

Chapter 6

Storage, Network and other I/O Topics

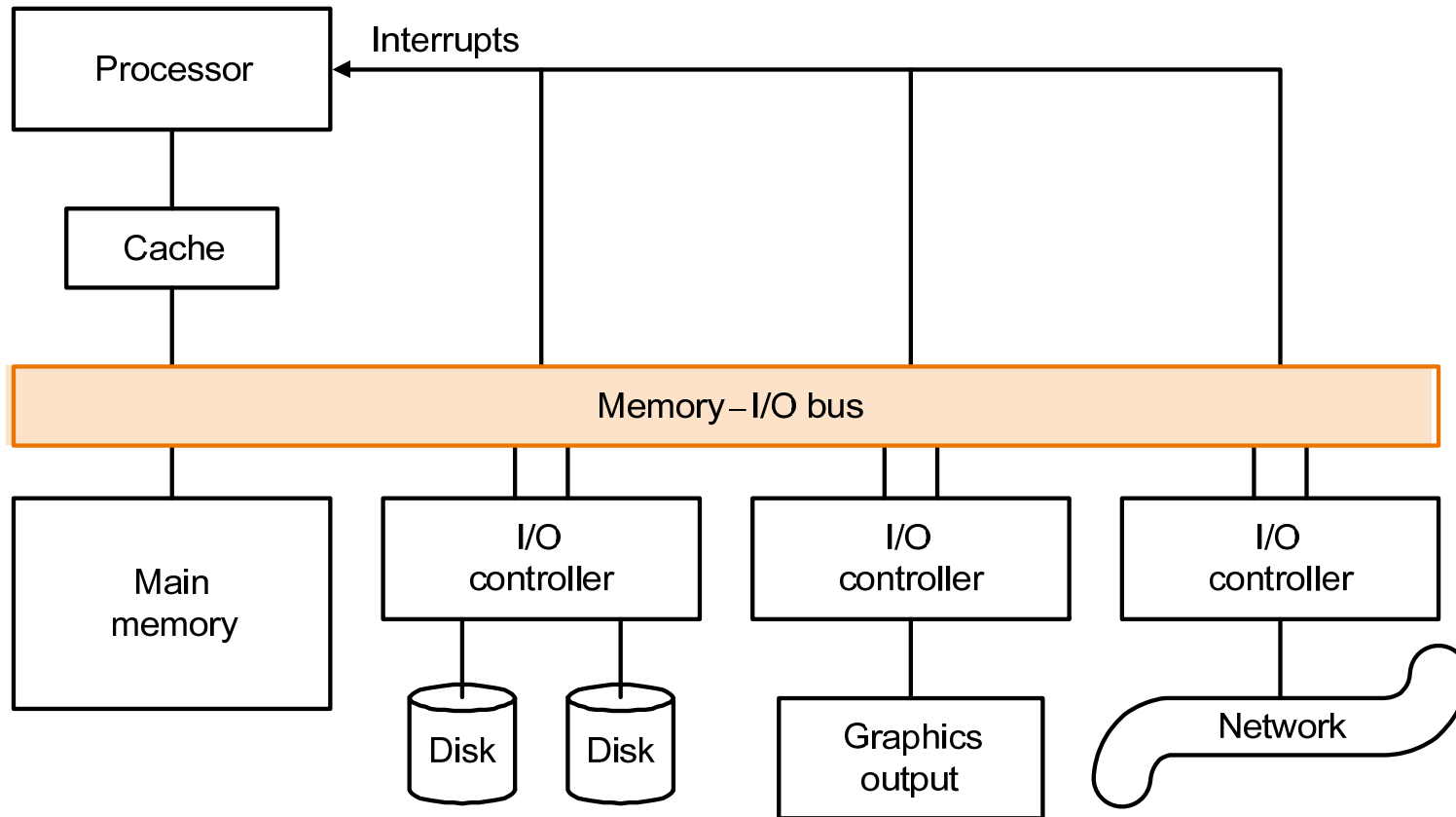
Contents

- ❖ 6.1 Introduction
- ❖ 6.2 Disk Storage and Dependability
- ❖ 6.3 Networks (Skim)
- ❖ 6.4 Buses and Other Connections between Processors Memory, and I/O Devices
- ❖ 6.5 Interfacing I/O Devices to the Memory, Processor, and Operating System
- ❖ 6.6 I/O Performance Measures:
Examples from Disk and File Systems
- ❖ 6.7 Designing an I/O system
- ❖ 6.8 Real Stuff: A Typical Desktop I/O System

6.1 Introduction

- ❖ I/O Designers must consider many factors
 - ⌘ such as expandability and resilience(resume),as well as performance.
- ❖ Assessing I/O system performance is very difficult.
 - ⌘ Different situations need different measurements.
- ❖ Performance of I/O system depends on:
 - ⌘ connection between devices and the system
 - ⌘ the memory hierarchy
 - ⌘ the operating system

❖ Typical collection of I/O devices



❖ Three characteristics

❧ Behavior

- ❖ Input (read once), output (write only, cannot read) ,or storage (can be reread and usually rewritten)

❧ Partner

- ❖ Either a human or a machine is at the other end of the I/O device, either feeding data on input or reading data on output.

❧ Data rate

- ❖ The peak rate at which data can be transferred between the I/O device and the main memory or processor.

❖ The diversity of I/O devices

Device	Behavior	Partner	Data rate (KB/sec)
Keyboard	input	human	0.01
Mouse	input	human	0.02
Voice input	input	human	0.02
Scanner	input	human	400.00
Voice output	output	human	0.60
Line printer	output	human	1.00
Laser printer	output	human	200.00
Graphics display	output	human	60,000.00
Modem	input or output	machine	2.00-8.00
Network/LAN	input or output	machine	500.00-6000.00
Floppy disk	storage	machine	100.00
Optical disk	storage	machine	1000.00
Magnetic tape	storage	machine	2000.00
Magnetic disk	storage	machine	2000.00-10,000.00

❖ I/O performance depends on the application:

↻ *throughput*:

In these cases, I/O bandwidth is the most important. Even I/O bandwidth can be measured in two different ways according to different situations:

1. How much data can we move through the system in a certain time?

For example, in many supercomputer applications, most I/O requires are for long streams of data, and transfer bandwidth is an important characteristic.

2. How many I/O operations can we do per unit of time?

For example, National Income Tax Service mainly processes large number of small files.

∞ *response time* (e.g., workstation and PC)

∞ both *throughput* and *response time* (e.g., ATM)

❖ Important but neglected

- ❧ *“The difficulties in assessing and designing I/O systems have often relegated I/O to second class status”*
- ❧ *“courses in every aspect of computing, from programming to computer architecture often ignore I/O or give it scanty coverage”*
- ❧ *“textbooks leave the subject to near the end, making it easier for students and instructors to skip it!”*

❖ Amdahl's law remind us that ignoring I/O is dangerous

❧ Assume:

❖ a bench mark executes in 100 seconds of elapsed time , where 90 seconds is for CPU and the rest is for I/O.

❖ CPU performance doubles per year, but I/O time doesn't improve.

❖ How much faster will our program run after three years?

❧ Elapsed time = CPU time+ I/O time

❧ Improvement in CPU performance is $90/11 = 8$ times

❧ Improvement in elapsed time is only $100/21 = 4.5$ times.

6.2 Disk Storage and Dependability

- ❖ Two major types of magnetic disks

 - ❧ floppy disks

 - ❧ hard disks

 - ❖ larger

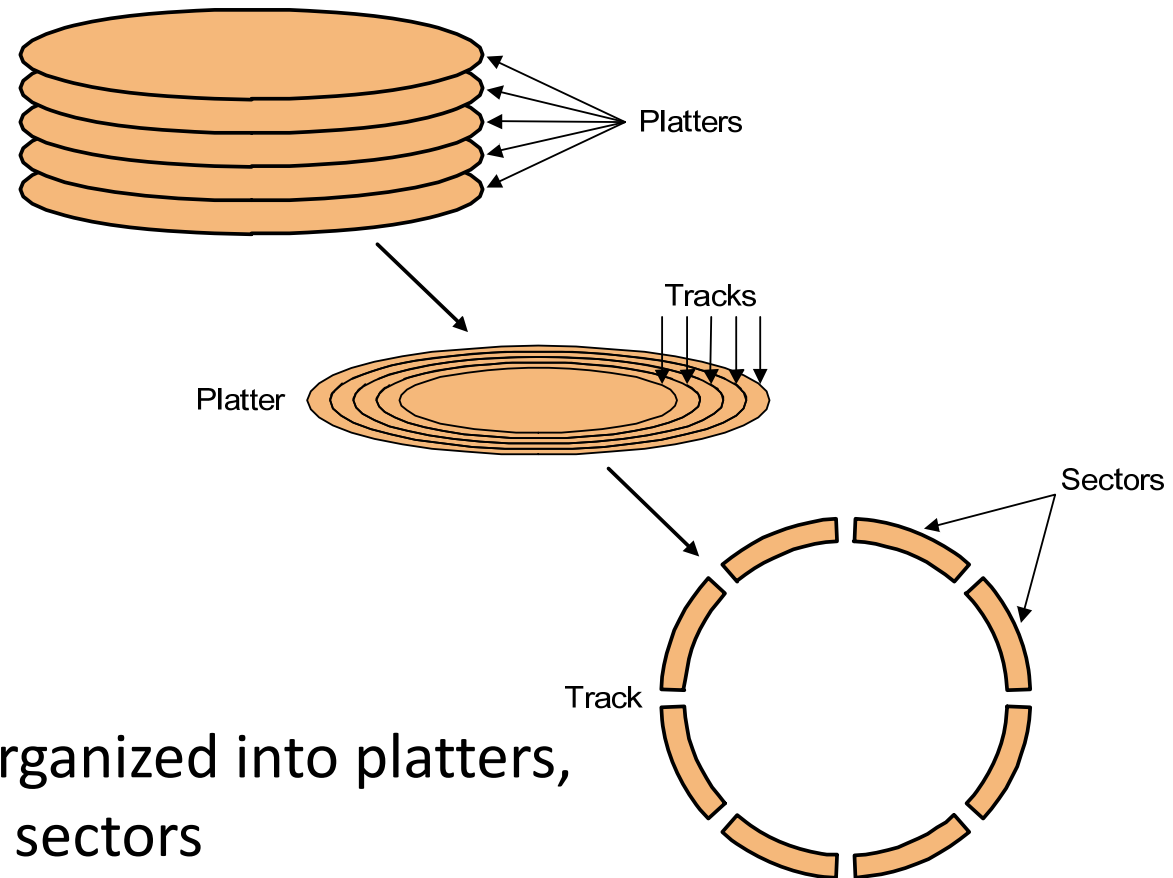
 - ❖ higher density

 - ❖ higher data rate

 - ❖ more than one platter

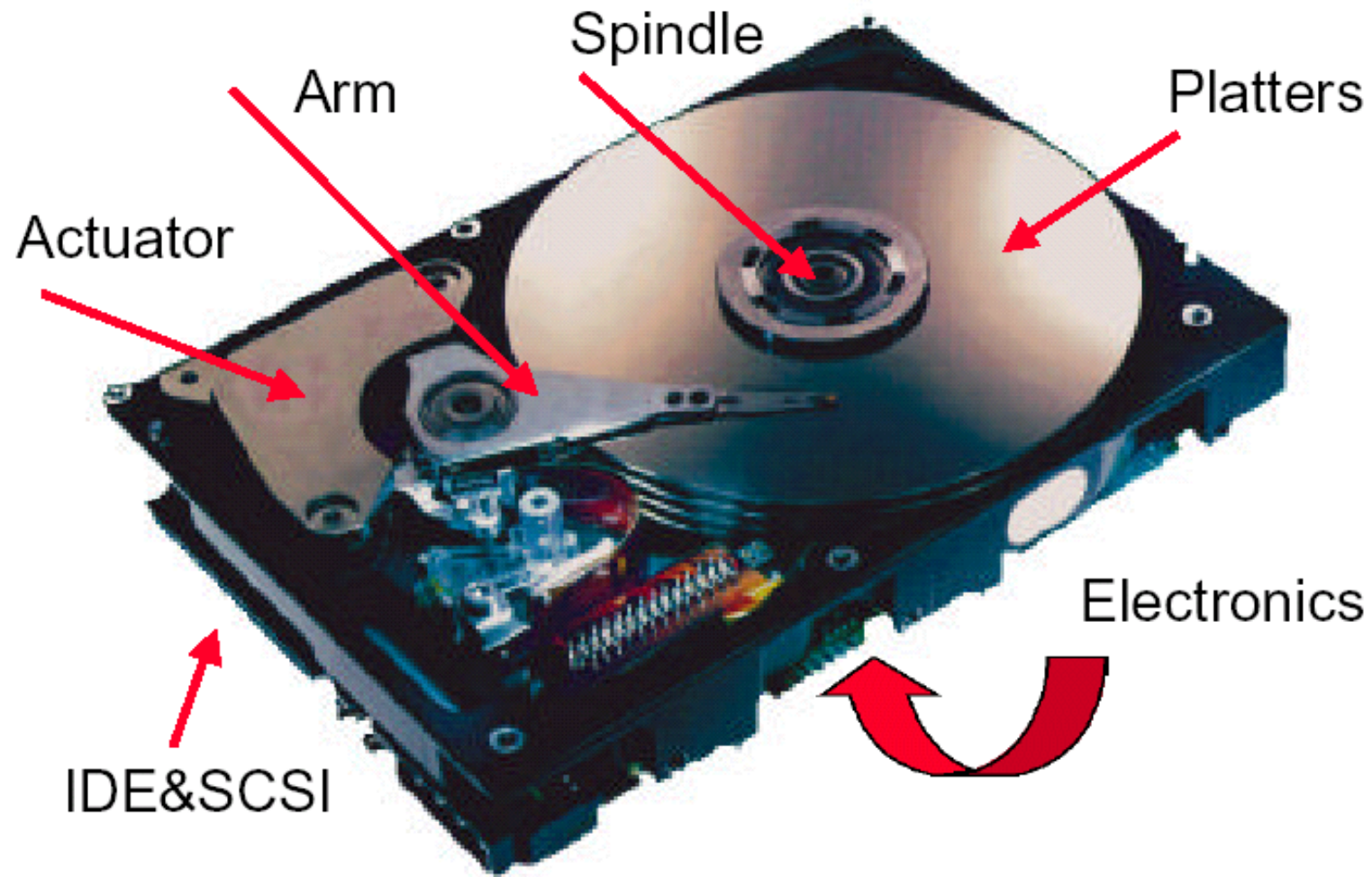
❖ The organization of hard disk

- ⌘ platters: disk consists of a collection of *platters*, each of which has two recordable disk surfaces
- ⌘ tracks: each disk surface is divided into concentric circles
- ⌘ sectors: each track is in turn divided into *sectors*, which is the smallest unit that can be read or written



❖ Disks are organized into platters, tracks, and sectors

What's Inside A Disk Drive?



