time:
  - The queue should be empty
  - The server will be idle

## I/O Device Examples

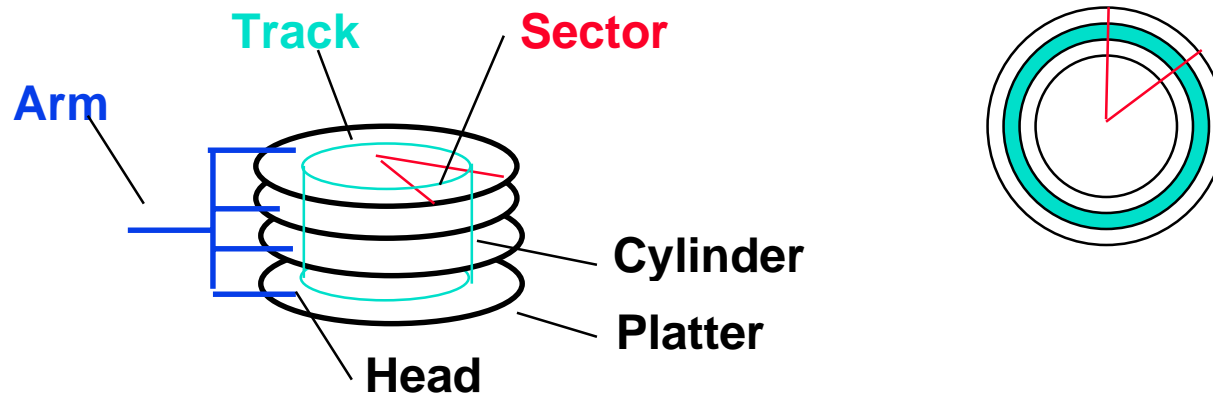
<b>Device</b>	<b>Behavior</b>	<b>Partner</b>	<b>Data Rate (KB/sec)</b>
<b>Keyboard</b>	<b>Input</b>	<b>Human</b>	<b>0.01</b>
<b>Mouse</b>	<b>Input</b>	<b>Human</b>	<b>0.02</b>
<b>Line Printer</b>	<b>Output</b>	<b>Human</b>	<b>1.00</b>
<b>Laser Printer</b>	<b>Output</b>	<b>Human</b>	<b>100.00</b>
<b>Graphics Displa</b>	<b>Output</b>	<b>Human</b>	<b>30,000.00</b>
<b>Network-LAN</b>	<b>Input/Output</b>	<b>Machine</b>	<b>10,000.00</b>
<b>Floppy disk</b>	<b>Storage</b>	<b>Machine</b>	<b>50.00</b>
<b>Optical Disk</b>	<b>Storage</b>	<b>Machine</b>	<b>500.00</b>
<b>Magnetic Disk</b>	<b>Storage</b>	<b>Machine</b>	<b>5,000.00</b>

# Types of Storage Devices

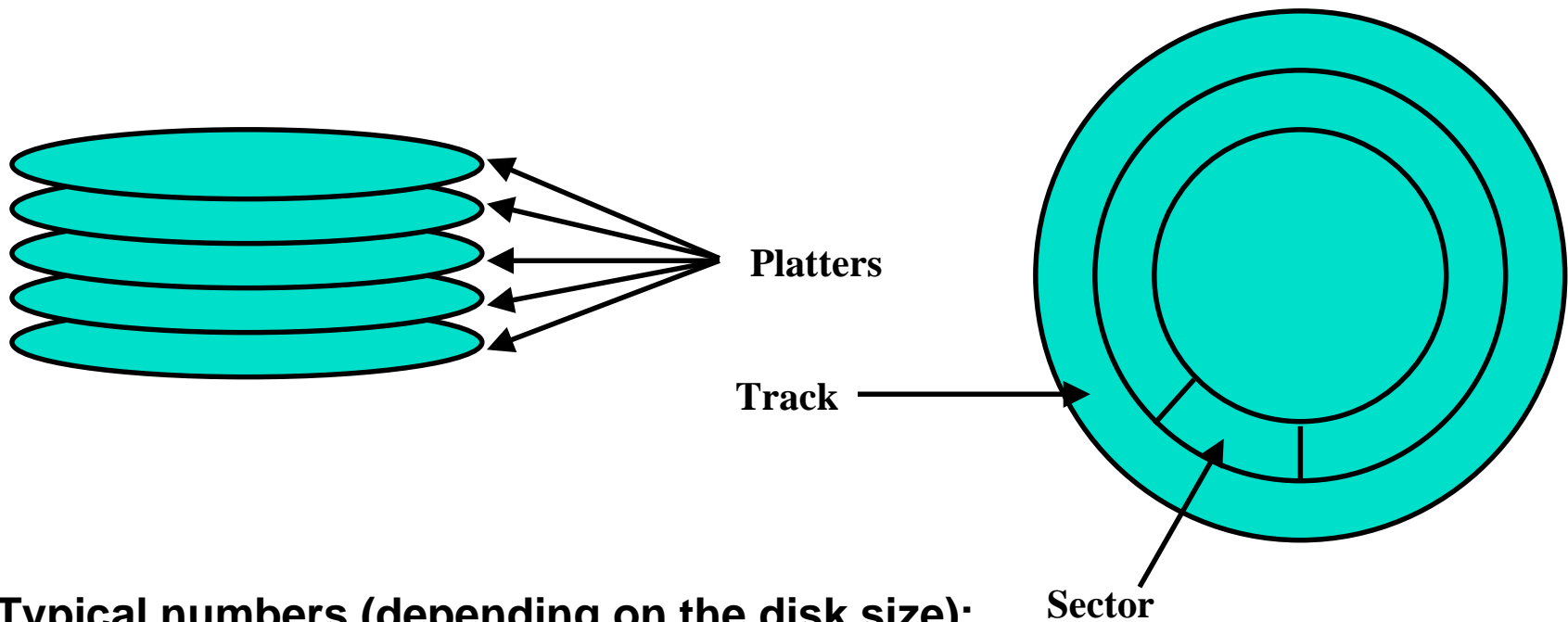
- **Magnetic Disks**
- **Magnetic Tapes**
- **CD ROM**
- **Juke Box (automated tape library, robots)**

# Magnetic Disks

- Long term nonvolatile storage
- Another slower, less expensive level of memory hierarchy



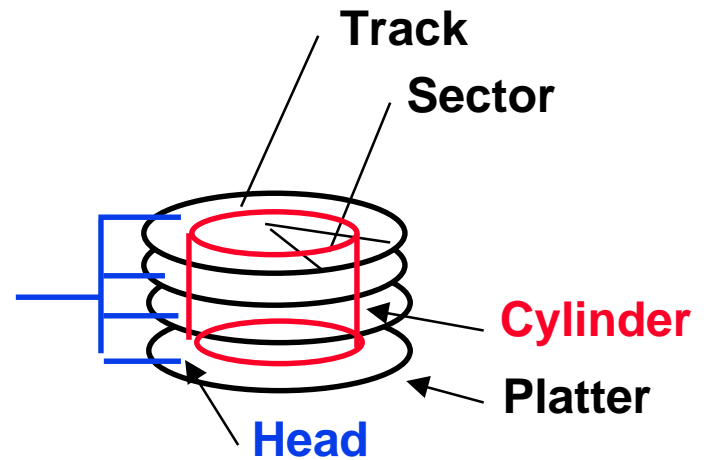
# Organization of a Hard Magnetic Disk



- **Typical numbers (depending on the disk size):**
  - \* 500 to 2,000 tracks per surface
  - \* 32 to 128 sectors per track
  - A sector is the smallest unit that can be read or written
- **Traditionally all tracks have the same number of sectors:**
  - \* **Constant bit density:** record more sectors on the outer tracks
  - \* **Recently relaxed:** constant bit size, speed varies with track location

# Magnetic Disk Characteristic

- **Cylinder:** all the tracks under the heads for a given arm position (tracks with same radius)