❖ To access data of disk(p562):

☞ **Seek**: position read/write head over the proper track

❖ minimum seek time

❖ maximum seek time

❖ average seek time (3 to 14 ms)

☞ **Rotational latency**: wait for desired sector

❖ average latency is the half-way round the disk.

$$\begin{aligned} \text{Average rotational latency} &= \frac{0.5 \text{ rotation}}{5400\text{RPM}} = \frac{0.5 \text{ rotation}}{5400\text{RPM} \div \left(60 \frac{\text{seconds}}{\text{minute}}\right)} \\ &= 0.0056 \text{ seconds} = 5.6 \text{ ms} \end{aligned}$$

$$\begin{aligned} \text{Average rotational latency} &= \frac{0.5 \text{ rotation}}{15000\text{RPM}} = \frac{0.5 \text{ rotation}}{15000\text{RPM} \div \left(60 \frac{\text{seconds}}{\text{minute}}\right)} \\ &= 0.0020 \text{ seconds} = 2.0 \text{ ms} \end{aligned}$$

∞ **Transfer:** time to transfer a sector (1 KB/sector): function of rotation speed, Transfer rate of today's drives - 30 to 80 MBytes/second

∞ **Disk controller,** which controls the transfer between the disk and the memory

Disk Read Time 512B/sector

Access Time = Seek time + Rotational Latency + Transfer time + Controller Time

$$= 6\text{ms} + \frac{0.5}{10,000\text{PRM}} + \frac{0.5\text{KB}}{50\text{MB/sec}} + 0.2\text{ms}$$

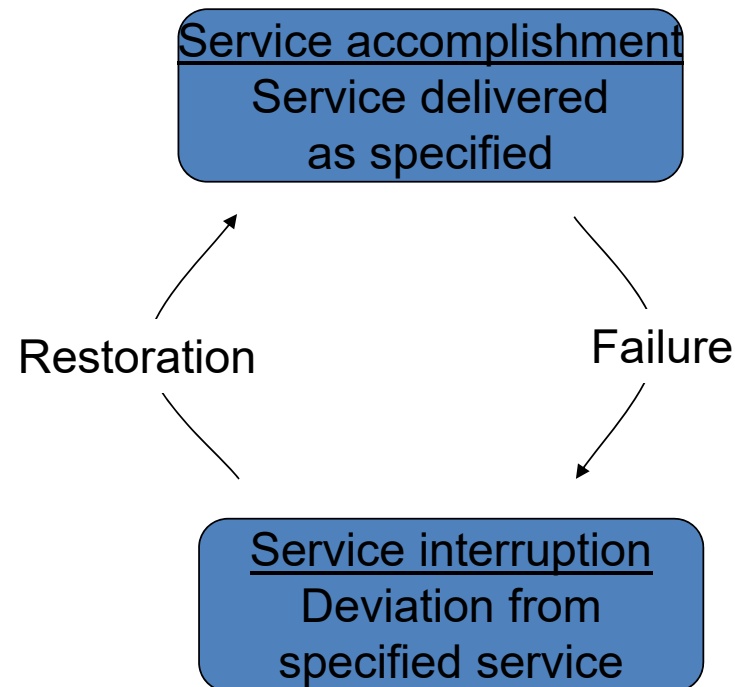
$$= 6\text{ms} + 3.0 + 0.01 + 0.2 = \mathbf{9.2\text{ms}}$$

Assuming the measured seek time is 25% of the calculated average

$$\text{Access Time} = 25\% \times 6\text{ms} + 3.0\text{ms} + 0.01\text{ms} + 0.2\text{ms} = \mathbf{4.7\text{ms}}$$

Dependable Memory Hierarchy

- **Dependability:** *Computer system **dependability** is the **quality** of delivered service such that reliance can justifiably be placed on this service. The service delivered by a system is its observed actual behavior as perceived by other system (s) interacting with this system's users.*



▣ **Fault: failure of a component**

- May or may not lead to system failure

Dependability Measures

- ❖ *MTTF* mean time to failure -- *reliability* 平均无故障时间
- ❖ *MTTR* mean time to repair 平均修复时间
- ❖ *MTBF* (Mean Time Between Failures)
= *MTTF* + *MTTR* 平均故障间隔时间

❖ *Availability*

$$Availability = \frac{MTTF}{MTTF + MTTR}$$

Three way to improve MTTF

Fault avoidance:

preventing fault occurrence by construction

Fault tolerance:

using redundancy to allow the service to comply with the service specification despite faults occurring, which applies primarily to hardware faults

Fault forecasting:

predicting the presence and creation of faults, which applies to hardware and software faults

Reasons for failure

Operator	Software	Hardware	System	Year data collected
42%	25%	18%	Data center (Tandem)	1985
15%	55%	14%	Data center (Tandem)	1989
18%	44%	39%	Data center (DEC VAX)	1985
50%	20%	30%	Data center (DEC VAX)	1993
50%	14%	19%	U.S. public telephone network	1996
54%	7%	30%	U.S. public telephone network	2000
60%	25%	15%	Internet services	2002

The Hamming SEC Code

- Hamming distance
 - Number of bits that are different between two bit patterns
- Minimum distance = 2 provides single bit error detection
 - E.g. parity code
- Minimum distance = 3 provides single error correction, 2 bit error detection

Encoding SEC

- To calculate Hamming code:
 - Number bits from 1 on the left
 - All bit positions that are a power 2 are parity bits (1,2,4,8,16,.....)
 - Each parity bit checks certain data bits:

Bit position		1	2	3	4	5	6	7	8	9	10	11	12
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
Parity bit coverate	p1	X		X		X		X		X		X	
	p2		X	X			X	X			X	X	
	p4				X	X	X	X					X
	p8								X	X	X	X	X

Decoding SEC

- Value of parity bits indicates which bits are in error
 - Use numbering from encoding procedure
 - E.g.
 - Parity bits = 0000 indicates no error
 - Parity bits = 1010 indicates bit 10 was flipped

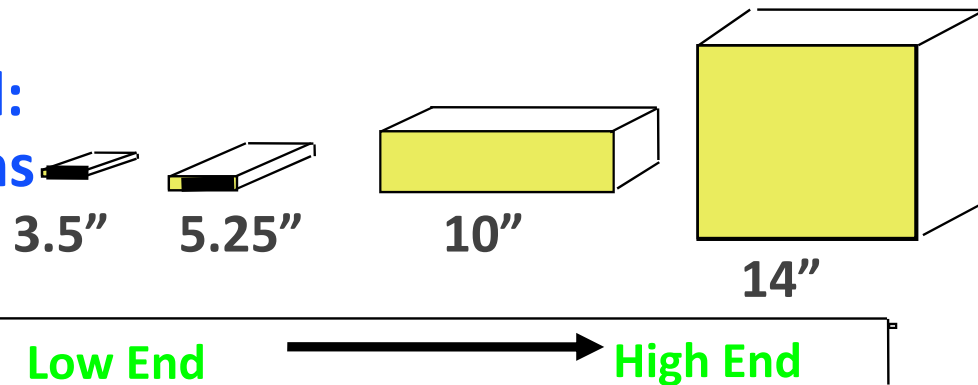
SEC/DED Code

- Add an additional parity bit for the whole word (p_n)
- Make Hamming distance = 4
- Decoding:
 - Let H = SEC parity bits
 - H even, p_n even, no error
 - H odd, p_n odd, correctable single bit error
 - H even, p_n odd, error in p_n bit
 - H odd, p_n even, double error occurred
- Note: ECC DRAM uses SEC/DEC with 8 bits protecting each 64 bits

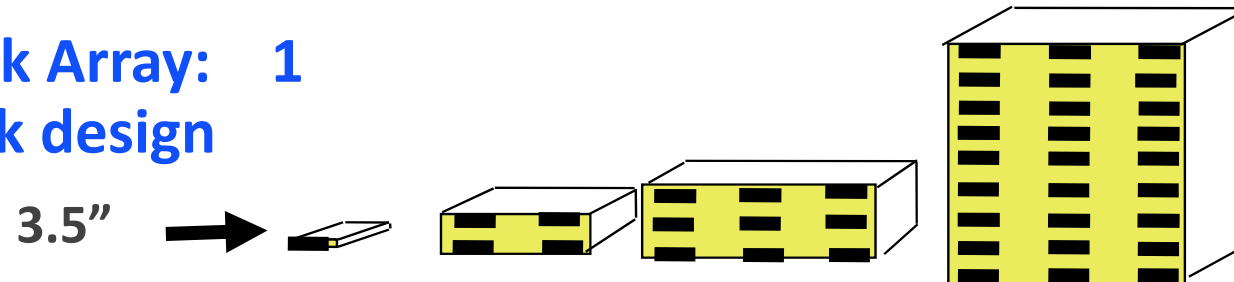
Use Arrays of Small Disks?

- Katz and Patterson asked in 1987:
 - Can smaller disks be used to close gap in performance between disks and CPUs?

Conventional:
4 disk designs



Disk Array: 1
disk design



Array Reliability

- **Reliability of N disks = Reliability of 1 Disk \div N**

50,000 Hours \div 70 disks = 700 hours

Disk system MTTF: Drops from 6 years to 1 month!

- **Arrays (without redundancy) too unreliable to be useful!**

Hot spares support reconstruction in parallel with access: very high media availability can be achieved

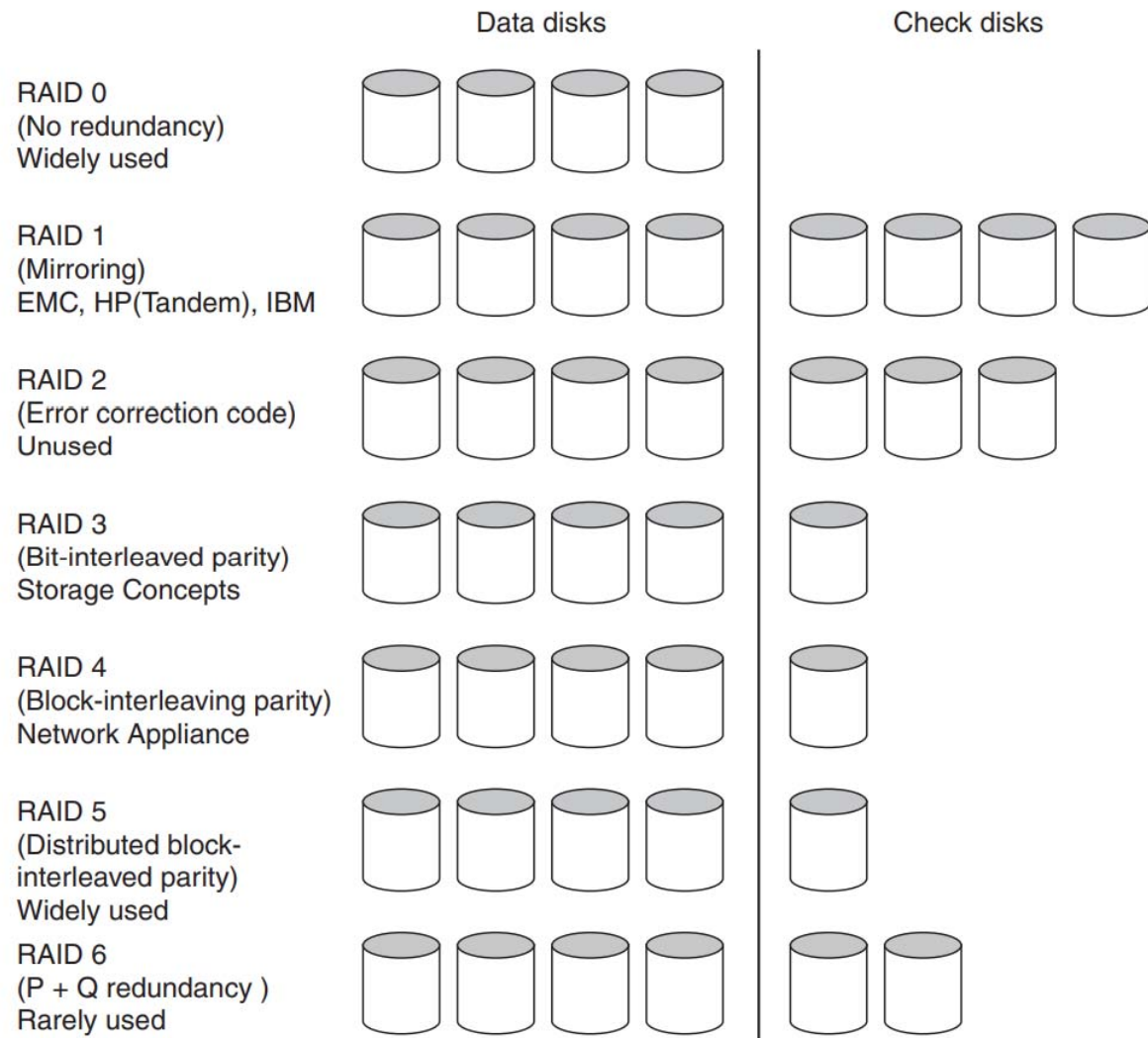
Redundant Arrays of (Inexpensive) Disks

- Files are "striped" across multiple disks
- Redundancy yields high data availability
 - Availability: service still provided to user, even if some components failed
- Disks will still fail
- Contents reconstructed from data redundantly stored in the array
 - ⇒ Capacity penalty to store redundant info
 - ⇒ Bandwidth penalty to update redundant info

RAID: Redundant Arrays of Inexpensive Disks

A disk arrays replace larger disk

RAID level		Minimum number of Disk faults survived	Example Data disks	Corresponding Check disks	Corporations producing RAID products at this level
0	Non-redundant striped	0	8	0	Widely used
1	Mirrored	1	8	8	EMC, Compaq (Tandem), IBM
2	Memory-style ECC	1	8	4	Error Checking and Correcting
3	Bit-interleaved parity	1	8	1	Storage Concepts
4	Block-interleaved parity	1	8	1	Network Appliance
5	Block-interleaved distributed parity	1	8	1	Widely used
6	P+Q redundancy	2	8	2	

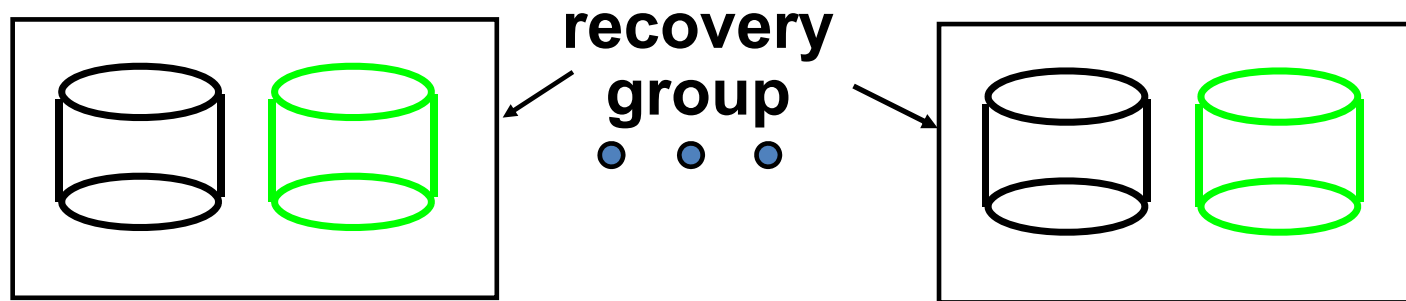


RAID 0: No Redundancy(p587)

- **Data is striped across a disk array but there is no redundancy to tolerate disk failure. It also improves performance for large accesses, since many disks can operate at once.**

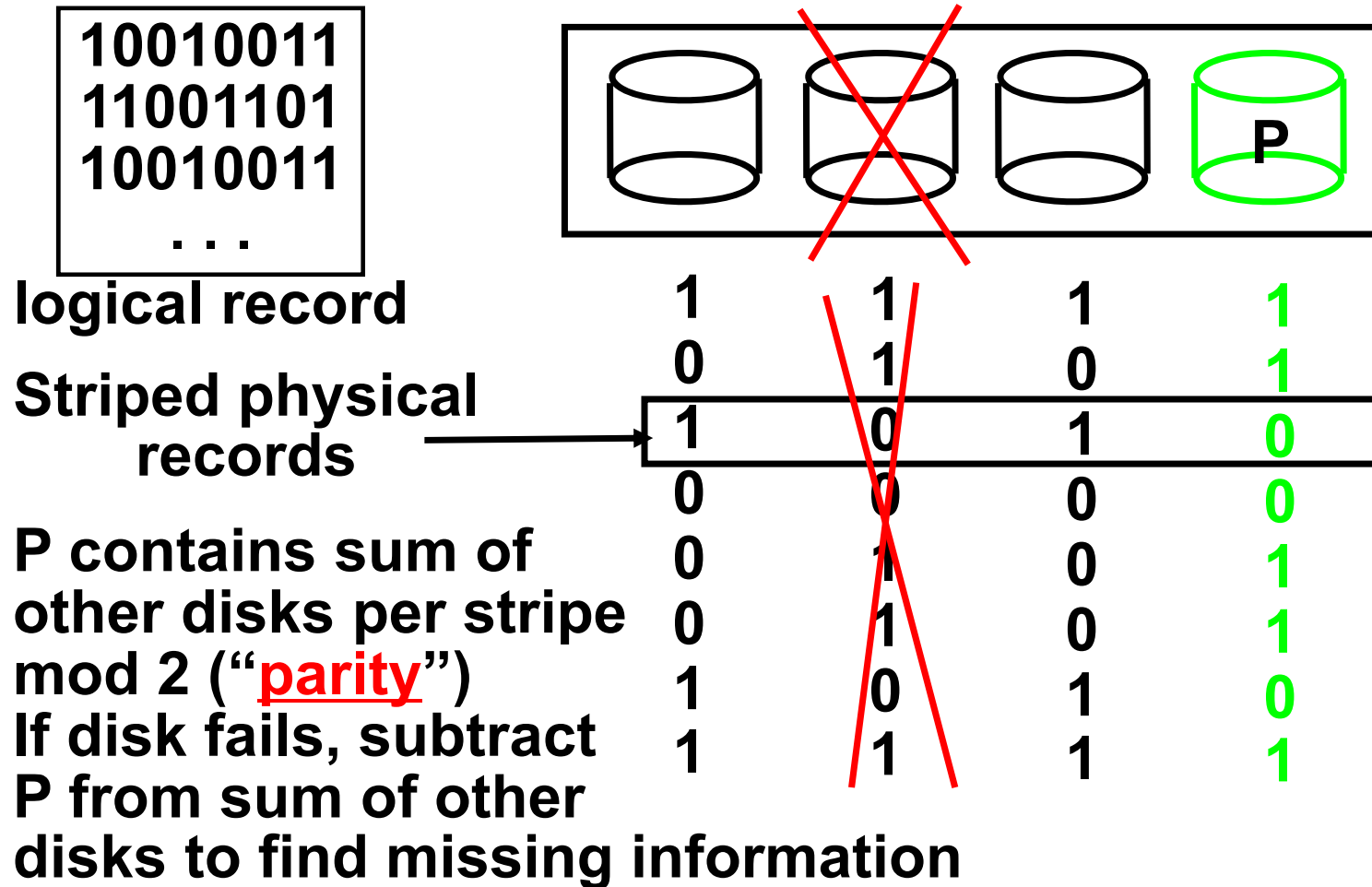
RAID 0 is something of a **misnomer as there is no Redundancy,(p587)**

RAID 1: Disk Mirroring/Shadowing



- Each disk is fully duplicated onto its “mirror”
Very high availability can be achieved
- Bandwidth sacrifice on write:
Logical write = two physical writes
 - Reads may be optimized
- Most expensive solution: 100% capacity overhead
- (RAID 2 not interesting, so skip)

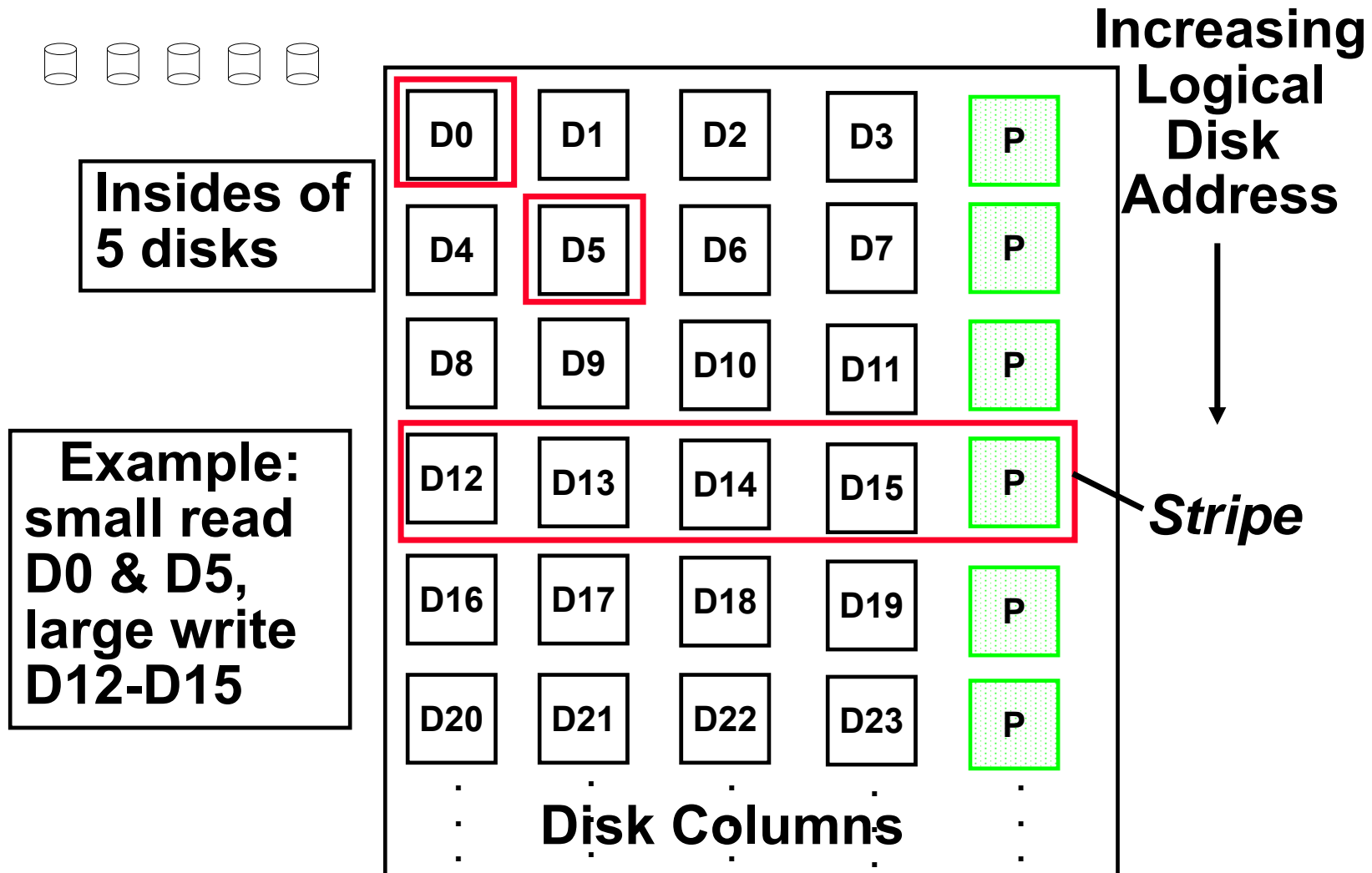
RAID 3: Bit-Interleaved Parity Disk



Inspiration for RAID 4

- RAID 3 relies on parity disk to discover errors on Read
- Every sector has an error detection field
- Relies on error detection field to catch errors on read, not on the parity disk
- Allows independent reads to different disks simultaneously

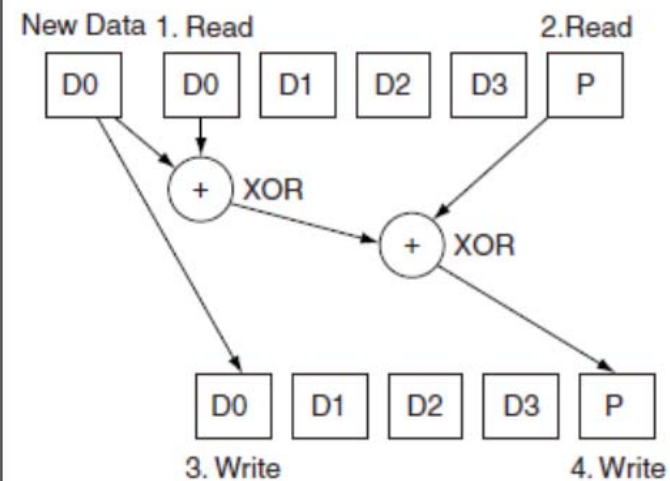
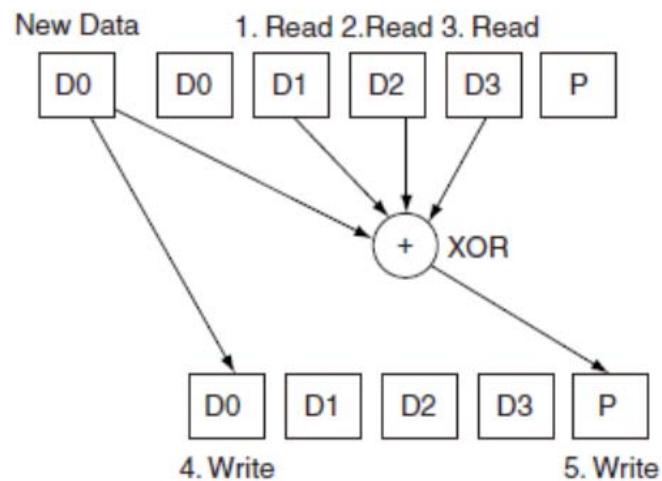
RAID 4: High I/O Rate Parity