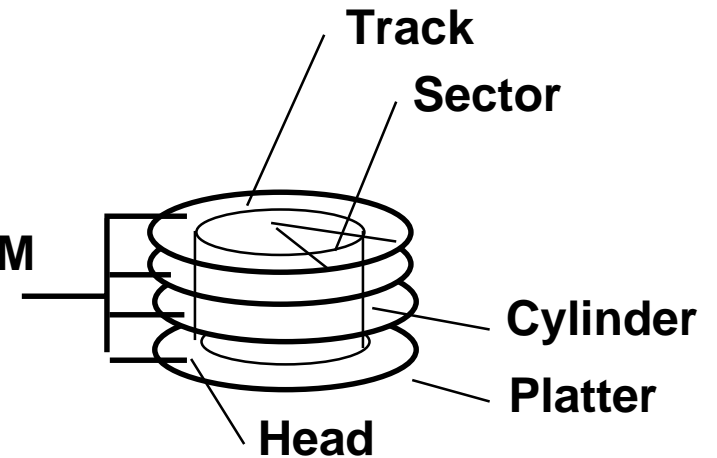


- **Read/write data is a three-stage process:**
  - \* **Seek time:** position the arm over the proper track
  - \* **Rotational latency:** wait for the desired sector to rotate under the read/write head
  - \* **Transfer time:** transfer a block of bits (sector) under the read-write head
- **Average seek time as reported by the industry:**
  - \* Typically in the range of 8 ms to 12 ms
  - \*  $(\text{Sum of the time for all possible seek}) / (\text{total \# of possible seeks})$
- **Due to locality of disk reference, actual average seek time may:**
  - \* Only be 25% to 33% of the advertised number

# Typical Numbers of a Magnetic Disk

- **Rotational Latency:**

- \* **Most disks rotate at 3,600 to 10000 RPM**
- \* **Approximately 16 ms to 3.5 ms per revolution, respectively**
- \* **An average latency to the desired information is halfway around the disk: 8 ms at 3600 RPM, 4 ms at 7200 RPM**



- **Transfer Time is a function of :**

- \* **Transfer size (usually a sector): 1 KB / sector**
- \* **Rotation speed: 3600 RPM to 7200 RPM**
- \* **Recording density: bits per inch on a track**
- \* **Diameter typical diameter ranges from 2.5 to 5.25 in**
- \* **Typical values: 2 to 12 MB per second**

# Disk Access

- Access time =

**queue + seek + rotational + transfer + overhead**

- Seek time

- \* move arm over track
- \* average is confusing (startup, slowdown, locality of accesses)

- Rotational latency

- \* wait for sector to rotate under head
- \* average =  $0.5 / (3600 \text{ RPM}) = 8.3\text{ms}$

- Transfer Time

- \*  $f(\text{size, BW bytes/sec})$

# Disk Access Time Example

- **Disk Parameters:**
  - \* Transfer size is 8K bytes
  - \* Advertised average seek is 12 ms
  - \* Disk spins at 7200 RPM
  - \* Transfer rate is 4 MB/sec
- **Controller overhead is 2 ms**
- **Assume that disk is idle so no queuing delay**
- **What is Average Disk Access Time for a Sector?**
  - \* Ave seek + ave rot delay + transfer time + controller overhead
  - \*  $12 \text{ ms} + 0.5 / (7200 \text{ RPM} / 60) + 8 \text{ KB} / 4 \text{ MB/s} + 2 \text{ ms}$
  - \*  $12 + 4.15 + 2 + 2 = 20 \text{ ms}$
- **Advertised seek time assumes no locality: typically 1/4 to 1/3 advertised seek time: 20 ms => 12 ms**

# DRAM as Disk

- **Solid state disk, Expanded Storage, NVRAM**
- **Disk is slow, DRAM is fast => replace Disk with battery backed DRAM**
- **BUT, Disk is cheap, much cheaper than DRAM**
- **Network Memory**
  - \* **fast networks (e.g., Myrinet)**
  - \* **use DRAM of other workstations as backing store**
  - \* **Trapeze/GMS project here**

# Tape vs. Disk

- Longitudinal tape uses same technology as hard disk; tracks its density improvements
- Inherent cost-performance based on geometry:

fixed rotating platters with gaps  
(random access, limited area, fixed media)

**Vs.**

removable long strips wound on spool  
(sequential access, "unlimited" length)

- New technology trend:  
Helical Scan (VCR, Camcoder, DAT)  
Spins head at angle to tape to improve density

# Storage costs

Media	Capacity	Cost	\$/byte
Disk	4-20GB	\$100-\$1000	$2 \times 10^{-8}$
Tape	2-70GB	\$4-\$100	$1 \times 10^{-9}$
CDR	0.75GB	\$1	$1.4 \times 10^{-9}$
Flash EPROM	2MB	\$10	$5 \times 10^{-5}$
Paper	10KB	\$0.01	$1 \times 10^{-6}$

- **Digital storage cost has (will have) profound effect on the way we store and access Information!**
- **Digital storage (and the Web) opens-up new exciting possibilities for access to information culture & knowledge.**
- **Digital storage introduces whole new set of problems to long term information storage.**

## CDR vs. Tape

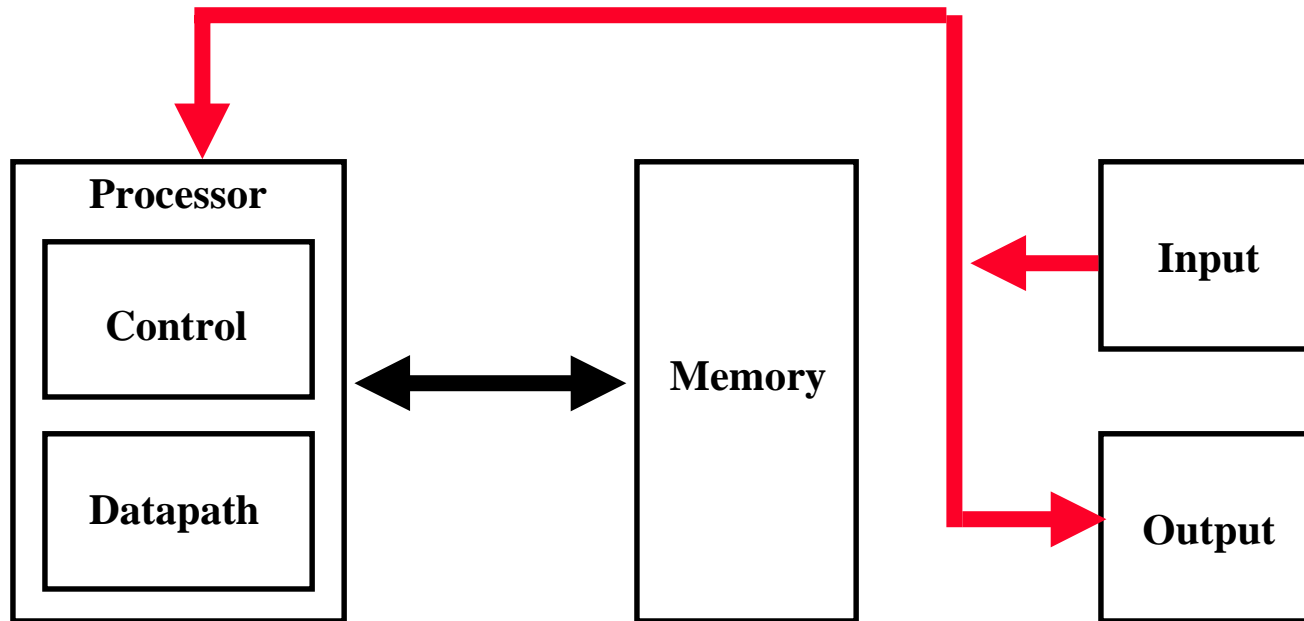
	CDR	Helical Scan Tape
Type	5.25"	8mm
Capacity	0.75 GB	5-12 GB
Media Cost	\$1	\$8
Drive Cost	\$300	\$600
Access	Write Once	Read/Write
Robot Time	10 - 20 s	10 - 20 s