Problems of Disk Arrays: Small Writes

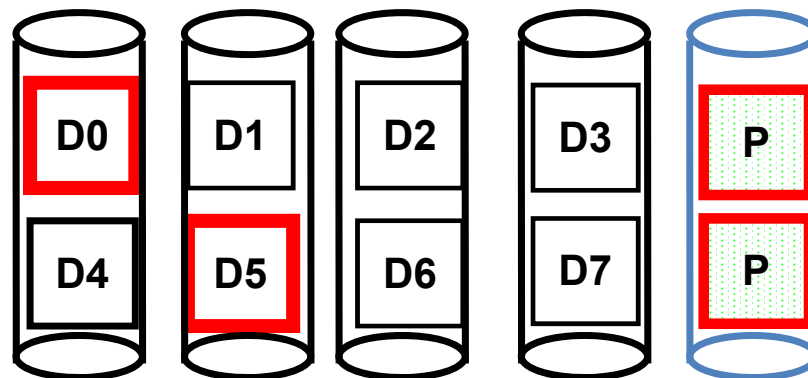
RAID-4: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes

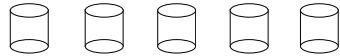


Inspiration for RAID 5

- RAID 4 works well for small reads
- Small writes (write to one disk):
 - Option 1: read other data disks, create new sum and write to Parity Disk
 - Option 2: since P has old sum, compare old data to new data, add the difference to P
- Small writes are limited by Parity Disk: Both Writes to D0 and D5 also write to P disk

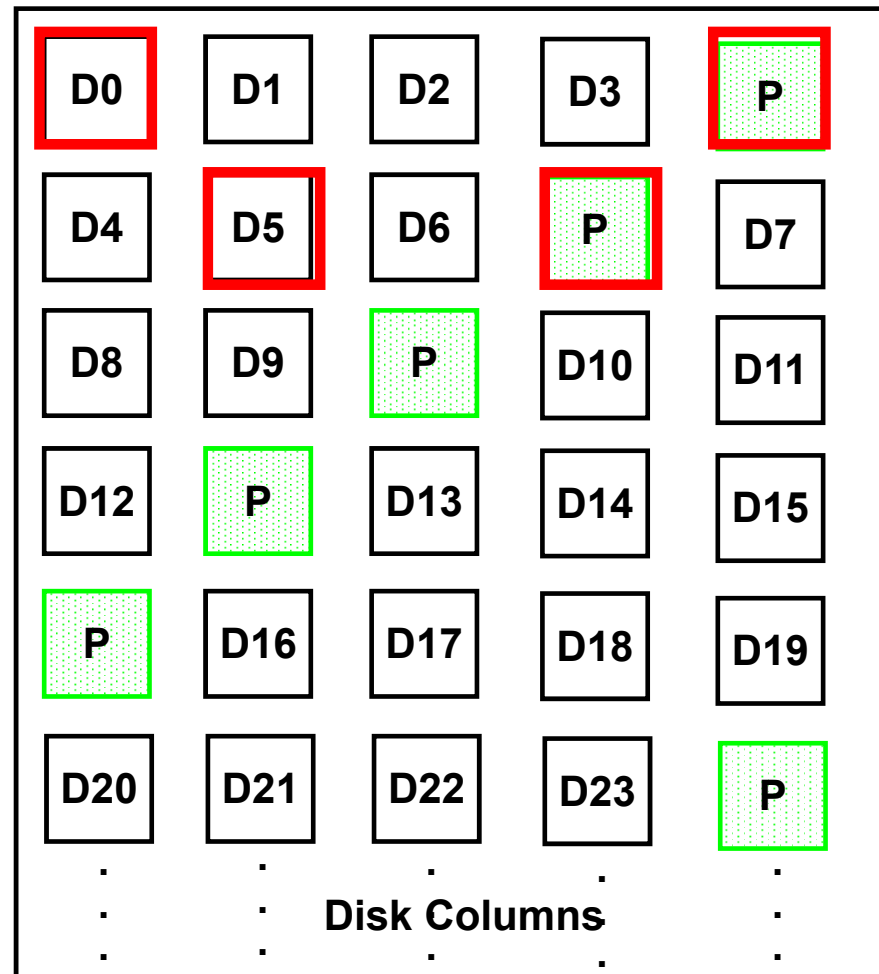


RAID 5: High I/O Rate Interleaved Parity



Independent writes are possible because of interleaved parity

**Example:
write to D0,
D5 uses
disks 0, 1,
3, 4**



RAID 6: P+Q Redundancy

- When a single failure correction is not sufficient, Parity can be generalized to have a second calculation over data and another check disk of information.

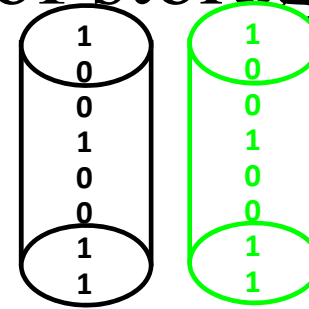
RAID Techniques: Goal is performance, popularity due to reliability of storage

- *Disk Mirroring, Shadowing (RAID 1)*

Each disk is fully duplicated onto its "shadow"

Logical write = two physical writes

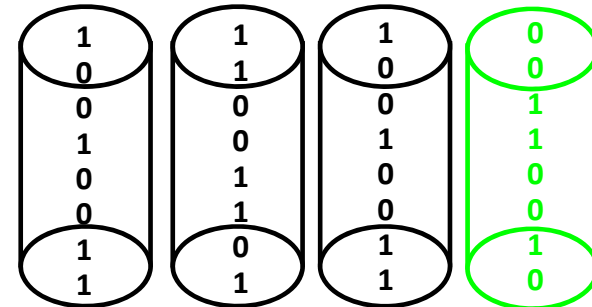
100% capacity overhead



- *Parity Data Bandwidth Array (RAID 3)*

Parity computed horizontally

Logically a single high data bw disk

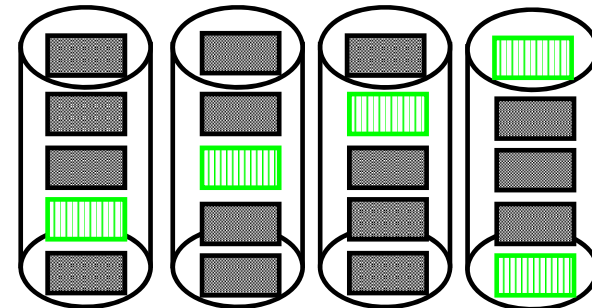


- *High I/O Rate Parity Array (RAID 5)*

Interleaved parity blocks

Independent reads and writes

Logical write = 2 reads + 2 writes



Which of the following are true about RAID levels 1, 3, 4, 5, and 6?

1. RAID systems rely on redundancy to achieve high availability.
2. RAID 1 (mirroring) has the highest check disk overhead.
3. For small writes, RAID 3 (bit-interleaved parity) has the worst throughput.
4. For large writes, RAID 3, 4, and 5 have the same throughput.

6.3 Networks (skim)

∞ Key characteristics of typical networks include the following

- ❖ **Distance: 0.01 to 10,000 kilometers**
- ❖ **Speed: 0.001MB/sec to 100MB/sec**
- ❖ **Topology: Bus, ring, star, tree**
- ❖ **Shared lines: None (point-to-point) or shared (multidrop)**

- ❧ Local area network (LAN) **e.g., Ethernet**
- ❧ Packet-switched network ,which are common in long-haul networks
e.g., ARPANET
- ❧ TCP/IP is the key to interconnecting different networks
- ❧ The bandwidths of networks are probably growing faster than the bandwidth of any other type of device at present.

6.4 Buses and Other Connections between Processors Memory, and I/O Devices(p568)

- ❖ Shared communication link (one or more wires)
- ❖ Difficult design:
 - ❧ may be bottleneck
 - ❧ length of the bus
 - ❧ number of devices
 - ❧ tradeoffs (fast bus accesses and high bandwidth)
 - ❧ support for many different devices
 - ❧ cost

❖ A bus contains two types of lines

❧ **Control lines**, which are used to signal requests and acknowledgments, and to indicate what types of information is on the data lines.

❧ **Data lines**, which carry information (e.g., data, addresses, and complex commands) between the source and the destination.

❖ Bus transaction

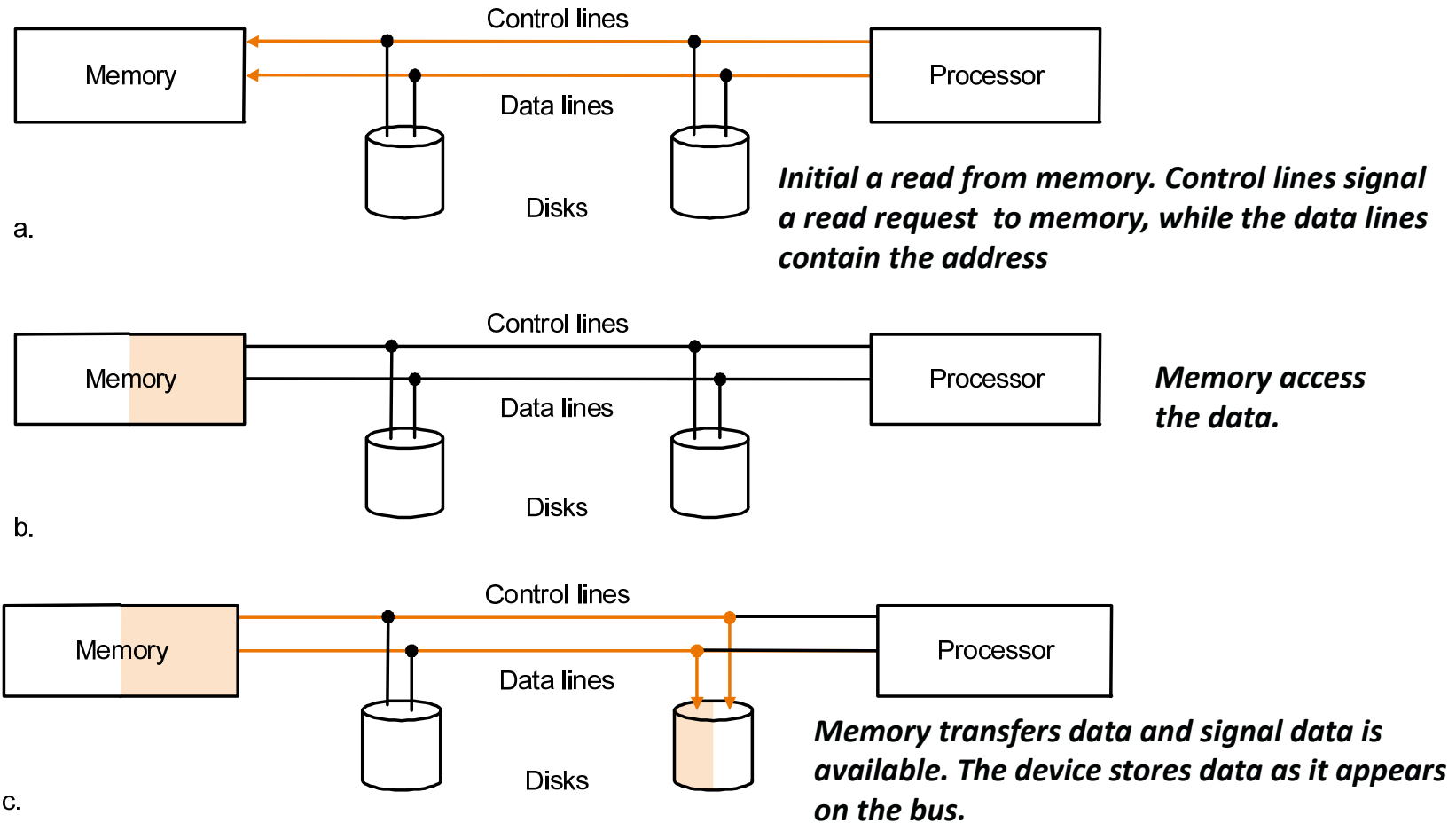
❧ include two parts: sending the address and receiving or sending the data

❧ two operations

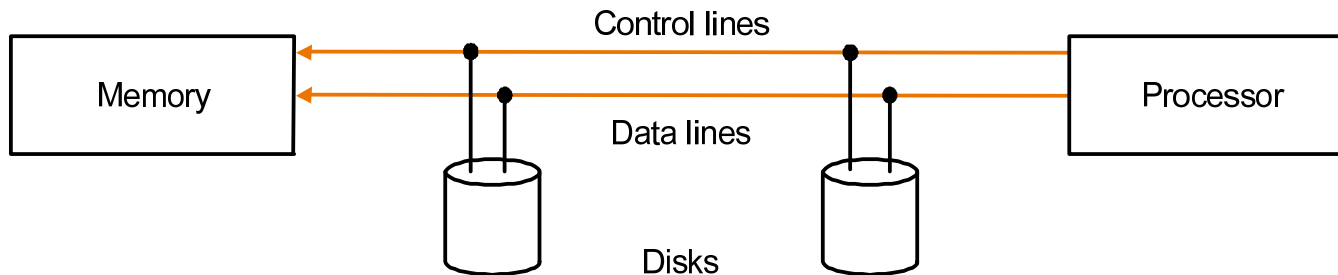
❖ **input**: inputting data from the device to memory

❖ **output**: outputting data to a device from memory

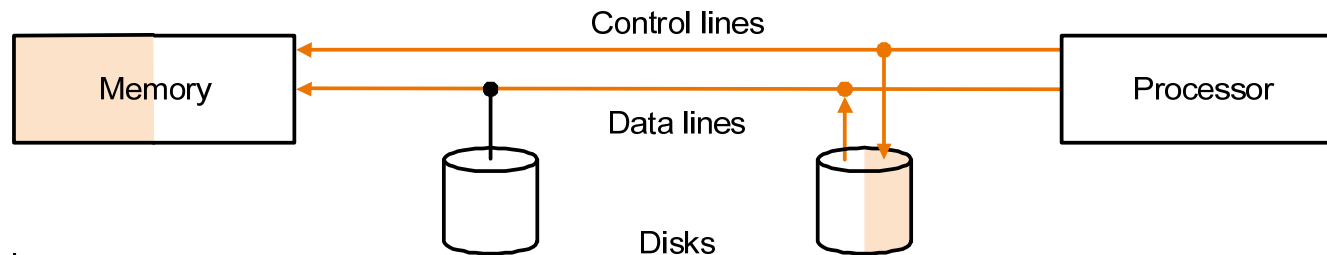
❖ The steps of an output operation.



❖ The steps of an input operation.



- a. *Control lines indicate a write request for memory, while the data lines contain the address*

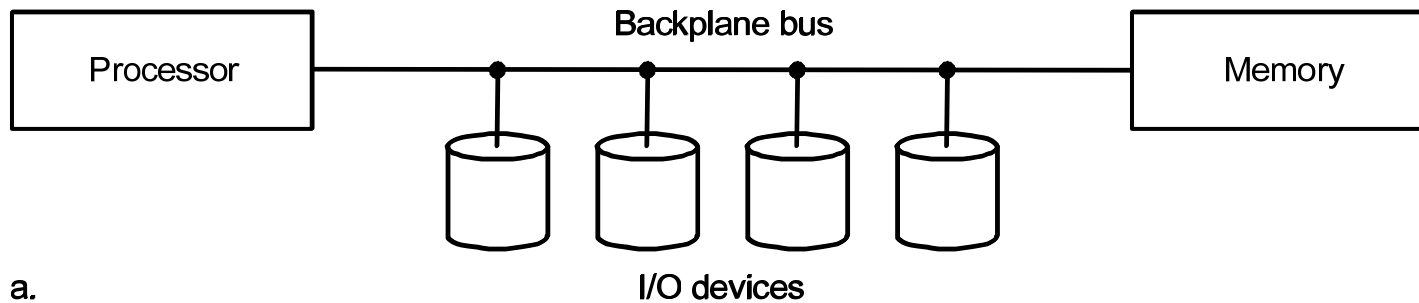


- b.

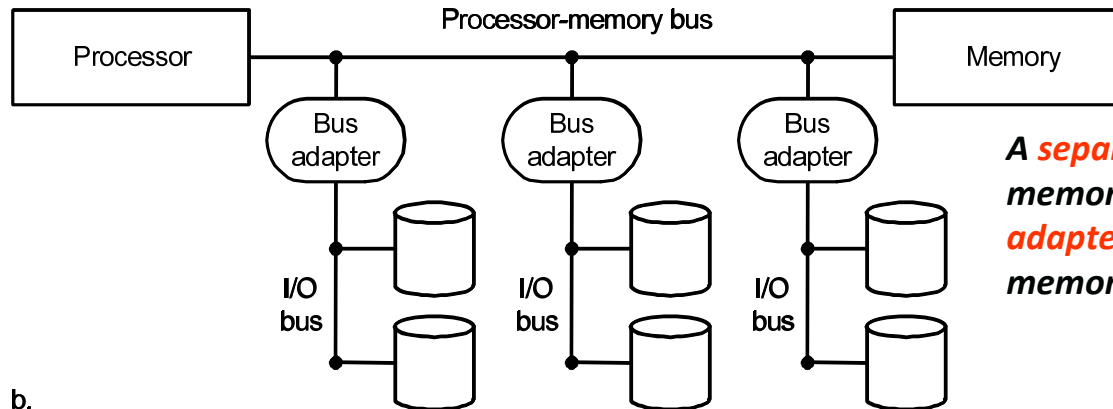
When the memory is ready, it signals the device, which then transfers the data. The memory will store the data as it receives it. The device need not wait for the store to be completed.

❖ Types of buses: (p566)

- ❧ processor-memory (short, high speed, custom design)
- ❧ backplane (high speed, often standardized, e.g., PCI)
- ❧ I/O (lengthy, different devices, standardized, e.g., SCSI)

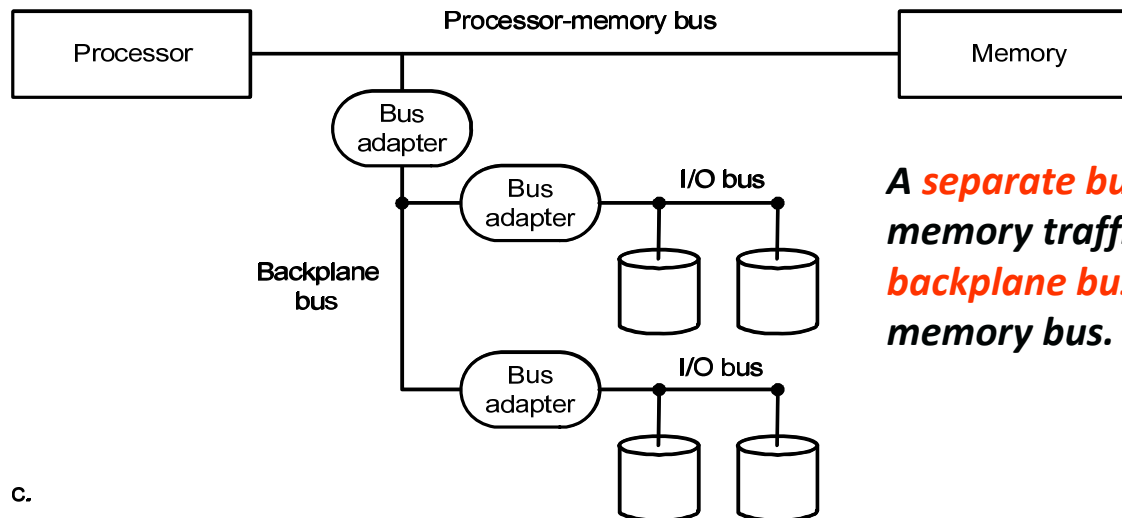


*Older PCs often use a **single bus** for processor-to-memory communication, as well as communication between I/O devices and memory.*



*A **separate bus** is used for processor-memory traffic. The I/O bus use a **bus adapter** to interface to the processor-memory bus.*

b.



*A **separate bus** is used for processor-memory traffic. A small number of **backplane buses** tap into the processor-memory bus.*

c.

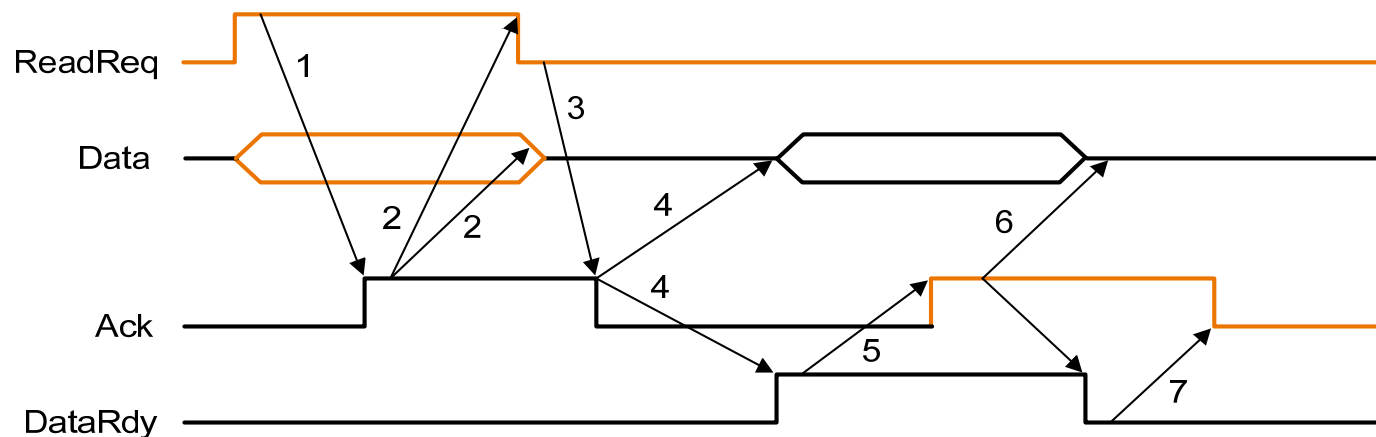
❖ Synchronous vs. Asynchronous

- ❧ Synchronous bus use a clock and a synchronous protocol, fast and small but every device must operate at same rate and clock skew requires the bus to be short
- ❧ Asynchronous bus don't use a clock and instead use *handshaking*

❖ Handshaking protocol

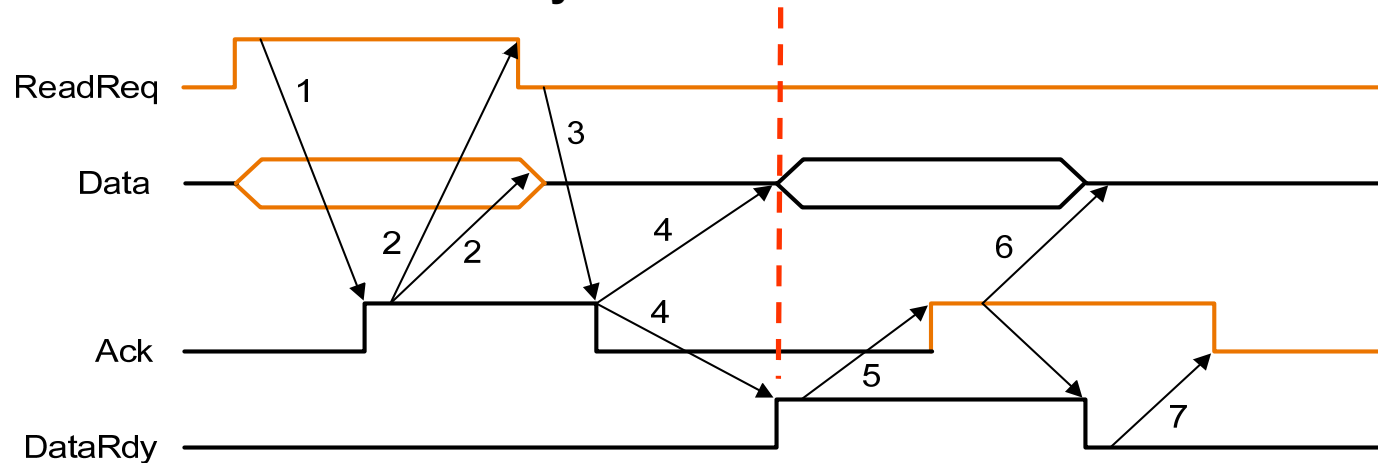
- ❧ Our example ,which illustrates how asynchronous buses use handshaking, assumes there are three control lines.
 - ❖ *ReadReq: Used to indicate a read request for memory. The address is put on the data lines at the same time.*
 - ❖ *DataRdy: Used to indicate that data word is now ready on the data lines.*
 - ❖ *Ack: Used to acknowledge the ReadReq or the DataRdy signal of the other party*

- ❖ **Example:** The asynchronous handshaking consists of seven steps to read a word from memory and receive it in an I/O device.