Media cost ratio CDR vs. helical tape  
= 2 : 1

## Current Drawbacks to Tape

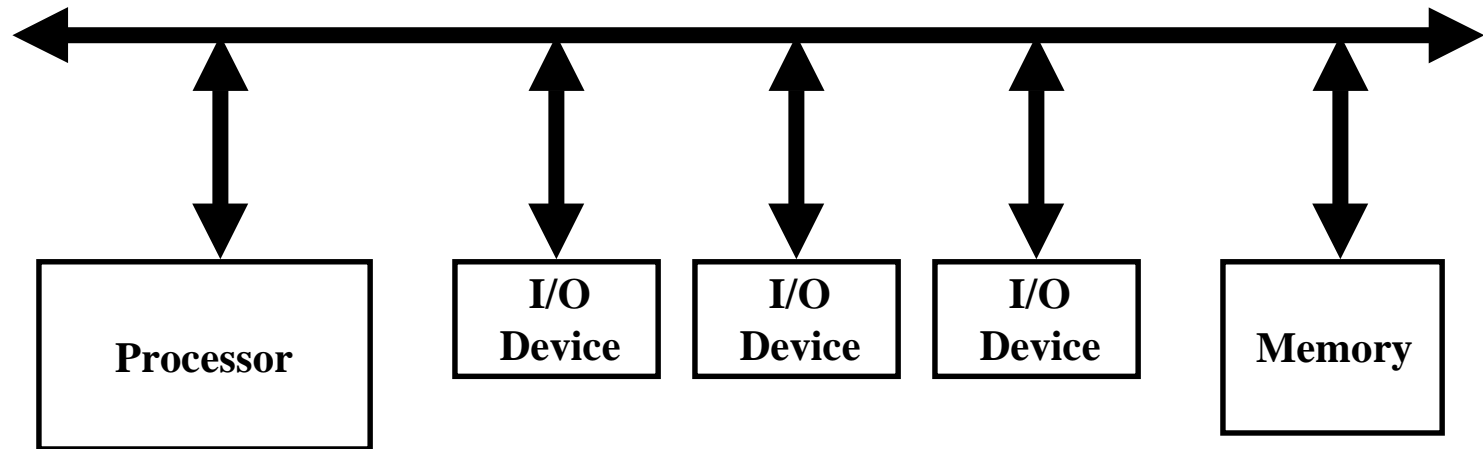
- **Tape wear out:**
  - Helical 100s of passes to 1000s for longitudinal
- **Head wear out:**
  - 2000 hours for helical
- **Both must be accounted for in economic / reliability model**
- **Long rewind, eject, load, spin-up times;**

# Buses: Connecting I/O to Processor and Memory



- A bus is a shared communication link
- It uses one set of wires to connect multiple subsystems

# Advantages of Buses



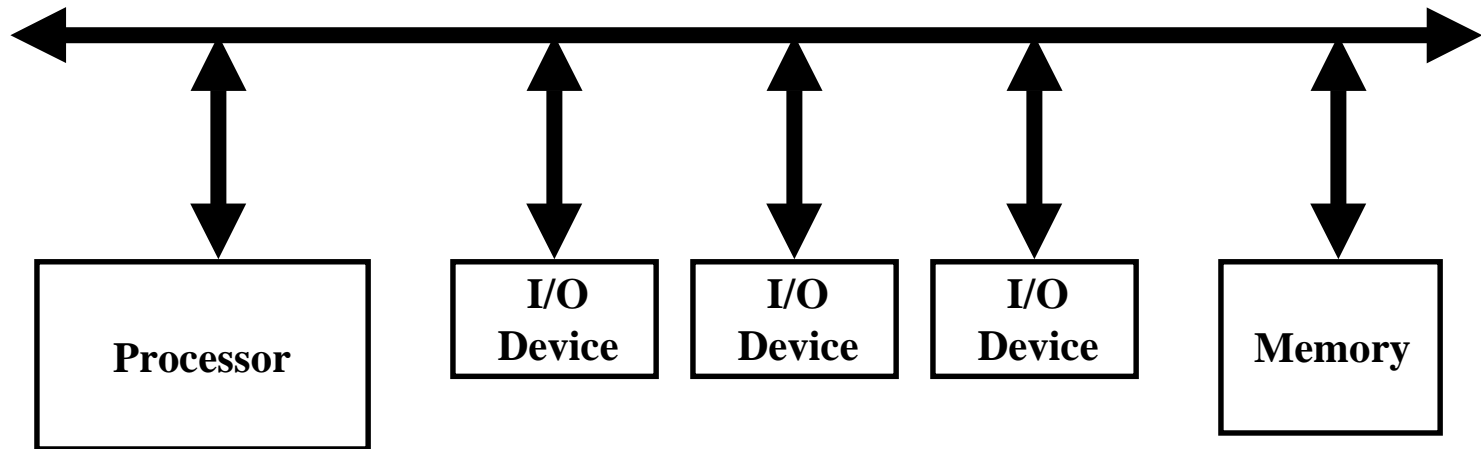
- **Versatility:**

- \* New devices can be added easily
- \* Peripherals can be moved between computer systems that use the same bus standard

- **Low Cost:**

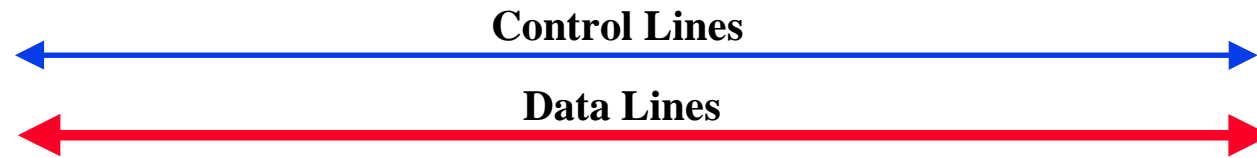
- \* A single set of wires is shared in multiple ways

# Disadvantages of Buses



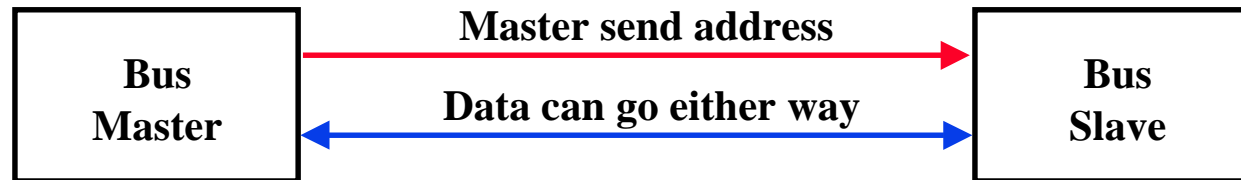
- **The bus creates a communication bottleneck**
  - \* **Bus bandwidth can limit the maximum I/O throughput**
- **The maximum bus speed is largely limited by:**
  - \* **The length of the bus**
  - \* **The number of devices on the bus**
  - \* **The need to support a range of devices with:**
    - **Widely varying latencies**
    - **Widely varying data transfer rates**

# The General Organization of a Bus



- **Control lines:**
  - \* Signal requests and acknowledgments
  - \* Indicate what type of information is on the data lines
- **Data lines** carry information between the source and the destination:
  - \* Data and Addresses
  - \* Complex commands