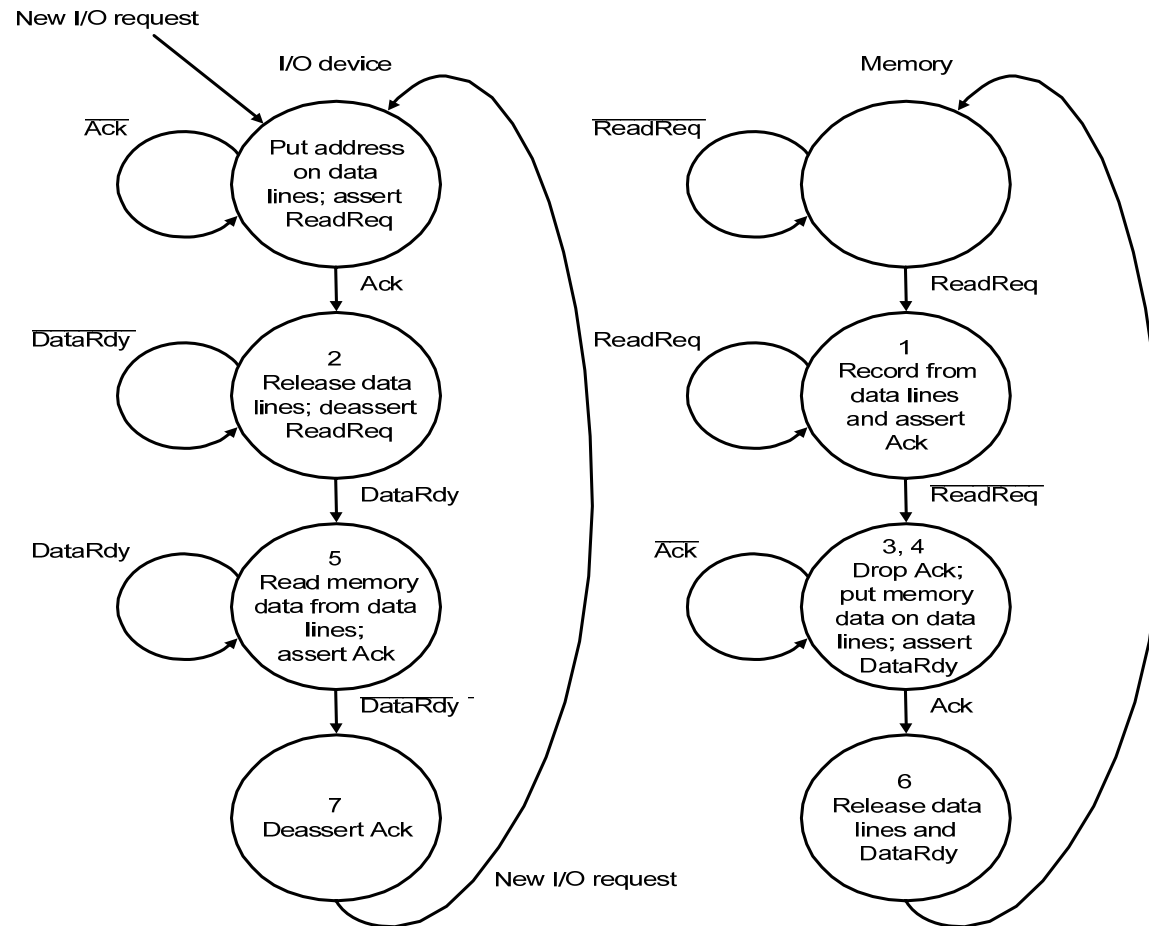
1. When memory saw the **ReadReq** line, it reads the address from the data bus, starts the memory read operation, then raises **Ack** to tell the device that the ReadReq signal has been seen.
2. I/O device saw the Ack line high and releases the **ReadReq** data lines.
3. Memory sees that ReadReq is low and drops the **Ack** line.

- ❖ **Example:** The asynchronous handshaking consists of seven steps to read a word from memory and receive it in an I/O device.



4. When the memory has the data ready, it places the data on the data lines and raises **DataRdy**.
5. The I/O device sees DataRdy, reads the data from the bus , and signals that it has the data by raising **ACK**.
6. The memory sees Ack signals, drops **DataRdy**, and releases the data lines.
7. Finally, the I/O device, seeing DataRdy go low, drops the **ACK** line, which indicates that the transmission is completed.

- ❖ These finite state machines implement the control for handshaking protocol illustrated in former example.

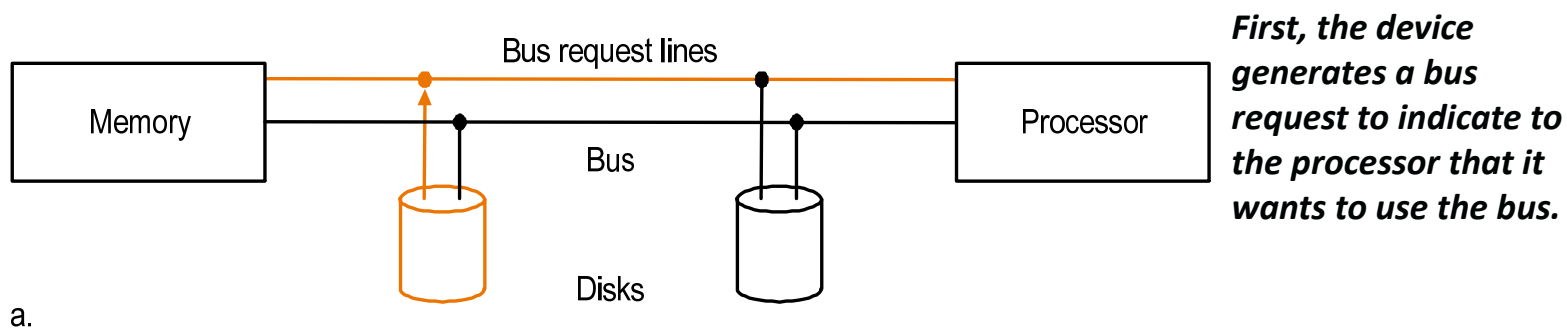


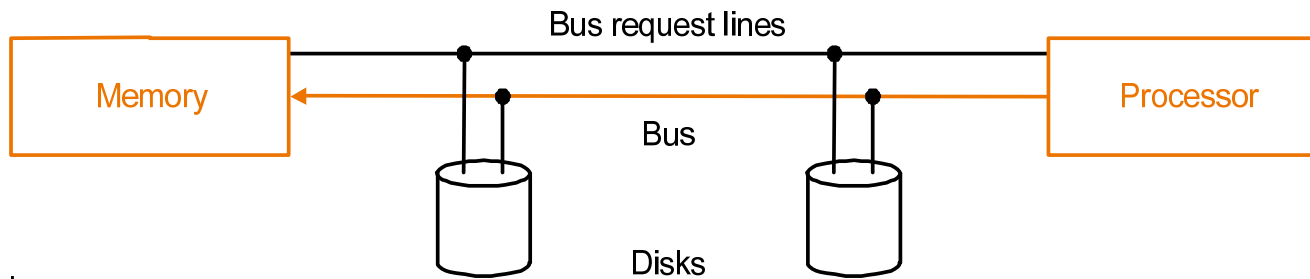
❖ Obtaining Access to the Bus

- “Without any control, multiple device desiring to communicate could each try to assert the control and data lines for different transfers!”
- So, a bus master is needed. Bus masters initiate and control all bus requests.

e.g., processor is always a bus master.

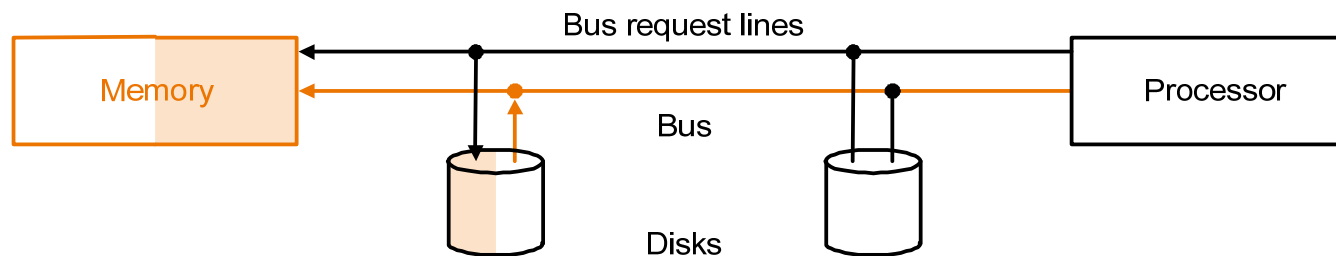
- Example: the initial steps in a bus transaction with a single master (the processor).





b.

The processor responds and generates appropriate bus control signals. For example, if the device wants to perform output from memory, the processor asserts the read request lines to memory.



c.

The processor also notifies the device that its bus request is being processed; as a result, the device knows it can use the bus and places the address for the request on the bus.

Bus Arbitration

- 一般都是多个设备共享一条总线进行数据通信，其中如果多个设备同时发送接收数据的话，从而产生总线竞争，会导致通信冲突导致通信失败，所以在总线上要引入一个仲裁机制来决定什么时间谁来占用总线的通信

❖ Bus Arbitration

☞ Deciding which device gets to use the bus next

☞ In a bus arbitration scheme, a device wanting to use the bus signals a bus request and is later granted the bus.

☞ four bus arbitration schemes:

- ❖ *daisy chain arbitration (not very fair):菊花链, 阻塞式级联*
- ❖ *centralized, parallel arbitration (requires an arbiter), e.g., PCI*
- ❖ *self selection, e.g., NuBus used in Macintosh*
- ❖ *collision detection, e.g., Ethernet*

❖ Two factors in choosing which device to grant the bus:

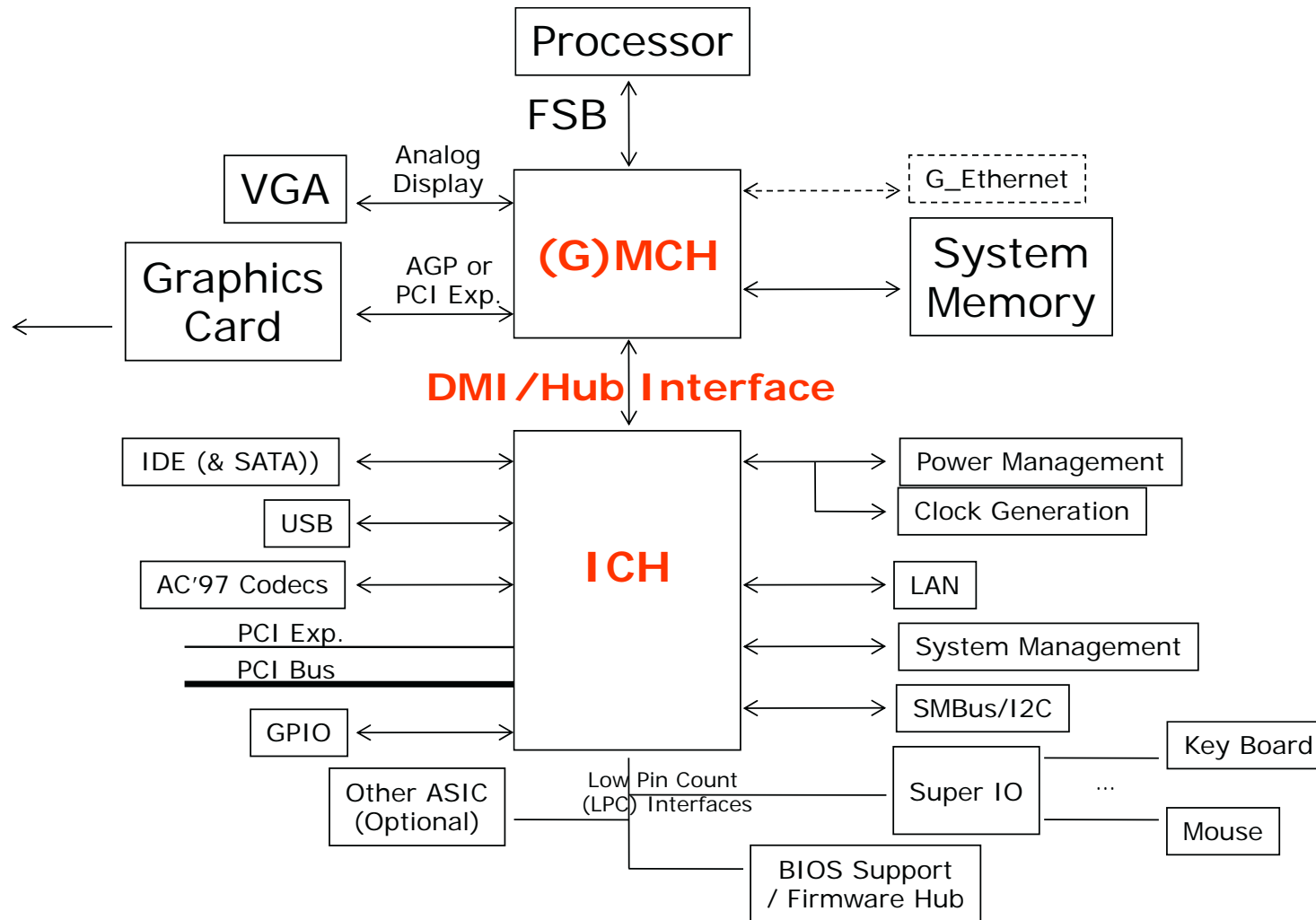
bus priority

fairness

❖ Bus Standards

- ❧ SCSI (*small computer system interface*)
- ❧ PCI (*peripheral component interconnect*)
- ❧ IPI (*intelligent peripheral interface*)
- ❧ IBMPC-AT IBMPC-XT
- ❧ ISA EISA

The Buses and Networks of Pentium(p571)



6.5 **Interfacing** I/O Devices to the Memory, Processor, and Operating System(p572)

❖ Three characteristics of I/O systems

- ❧ *shared by multiple programs using the processor.*
- ❧ *often use interrupts to communicate information about I/O operations.*
- ❧ *The low-level control of I/O devices is complex*

Three types of communication are required:

- ❧ *The OS must be able to give commands to the I/O devices.*
- ❧ *The device must be able to notify the OS, when I/O device completed an operation or has encountered an error.*
- ❧ *Data must be transferred between memory and an I/O device*

Giving Commands to I/O Devices

Two methods used to address the device

❧ **memory-mapped I/O:**

- ❖ portions of the memory address space are assigned to I/O devices, and lw and sw instructions can be used to access the I/O port.