# Master versus Slave

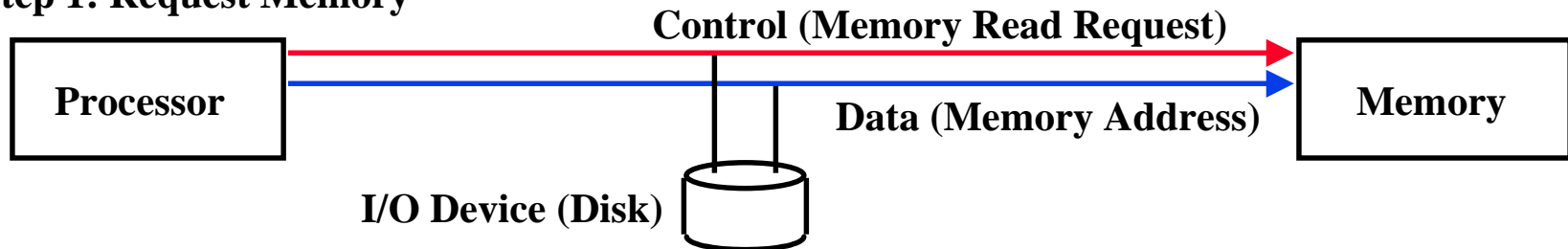


- **A bus transaction includes two parts:**
  - \* **Sending the address**
  - \* **Receiving or sending the data**
- **Master is the one who starts the bus transaction by:**
  - \* **Sending the address**
- **Slave is the one who responds to the address by:**
  - \* **Sending data to the master if the master ask for data**
  - \* **Receiving data from the master if the master wants to send data**

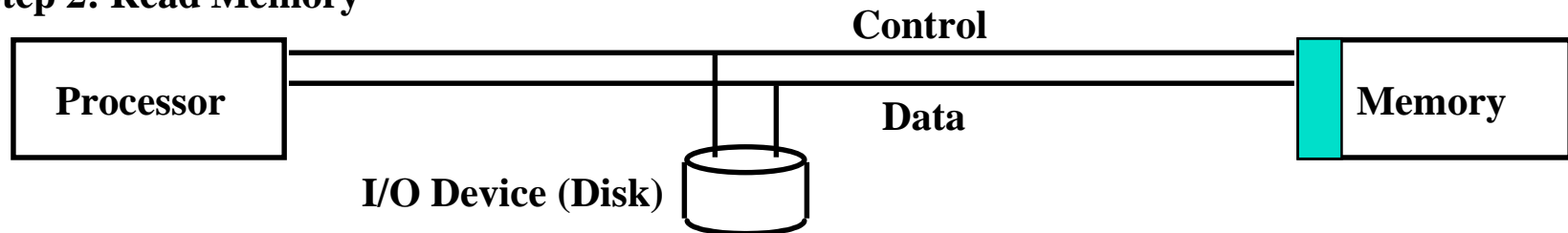
# Output Operation

- Output is defined as the **Processor** sending data **to** the I/O device:

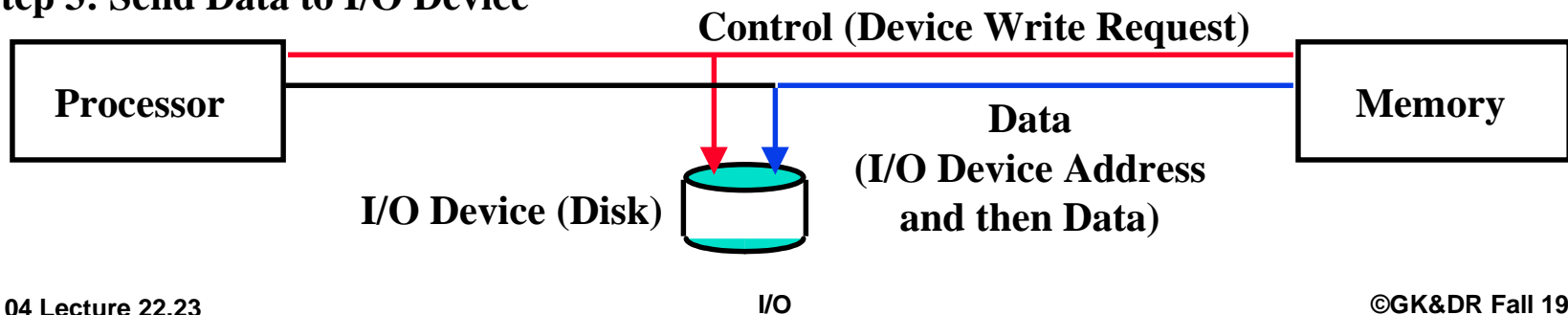
## Step 1: Request Memory



## Step 2: Read Memory



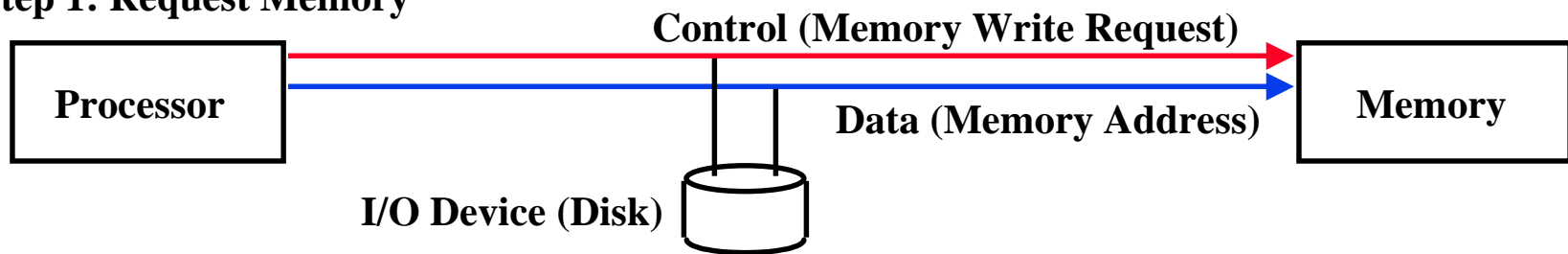
## Step 3: Send Data to I/O Device



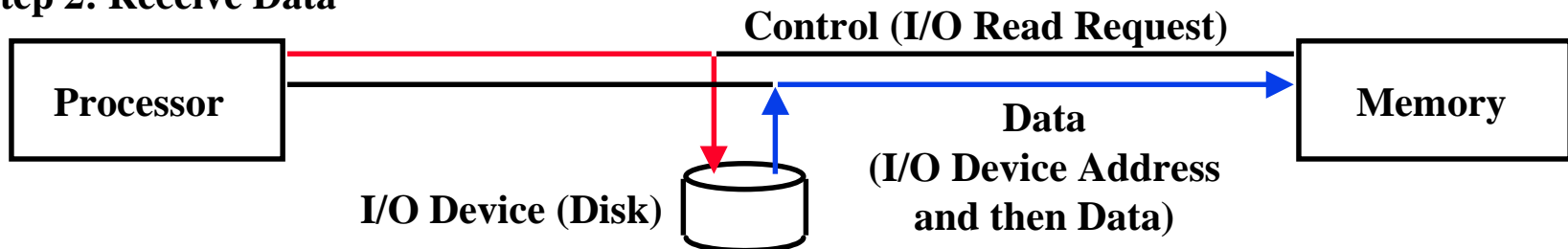
# Input Operation

- Input is defined as the **Processor** receiving data **from** the I/O device:

## Step 1: Request Memory



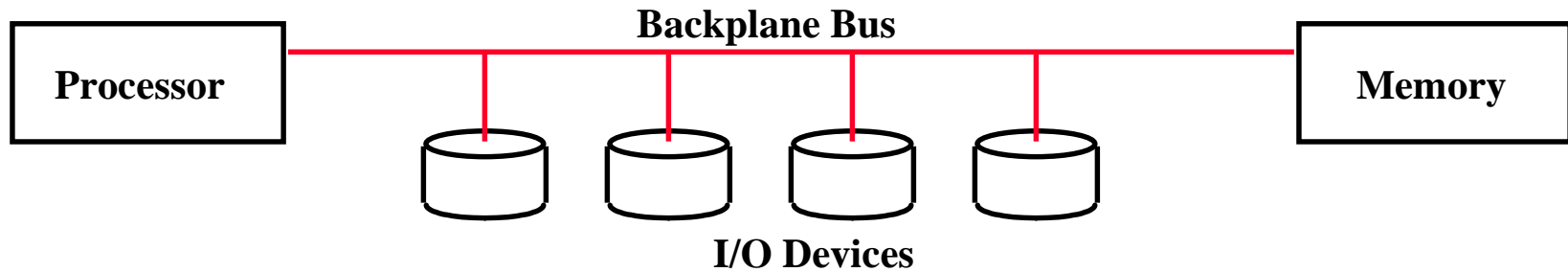
## Step 2: Receive Data



# Types of Buses

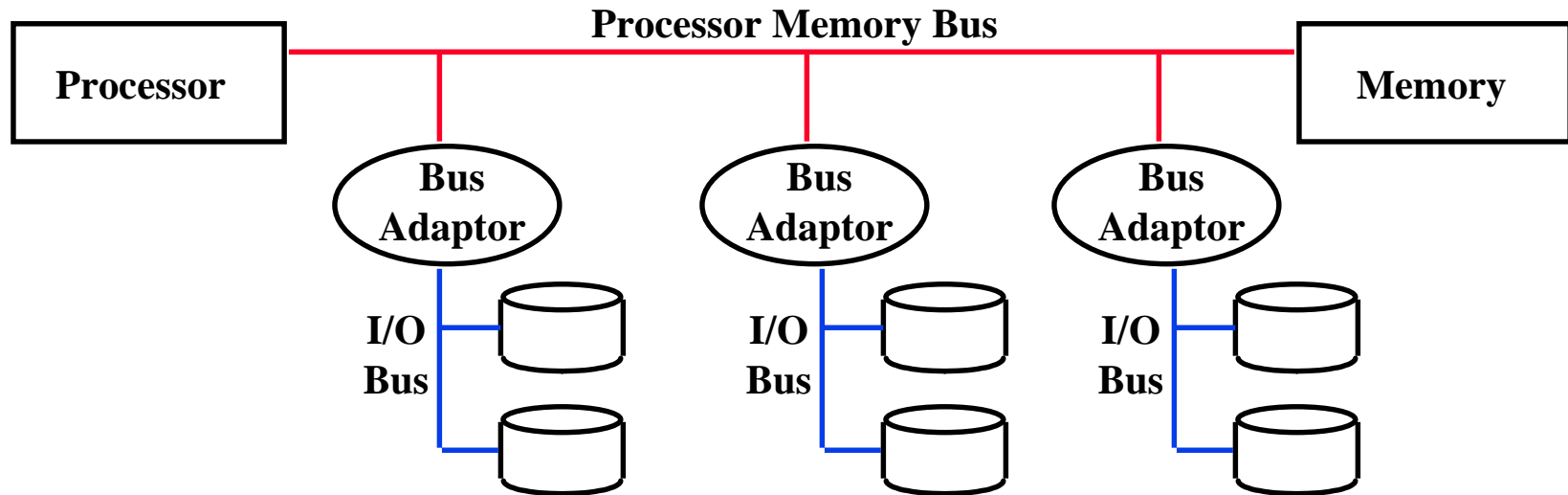
- **Processor-Memory Bus (design specific)**
  - \* Short and high speed
  - \* Only need to match the memory system
    - Maximize memory-to-processor bandwidth
  - \* Connects directly to the processor
- **External I/O Bus (industry standard)**
  - \* Usually is lengthy and slower
  - \* Need to match a wide range of I/O devices
  - \* Connects to the processor-memory bus or backplane bus
- **Backplane Bus (industry standard)**
  - \* Backplane: an interconnection structure within the chassis
  - \* Allow processors, memory, and I/O devices to coexist
  - \* Cost advantage: one single bus for all components
- **Bit-Serial Buses (New trend: SBI, Firewire .. )**
  - \* Use High speed unidirectional point-to-point communication
  - \* Use differential signaling
  - \* Use Distributed control

# A Computer System with One Bus: Backplane Bus



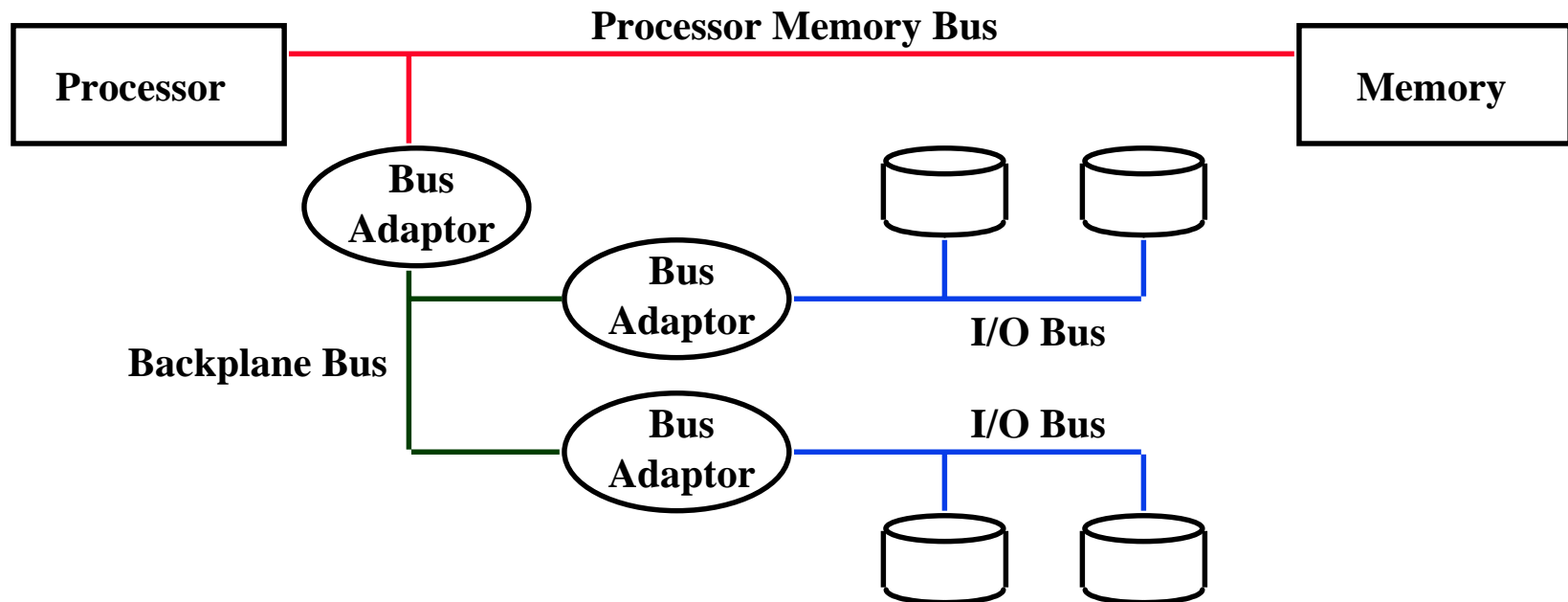
- A single bus (the backplane bus) is used for:
  - \* Processor to memory communication
  - \* Communication between I/O devices and memory
- Advantages: Simple and low cost
- Disadvantages: slow and the bus can become a major bottleneck
- Example: IBM PC

# A Two-Bus System



- **I/O buses tap into the processor-memory bus via bus adaptors:**
  - \* **Processor-memory bus:** mainly for processor-memory traffic
  - \* **I/O buses:** provide expansion slots for I/O devices
- **Example: Apple Macintosh-II**
  - \* **NuBus:** Processor, memory, and a few selected I/O devices
  - \* **SCCI Bus:** the rest of the I/O devices

# A Three-Bus System



- **A small number of backplane buses tap into the processor-memory bus**
  - \* **Processor-memory bus is used for processor memory traffic**
  - \* **I/O buses are connected to the backplane bus**
- **Advantage: loading on the processor bus is greatly reduced**