❧ **special I/O instructions**

- ❖ Give a command to an I/O device

❧ **command port ,data port**

- ❖ The Status register (a done bit, an error bit.....)
- ❖ The Data register, The command register

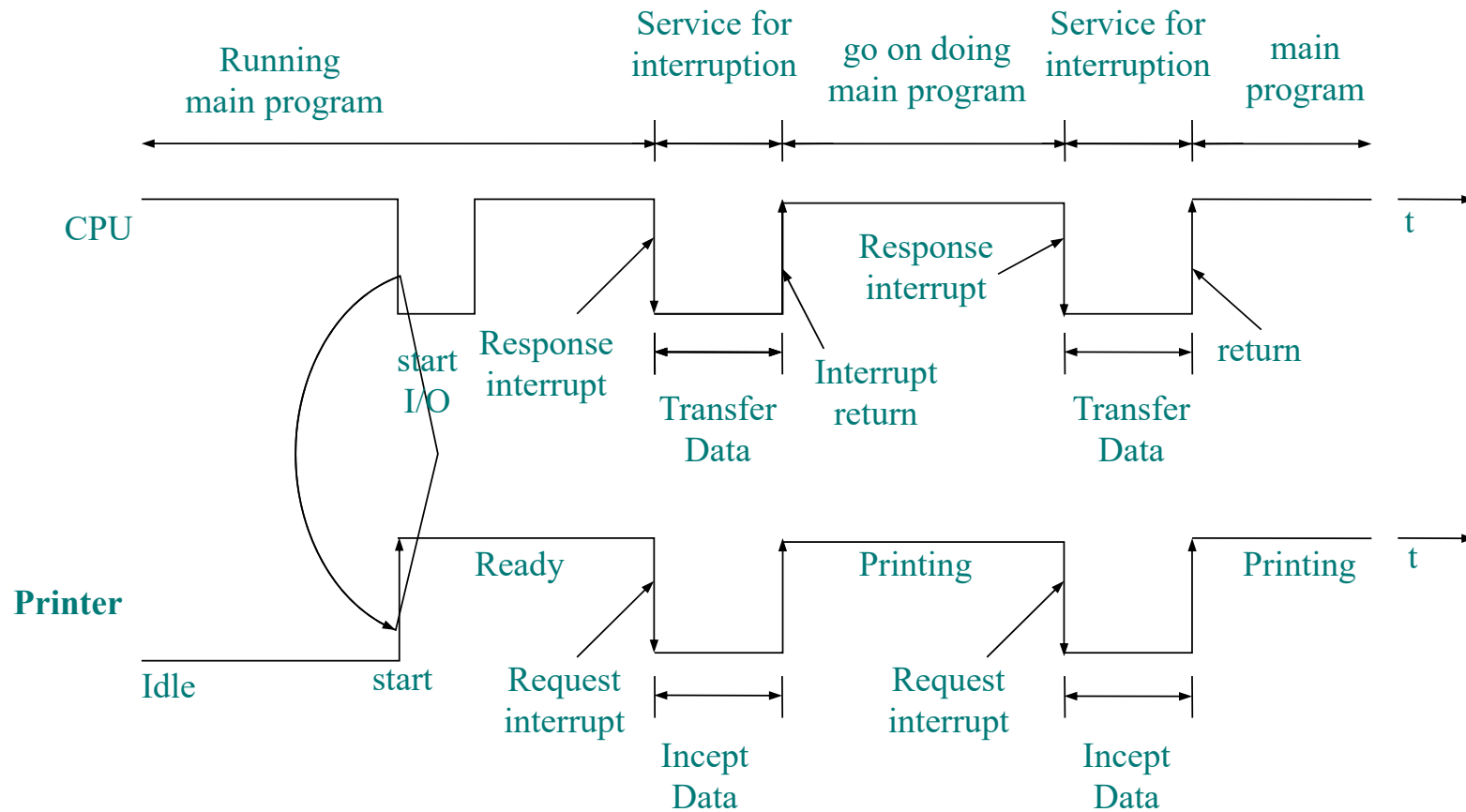
Communication with the Processor

I/O SYSTEM DATA TRANSFER CONTROL MODE

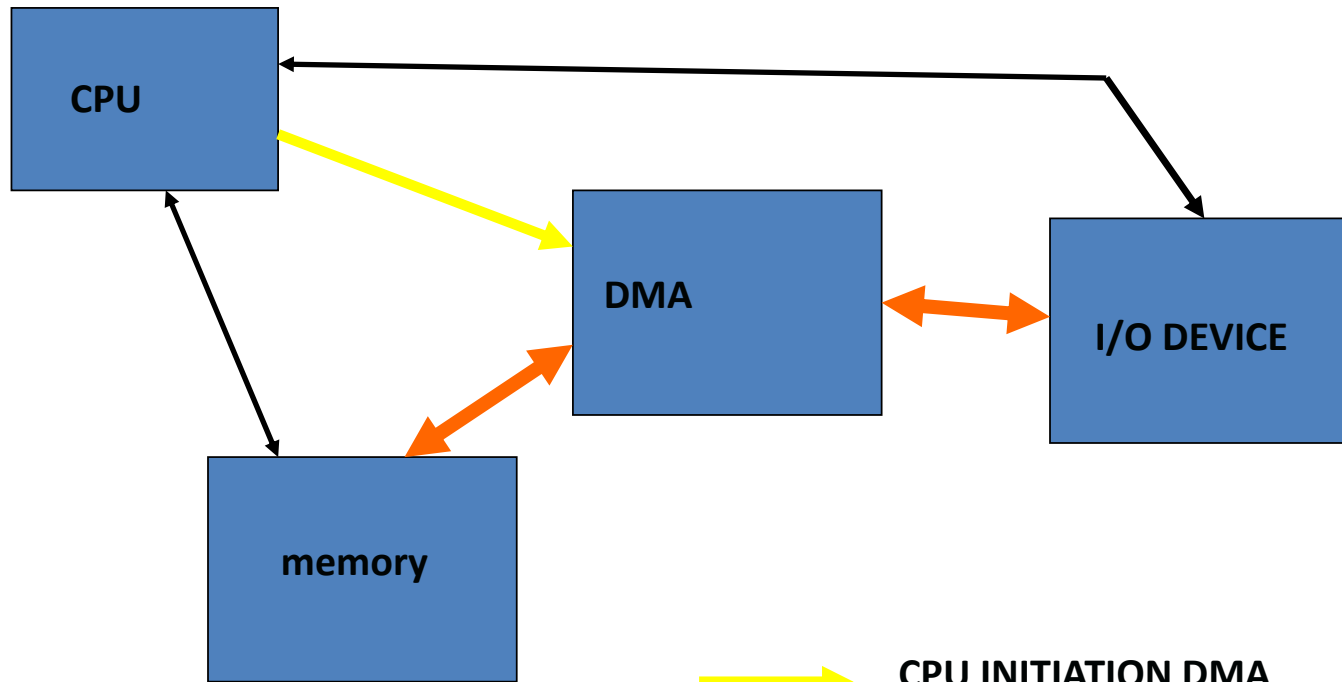
- ❧ **Polling**: The processor periodically checks status bit to see if it is time for the next I/O operation.
- ❧ **Interrupt**: When an I/O device wants to notify processor that it has completed some operation or needs attentions, it causes processor to be interrupted.
- ❧ **DMA** (*direct memory access*): the device controller transfer data directly to or from memory without involving processor.

Interrupt-Driven I/O mode

Advantage: concurrent operation



DMA transfer mode



→ CPU INITIATION DMA

NO DMA I/O-CPU--M

— DMA-- I/O---M I/O DIRECT ACCESS MEMORY

❖ A DMA transfer need three steps:

- ⌘ The processor **sets up** the DMA by supplying some information, including the ***identity of the device, the operation, the memory address that is the source or destination of the data to be transferred, and the number of bytes to transfer.***
- ⌘ The DMA **starts** the **operation** on the device and arbitrates for the **bus**. If the request requires more than one transfer on the bus, the DMA unit generates the next memory address and initiates the next transfer.
- ⌘ Once the DMA transfer is complete, the controller **interrupts** the processor, which then examines whether errors occur.

❖ **Compare polling, interrupts, DMA**

- ❧ The disadvantage of polling: wasting processor time.
When the CPU polls the I/O devices periodically, the I/O devices maybe have no request or have not get ready.
- ❧ If the I/O operations is interrupt driven, the OS can work on other tasks while data is being read from or written to the device.
- ❧ Because DMA doesn't need the control of processor, it will not consume much of processor time.

6.6 I/O Performance Measures: Examples from Disk and File Systems

- ❖ Supercomputer I/O Benchmarks

- ❖ Transaction Processing I/O Benchmarks

 - ⌘ I/O rate: the number of disk access per second, as opposed to data rate.

- ❖ File System I/O Benchmarks

 - ⌘ MakeDir, Copy, ScanDir, ReadAll, Make

Performance analysis of Synchronous versus Asynchronous buses

Assume: The synchronous bus has a clock cycle time of **50 ns**, and each bus transmission takes **1 clock cycle**.

The asynchronous bus requires 40 ns per handshake.

The data portion of both buses is **32 bits wide**.

Question: Find the bandwidth for each bus when reading one word from a **200-ns memory**.

Answer:  *synchronous bus:*

the bus cycles is 50 ns. The steps and times required for the synchronous bus are follows:

1. *Send the address to memory : 50ns*
2. *Read the memory : 200ns*
3. *Send the data to the device : 50ns*

Thus, the total time is 300 ns. So,
the bandwidth = 4bytes/300ns = 4MB/0.3seconds
= 13.3MB/second

❧ *asynchronous bus:*

If we look carefully at the right Figure, we realize that several of the steps can be overlapped with the memory access time.

In particular, the memory receives the address at the end of step 1 and does not need to put the data on the bus until the beginning of step 5; steps 2, 3, and 4 can overlap with the memory access time.

This leads to the following timing:

step1: 40ns

step2,3,4: maximum($3 \times 40ns$, $200ns$) = $200ns$

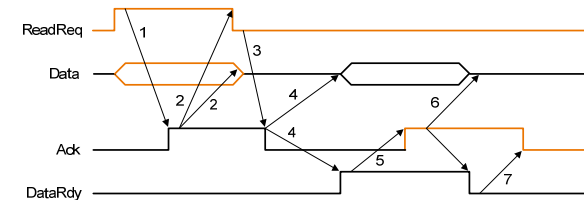
step5,6,7: $3 \times 40ns = 120ns$

Thus, the total time is 360ns, so

the maximum bandwidth = $4bytes/360ns = 4MB/0.36seconds$

= $11.1MB/second$

Accordingly, the synchronous bus is only about 20% faster.
(Why?)



Increasing the Bus Bandwidth

- ∞ Increasing data bus width
- ∞ Use separate address and data lines
- ∞ transfer multiple words

Performance Analysis of Two Synchronous Bus Schemes.

Suppose we have a system with the following characteristic:

1. A memory and bus system supporting block access of 4 to 16 32-bit words
2. A 64-bit synchronous bus clocked at 200 MHz, with each 64-bit transfer taking 1 clock cycle, and 1 clock cycle required to send an address to memory.
3. Two clock cycles needed between each bus operation.
4. A memory access time for the first four words of 200ns; each additional set of four words can be read in 20 ns. Assume that a bus transfer of the most recently read data and a read of the next four words can be overlapped.

Find the sustained bandwidth and the latency for a read of 256 words for transfers that use 4-word blocks and for transfers that use 16-word blocks. Also compute effective number of bus transactions per second for each case.

Answer:

64-bit synchronous bus

☞ *The 4-word block transfers:*

each block takes

- 1. 1 clock cycle to send the address to memory*
- 2. $200\text{ns}/(5\text{ns}/\text{cycle}) = 40$ clock cycles to read memory*
- 3. 2 clock cycles to send the data from the memory*
- 4. Two clock cycles needed between each bus operation.*

This is a total of 45cycles.

There are $256/4 = 64$ blocks.

So the transfer of 256 words takes

$$45 \times 64 = 2880 \text{ clock cycles}$$

The latency for the transfer of 256 words is:

$$2880 \text{ cycles} \times 5\text{ns}/\text{cycle} = 14,400\text{ns}.$$

so the number of bus transactions per second is:

$$64 \text{ transactions} \times \frac{1 \text{ second}}{14,400 \text{ ns}} = 4.44 \text{M transactions/second}$$

The bus bandwidth is:

$$(256 \times 4) \text{ bytes} \times \frac{1 \text{ second}}{14,400 \text{ ns}} = 71.11 \text{ MB/sec}$$

❧ *the 16-word block transfers:*

the first block requires

- 1. 1 clock cycle to send an address to memory*
- 2. 200ns or 40 cycles to read the first four words in memory.*
- 3. 2 cycles to transfer the data of the set, during which time the read of the next 4-word set is started.*
- 4. It only takes 20ns or 4 cycles to read the next set.
After the read is completed, the set will be transferred.
Each of the three remaining sets requires repeating only the last two steps.*
- 5. Two clock cycles needed between each bus operation.*

Thus, the total number of cycles for each 16- word block is:

$$1+40+4 \times 3+2+2=57 \text{ cycles.}$$

There are $256/16=16$ blocks.

so the transfer of 256 words takes $57 \times 16=912$ cycles.

Thus the latency is:

$$912 \text{ cycles} \times 5 \text{ ns/cycles} = 4560 \text{ ns.}$$

The number of bus transactions per second with 16-word blocks is:

$$16 \text{ transactions} \times \frac{1 \text{ second}}{4560 \text{ ns}} = 3.51 \text{M transactions/second}$$

The bus bandwidth with 16-word blocks is:

$$(256 \times 4) \text{ bytes} \times \frac{1 \text{ second}}{4560 \text{ ns}} = 224.56 \text{ MB/sec}$$

Now, let's put two bus bandwidth together:

4-word blocks: 71.11 MB/sec

16-word blocks: 224.56 MB/sec

The bandwidth for the 16-word blocks is 3.16 times higher than for the 4-word blocks.

****Overhead of Polling in an I/O System**

Assume: that the number of clock cycles for a polling operation is 400 and that processor executes with a 500-Mhz clock.

Determine the fraction of CPU time consumed for the mouse, floppy disk, and hard disk.

We assuming that you poll often enough so that no data is ever lost and that those devices are potentially always busy.