# Synchronous and Asynchronous Bus

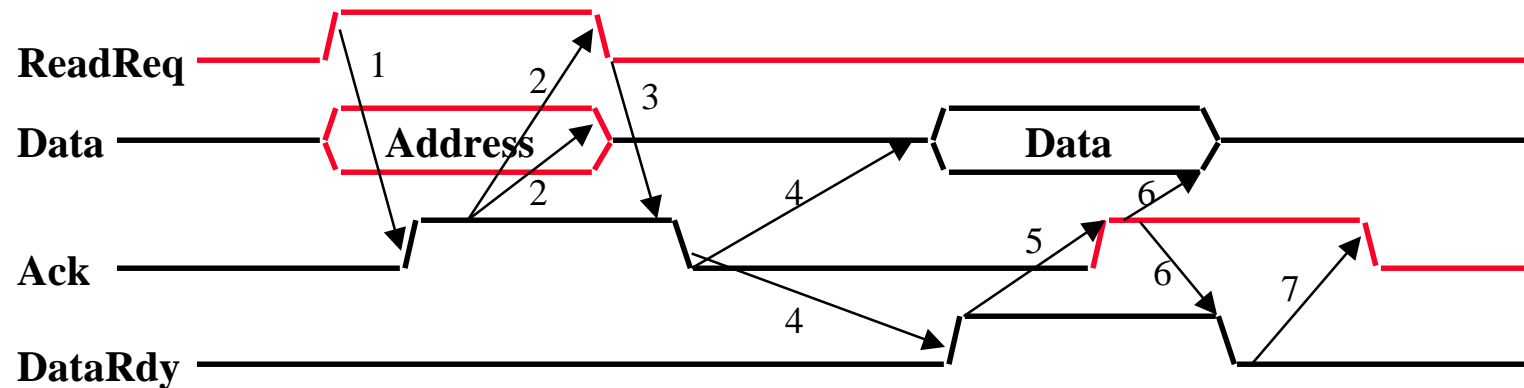
- **Synchronous Bus:**

- \* Includes a clock in the control lines
- \* A fixed protocol for communication that is relative to the clock
- \* Advantage: involves very little logic and can run very fast
- \* Disadvantages:
  - Every device on the bus must run at the same clock rate
  - To avoid clock skew, bus must be short if it is fast.

- **Asynchronous Bus:**

- \* It is not clocked
- \* It can accommodate a wide range of devices
- \* It can be lengthened without worrying about clock skew
- \* It requires a handshaking protocol

# A Handshaking Protocol



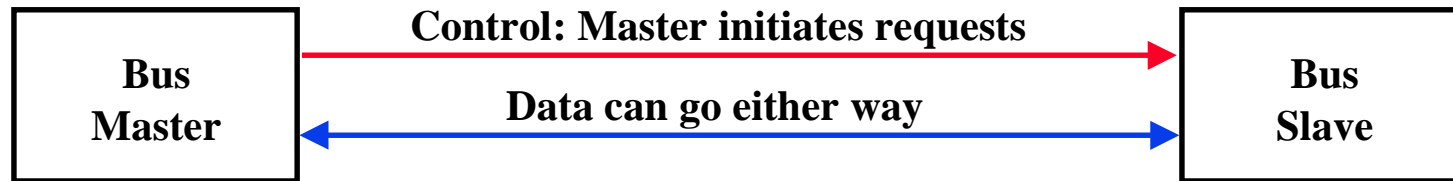
- **Three control lines**

- \* **ReadReq:** indicate a read request for memory  
Address is put on the data lines at the same line
- \* **DataRdy:** indicate the data word is now ready on the data lines  
Data is put on the data lines at the same time
- \* **Ack:** acknowledge the ReadReq or the DataRdy of the other party

# Increasing the Bus Bandwidth

- **Separate versus multiplexed address and data lines:**
  - \* **Address and data can be transmitted in one bus cycle if separate address and data lines are available**
  - \* **Cost: (a) more bus lines, (b) increased complexity**
- **Data bus width:**
  - \* **By increasing the width of the data bus, transfers of multiple words require fewer bus cycles**
  - \* **Example: SPARCstation 20's memory bus is 128 bit wide**
  - \* **Cost: more bus lines**
- **Block transfers:**
  - \* **Allow the bus to transfer multiple words in back-to-back bus cycles**
  - \* **Only one address needs to be sent at the beginning**
  - \* **The bus is not released until the last word is transferred**
  - \* **Cost: (a) increased complexity  
(b) decreased response time for request**

# Obtaining Access to the Bus

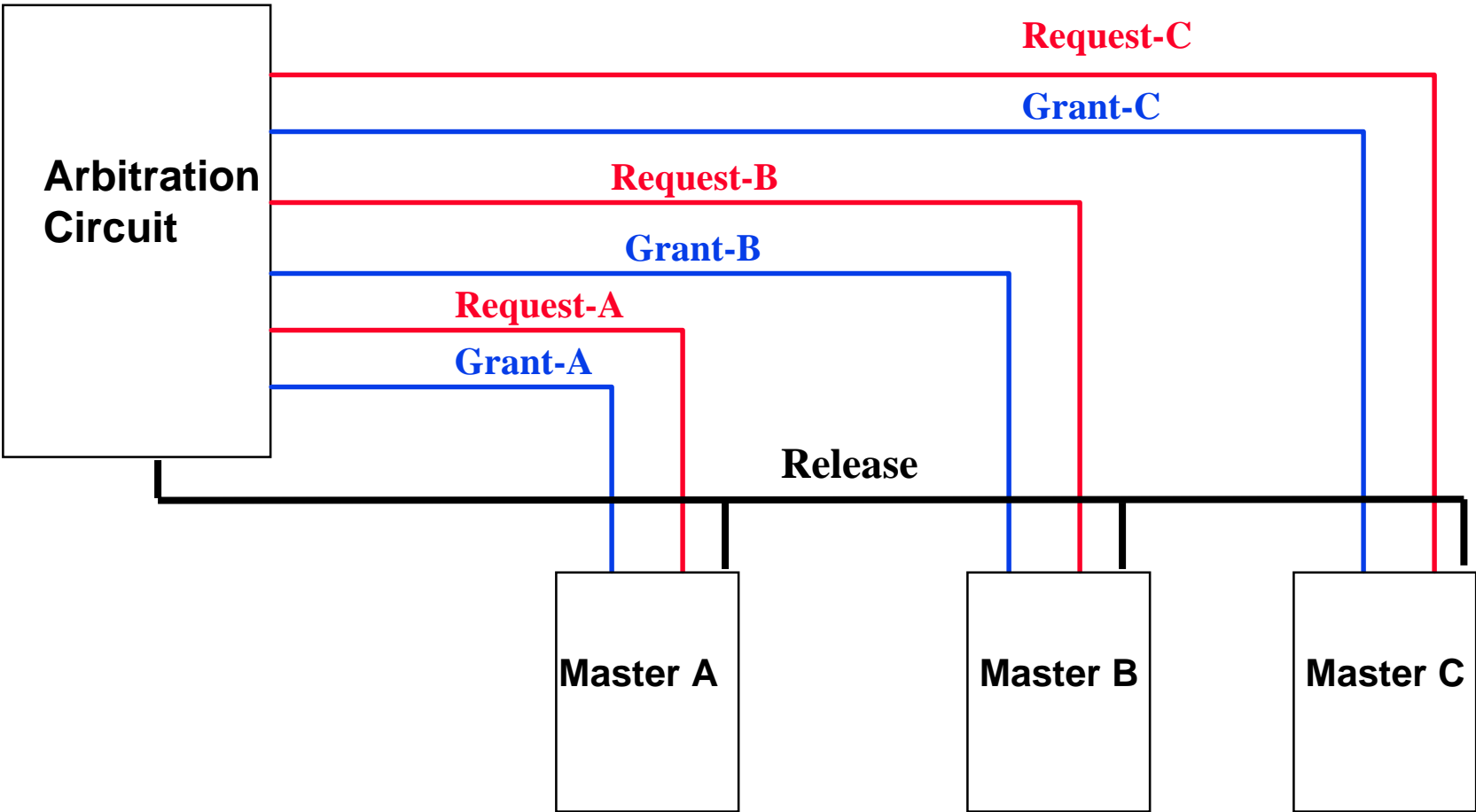


- One of the most important issues in bus design:
  - \* How is the bus reserved by a devices that wishes to use it?
- Chaos is avoided by a master-slave arrangement:
  - \* Only the bus master can control access to the bus:  
It initiates and controls all bus requests
  - \* A slave responds to read and write requests
- The simplest system:
  - \* Processor is the only bus master
  - \* All bus requests must be controlled by the processor
  - \* Major drawback: the processor is involved in every transaction

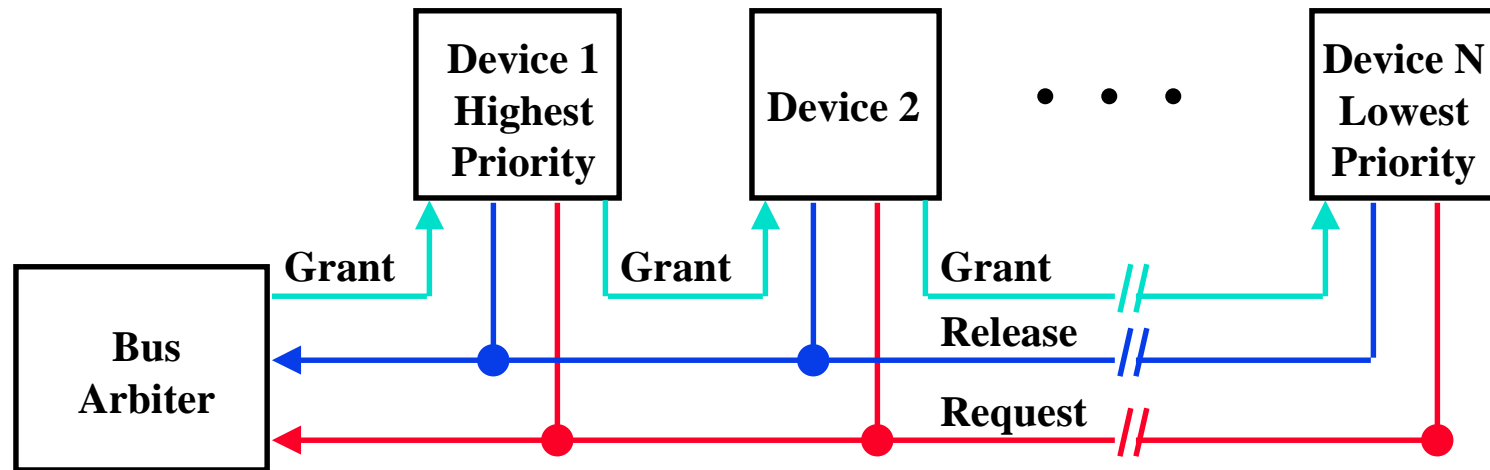
# Multiple Potential Bus Masters: the Need for Arbitration

- **Bus arbitration scheme:**
  - \* A bus master wanting to use the bus asserts the bus request
  - \* A bus master cannot use the bus until its request is granted
  - \* A bus master must signal to the arbiter after finish using the bus
- **Bus arbitration schemes usually try to balance two factors:**
  - \* **Bus priority:** the highest priority device should be serviced first
  - \* **Fairness:** Even the lowest priority device should never be completely locked out from the bus
- **Bus arbitration schemes can be divided into four broad classes:**
  - \* **Distributed arbitration by self-selection:** each device wanting the bus places a code indicating its identity on the bus.
  - \* **Distributed arbitration by collision detection:** Ethernet uses this.
  - \* **Daisy chain arbitration:** single device with all request lines.
  - \* **Centralized, parallel arbitration:** see next-next slide

# Centralized Bus Arbitration



# The Daisy Chain Bus Arbitration Scheme



- **Advantage: simple**
- **Disadvantages:**
  - \* **Cannot assure fairness:**  
A low-priority device may be locked out indefinitely
  - \* **The use of the daisy chain grant signal also limits the bus speed**

# Summary of Bus Options

<i>Option</i>	<i>High performance</i>	<i>Low cost</i>
<b>Bus width</b>	<b>Separate address &amp; data lines</b>	<b>Multiplex address &amp; data lines</b>
<b>Data width</b>	<b>Wider is faster (e.g., 64 bits)</b>	<b>Narrower is cheaper (e.g., 8 bits)</b>
<b>Transfer size</b>	<b>Multiple words has less bus overhead</b>	<b>Single-word transfer is simpler</b>
<b>Bus masters</b>	<b>Multiple (requires arbitration)</b>	<b>Single master (no arbitration)</b>
<b>Clocking</b>	<b>Synchronous</b>	<b>Asynchronous</b>

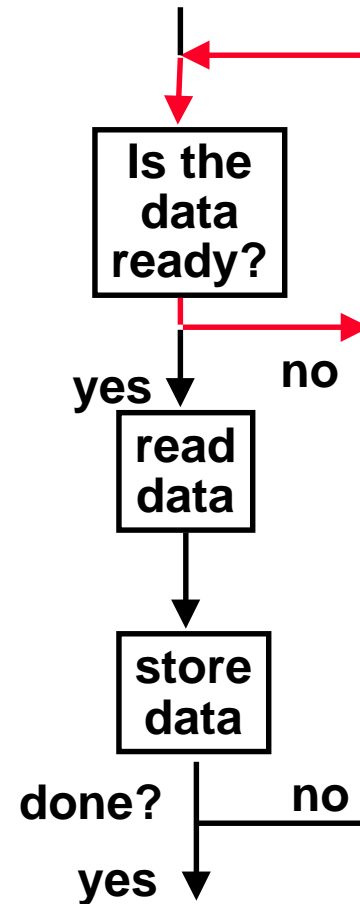
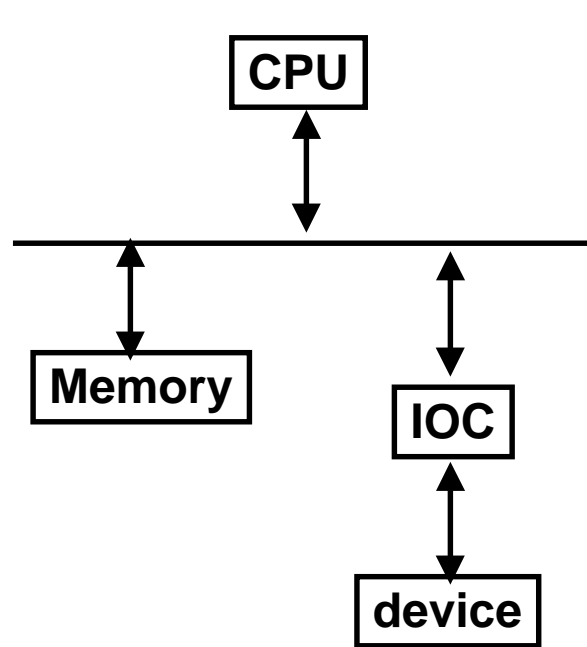
# 1993 Backplane/IO Bus Survey

<b>Bus</b>	<b>SBus</b>	<b>TurboChannel</b>	<b>MicroChannel</b>	<b>PCI</b>
<b>Originator</b>	<b>Sun</b>	<b>DEC</b>	<b>IBM</b>	<b>Intel</b>
<b>Clock Rate (MHz)</b>	<b>16-25</b>	<b>12.5-25</b>	<b>async</b>	<b>33-66</b>
<b>Addressing</b>	<b>Virtual</b>	<b>Physical</b>	<b>Physical</b>	<b>Physical</b>
<b>Data Sizes (bits)</b>	<b>8,16,32</b>	<b>8,16,24,32</b>	<b>8,16,24,32,64</b>	<b>8,16,24,32,64</b>
<b>Master</b>	<b>Multi</b>	<b>Single</b>	<b>Multi</b>	<b>Multi</b>
<b>Arbitration</b>	<b>Central</b>	<b>Central</b>	<b>Central</b>	<b>Central</b>
<b>32 bit read (MB/s)</b>	<b>33</b>	<b>25</b>	<b>20</b>	<b>33</b>
<b>Peak (MB/s)</b>	<b>89</b>	<b>84</b>	<b>75</b>	<b>111 (222)</b>
<b>Max Power (W)</b>	<b>16</b>	<b>26</b>	<b>13</b>	<b>25</b>

# OS and I/O Systems Communication Requirements

- **The Operating System must be able to prevent:**
  - \* **The user program from communicating with the I/O device directly**
- **If user programs could perform I/O directly:**
  - \* **Protection to the shared I/O resources could not be provided**
- **Three types of communication are required:**
  - \* **The OS must be able to give commands to the I/O devices**
  - \* **The I/O device must be able to notify the OS when the I/O device has completed an operation or has encountered an error**
  - \* **Data must be transferred between memory and an I/O device**

# Polling: Programmed I/O



**busy wait loop  
not an efficient  
way to use the CPU  
unless the device  
is very fast!**

**but checks for I/O  
completion can be  
dispersed among  
computation  
intensive code**

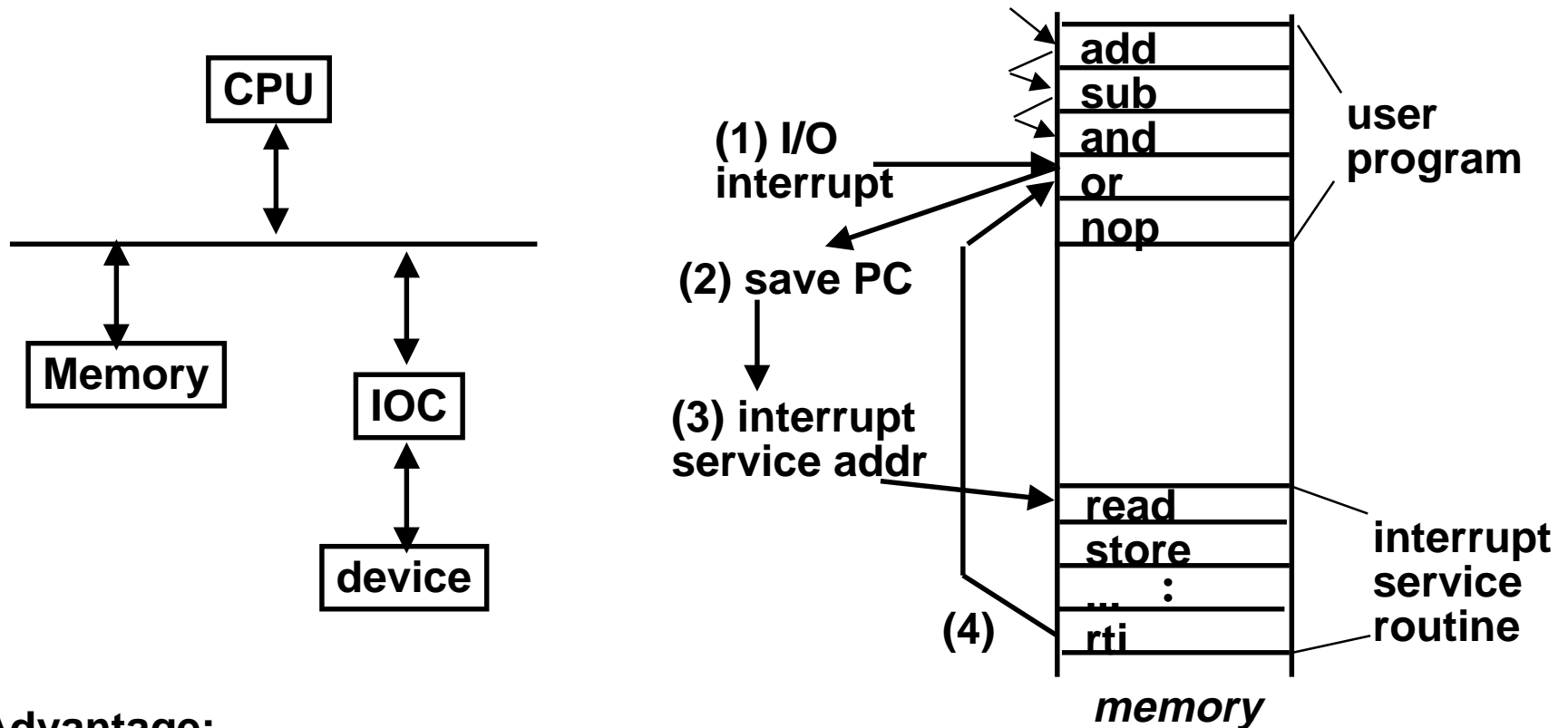
- **Advantage:**

- \* **Simple: the processor is totally in control and does all the work**

- **Disadvantage:**

- \* **Polling overhead can consume a lot of CPU time**

# Interrupt Driven Data Transfer



- **Advantage:**

- \* User program progress is only halted during actual transfer

- **Disadvantage, special hardware is needed to:**

- \* Cause an interrupt (I/O device)
- \* Detect an interrupt (processor)
- \* Save the proper states to resume after the interrupt (processor)

# I/O Interrupt

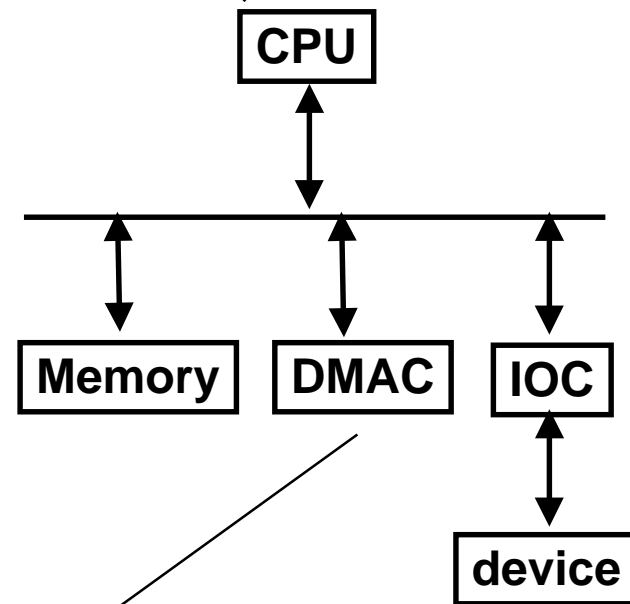
- **An I/O interrupt is just like the exceptions except:**
  - \* **An I/O interrupt is asynchronous**
  - \* **Further information needs to be conveyed**
- **An I/O interrupt is asynchronous with respect to instruction execution:**
  - \* **I/O interrupt is not associated with any instruction**
  - \* **I/O interrupt does not prevent any instruction from completion**
    - **You can pick your own convenient point to take an interrupt**
- **I/O interrupt is more complicated than exception:**
  - \* **Needs to convey the identity of the device generating the interrupt**
  - \* **Interrupt requests can have different urgencies:**
    - **Interrupt request needs to be prioritized**

# Delegating I/O Responsibility from the CPU: DMA

- **Direct Memory Access (DMA):**

- \* External to the CPU
- \* Act as a maser on the bus
- \* Transfer blocks of data to or from memory without CPU intervention

CPU sends a starting address, direction, and length count to DMAC. Then issues "start".



DMAC provides handshake signals for Peripheral Controller, and Memory Addresses and handshake signals for Memory.

# Summary:

- **Disk I/O Benchmarks: I/O rate vs. Data rate vs. latency**
- **Three Components of Disk Access Time:**
  - \* **Seek Time: advertised to be 8 to 12 ms. May be lower in real life.**
  - \* **Rotational Latency: 4.1 ms at 7200 RPM and 8.3 ms at 3600 RPM**
  - \* **Transfer Time: 2 to 12 MB per second**
- **Three types of buses: Processor-memory, I/O, Back-plane**
  - \* **performance Vs. cost**
- **Bus arbitration schemes: simplicity Vs. fairness**
- **I/O device notifying the operating system:**
  - \* **Polling: it can waste a lot of processor time**
  - \* **I/O interrupt: similar to exception except it is asynchronous**
- **Delegating I/O responsibility from the CPU: DMA, or even IOP**