We assume again that:

- 1. The mouse must be polled 30 times per second to ensure that we do not miss any movement made by the user.*
- 2. The floppy disk transfers data to the processor in 16-bit units and has a data rate of 50 KB/sec. No data transfer can be missed.*
- 3. The hard disk transfers data in four-word chunks and can transfer at 4 MB/sec. Again, no transfer can be missed.*

Answer:

✎ *the mouse:*

**clock cycles per second for polling :
 $30 \times 400 = 12,000$ cycles**

Fraction of the processor clock cycles consumed is

$$\frac{12 \times 10^3}{500 \times 10^6} = 0.002\%$$

✎ *the floppy disk:*

**the number of polling access per second:
 $50KB/2B = 25K$**

**clock cycles per second for polling: $25K \times 400$ cycles
Fraction of the processor clock cycles consumed:**

$$\frac{10 \times 10^6}{500 \times 10^6} = 2\%$$

✎ *The hard disk:*

The number of polling access per second :

$$4MB/16B = 250K$$

Clock cycles per second for polling = $250K \times 400$

Fraction of the processor clock cycles consumed:

$$\frac{100 \times 10^6}{500 \times 10^6} = 20\%$$

Now, let's put three fractions together:

Mouse: 0.002%

Floppy disk: 2%

Hard disk: 20%

Clearly, polling can be used for the mouse without much performance impact on the processor, but it is unacceptable for a hard disk on this machine.

Overhead of Interrupt-Driven I/O

Suppose we have the same hard disk and processor we used in the former example, but we used interrupt-driven I/O. The overhead for each transfer, including the interrupt, is **500** clock cycles. Find the fraction of the processor consumed if the hard disk is only transferring data **5%** of the time.

Answer: First, we assume the disk is **transferring data 100%** of the time. So, the interrupt rate is the same as the polling rate.

Cycles per second for disk is:

$$250K \times 500 = 125 \times 10^6 \text{ cycles per second}$$

Fraction of the processor consumed during a transfer is:

$$\frac{125 \times 10^6}{500 \times 10^6} = 25\%$$

Now, we assume that the disk is only transferring data 5% of the time. The fraction of the processor time consumed on average is:

$$25\% \times 5\% = 1.25\%$$

As we can see, no CPU time is needed when an interrupt-driven I/O device is not actually transferring. This is the major advantage of an interrupt-driven interface versus polling.

Overhead of I/O Using DMA

Suppose we have the same hard disk and processor we used in the former example.

Assume that the initial setup of a DMA transfer takes **1000** clock cycles for the processor, and assume the handling of the interrupt at DMA completion requires **500** clock cycles for the processor.

The hard disk has a transfer rate of **4MB/sec** and uses DMA. The average transfer from disk is **8 KB**. Assume the disk is actively transferring 100% of the time.

Please find what fraction of the processor time is consumed.

Answer:

Time for each 8KB transfer is:

$$8KB/(4MB/second)=2 \times 10^{-3} \text{seconds.}$$

It requires the following cycles per second:

$$\frac{1000+500 \frac{\text{cycles}}{\text{transfer}}}{2 \times 10^{-3} \frac{\text{seconds}}{\text{transfer}}} = 750 \times 10^3 \frac{\text{clock cycles}}{\text{second}}$$

Fraction of processor time: $\frac{750 \times 10^3}{500 \times 10^6} = 0.2\%$

Unlike either polling or interrupt-driven I/O, DMA can be used to interface a hard disk without consuming all the processor cycles for a single I/O.

6.7 Designing an I/O system

The general approaches to designing I/O system

- ❧ Find the **weakest** link in the I/O system, which is the component in the I/O path that will constrain the design. Both the workload and configuration limits may dictate where the weakest link is located.
- ❧ Configure this component to sustain the required bandwidth.
- ❧ Determine the requirements for the rest of the system and configure them to support this bandwidth.

❖ I/O System Design

Examples:


Consider the following computer system:

1. A CPU sustains **3 billion instructions per second** and it takes average **100,000 instructions** in the operating system per I/O operation.
2. A memory backplane bus is capable of sustaining a transfer rate of **1000 MB/sec**.
3. SCSI-Ultra320 controllers with a transfer rate of **320 MB/sec** and accommodating up to **7 disks**.
4. Disk drives with a read/write bandwidth of **75 MB/sec** and an average seek plus rotational latency of **6 ms**.

If the workload consists of 64-KB reads (assuming the data block is sequential on a track), and the user program need 200,000 instructions per I/O operation, **please find the maximum sustainable I/O rate and the number of disks and SCSI controllers required.**

Answer:

The two fixed component of the system are the memory bus and the CPU. Let's first find the I/O rate that these two components can sustain and determine which of these is the **bottleneck.**

$$\begin{aligned}
 \text{Maximum I/O rate of CPU} &= \frac{\text{Instruction execution rate}}{\text{Instruction per I/O}} \\
 &= \frac{3 \times 10^9}{(200 + 100) \times 10^3} = 10000 \frac{\text{I/Os}}{\text{seconds}} \\
 \text{Maximum I/O rate of bus} &= \frac{\text{Bus bandwidth}}{\text{Bytes per I/O}} = \frac{1000 \times 10^6}{64 \times 10^3} = 15625 \frac{\text{I/Os}}{\text{seconds}}
 \end{aligned}$$


The CPU is the **bottleneck**, so we can now configure the rest of the system to perform at the level dictated by the CPU, 10000 I/Os per second.

Now, let's determine how many disks we need to be able to Accommodate **10000 I/Os** per second. To find the number of disks, we first find the time per I/O operation at the disk:

$$\begin{aligned} \text{Time per I/O at disk} &= \text{Seek/rotational time} + \text{Transfer time} \\ &= 6 \text{ ms} + \frac{64\text{KB}}{75\text{MB/sec}} = 6.9 \text{ ms} \end{aligned}$$

This means each disk can complete 1000ms/6.9ms, or 146 I/Os per second. To saturate the bus, the system need 10000/146≈69 disks.

To compute the number of SCSI buses, we need to know the average transfer rate per disk, which is given by:

$$\begin{aligned} \text{Transfer rate} &= \frac{\text{Transfer size}}{\text{Transfer time}} = \frac{64KB}{6.9ms} \approx 9.56MB/sec \\ 7 \times 9.56MB/sec &< 320MB/s \\ 69 \div 7 &\approx 10 \end{aligned}$$

Assuming the disk accesses are not clustered so that we can use all the bus bandwidth, we can place 7 disks per bus and controller. This means we will need 69/7, or 10 SCSI buses and controllers.

End

计算题1

已知一硬盘单个盘片，有磁道**2048**个，硬盘每磁道分为**64**扇区，每扇区大小为**4KB**。

已知磁盘平均寻道时间为**6ms**，最大寻道时间**8ms**，最小寻道时间**4ms**。控制时间为**0.5ms**，转速**15000RPM**，IDE接口传输速率为**133MB/s**。

问1) 磁盘容量

解答： $2048 * 64 * 4KB = 512MB$

每个磁道数据= $64 * 4K = 256KB$

有一个游戏连连看执行程序，大小为1.78 MB。连续存放在硬盘0磁道位置起始，点击连连看把执行程序转载(loader)到内存。

问2) 所需要的时间、带宽、吞吐率

1.78M/4KB=445个扇区，需要445/64=7个磁道，其中6个磁道满扇区共计384，还有一个磁道61扇区

相邻磁道寻道时间=(最大时间-最小时间)/磁道数=(8ms-4ms)/2K=0.002ms

一圈时间=60s/15000RPM=0.004ms；半圈0.002ms

读取连连看的时间=控制时间+(0磁道寻道时间+半圈时间+读取1圈的所有扇区)+(0磁道换到1磁道时间+半圈时间+读取1圈的所有扇区)+...+(5磁道换到6磁道时间+半圈时间+读取61/64圈的扇区)+传输时间

=0.5ms+(4ms+0.002ms+0.004ms)+

(0.002ms+0.002ms+0.004ms)*5+(0.002ms+0.002ms+0.004ms*61/63)+1.78/133

=0.5ms+4.006ms+0.040ms+0.008+13.383ms=17.937ms

17.937ms读取了1.78M数据，带宽为一秒能传输的数据量

=1.78MB/17.937ms=99.2MBps=793.9Mbps

17.937ms读取了一个文件，即进行了一次IO操作，吞吐率为1秒能进行几次IO操作，因此吞吐率=1/17.937=55.75(tps: 每秒事务数)

把该硬盘分为C/D两个逻辑分区，分别为1024个磁道。如果连连看不是连续存放，在连连看copy到D盘时候，D盘有很多碎片且随机分配在各个磁道的各个扇区。

问3) 连连看转载(loader)时间比前题慢多少倍?

第一次寻道的平均时间=寻到3/4道的的时间=7ms

在D盘的各个磁道之间的寻道范围在6~8ms，在磁道之间的平均寻道时间为1ms

需要读取**445**个扇区

一圈时间= $60s/15000RPM=0.004ms$; $1/64$ 圈= $0.0000625ms$

读取时间=控制时间+(第一次磁道寻道平均时间+半圈时间+读取1/64圈的扇区)+444*(磁道转换时间+半圈时间+读取1/64圈的扇区)+传输时间

$=0.5 + (7 + 0.002 + 0.0000625) + (1 + 0.002 + 0.0000625) * 444 + 1.78/133$

$=0.5 + 7.0020625 + 1.0020625 * 444 + 13.383ms$

$=465.8008125ms$

速度慢了 **$465.8ms/17.937ms=25.97$ 倍**

说明：文件存储位置越连续，速度越快

有两种储存方式：1、把该硬盘分为C/D两个逻辑分区，分别为1024个磁道。连连看连续存放在D盘起始位置；2、该硬盘分为一个逻辑分区C盘，前1/4磁盘用于存储windows，连连看连续存储在C盘1/4位置

问4) 问两个存储方式，转载(loader)时间的速度比？

前者

$$\begin{aligned} &0.5\text{ms} + (6\text{ms} + 0.002\text{ms} + 0.004\text{ms}) + (0.002\text{ms} + 0.002\text{ms} + 0.004\text{ms}) * 5 \\ &+ (0.002\text{ms} + 0.002\text{ms} + 0.004\text{ms} * 61/63) + 1.78/133 \\ &= 0.5 + 6.006 + 0.040 + 0.008 + 13.383\text{ms} = 19.937\text{ms} \end{aligned}$$

后者

$$\begin{aligned} &0.5\text{ms} + (5\text{ms} + 0.002\text{ms} + 0.004\text{ms}) + (0.002\text{ms} + 0.002\text{ms} + 0.004\text{ms}) * 5 + (0.002\text{ms} + 0.002\text{ms} + 0.004\text{ms} * 61/63) + 1.78/133 \\ &= 0.5 + 5.006 + 0.040 + 0.008 + 13.383\text{ms} = 18.937\text{ms} \end{aligned}$$

快1ms，就是寻道时间的差异

说明：越接近0磁道，速度越快

如果连连看放在IDE硬盘的0磁道；或者换成SCSI硬盘的0磁道，传输速率320MB/s；或者换成USB硬盘的0磁道，USB2.0全速模式的传输速率为12MB/s（U盘速度一般写入12M读取32M）

问5）三种硬件接口，各自的速度为？

IDE硬盘时间：其中传输耗时13.383

$$=0.5\text{ms}+(4\text{ms}+0.002\text{ms}+0.004\text{ms})+(0.002\text{ms}+0.002\text{ms}+0.004\text{ms})*5+(0.002\text{ms}+0.002\text{ms}+0.004\text{ms}*61/63)+1.78/133=17.937\text{ms}$$

SCSI硬盘时间：其中传输耗时5.5625

$$=0.5\text{ms}+(4\text{ms}+0.002\text{ms}+0.004\text{ms})+(0.002\text{ms}+0.002\text{ms}+0.004\text{ms})*5+(0.002\text{ms}+0.002\text{ms}+0.004\text{ms}*61/63)+1.78/320=10.1165\text{ms}$$

U盘时间：其中传输耗时148.333

$$0.5\text{ms}+(4\text{ms}+0.002\text{ms}+0.004\text{ms})+(0.002\text{ms}+0.002\text{ms}+0.004\text{ms})*5+(0.002\text{ms}+0.002\text{ms}+0.004\text{ms}*61/63)+1.78/12=152.887\text{ms}$$

U盘最慢，IDE居中，SCSI最快，耗时比例为：15：1.7：1

IDE硬盘，连连看存储在0磁道。如果连连看每过一关，就读取一个新的文件作为新一关的数据，该文件大小为512MB。

问6)： 怎么样的高手，其过关速度，使得计算机无法实现新的关卡数据装载。

从前述可知，读取512M数据，需要 $512\text{M}/256\text{K}=2\text{K}$ 磁道

控制时间+(0磁道寻道时间+半圈时间+读取1圈的所有扇区)+(0磁道换到1磁道时间+半圈时间+读取1圈的所有扇区) +...+(最后磁道换道时间+半圈时间+读取14圈的所有扇区)+传输时间

\approx 控制时间+ 0磁道寻道时间+ $2\text{K} \times (\text{换道时间} + \text{半圈时间} + \text{读取1圈的所有扇区})$ +传输时间

$=0.5+4+2\text{k} \times (0.002+0.002+0.004)+512\text{M}/133\text{MBps}$

$=0.5\text{ms}+4\text{ms}+16\text{ms}+3850\text{ms}=3.87\text{s}$

**如果高手在3.87秒内过关，计算机来不及转入新的场景数据
耗时最多是数据传输时间**

如前的IDE硬盘分为2个逻辑硬盘。已知理想CPI为1，1cc=1ns。L1 cache访问时间为1cc，命中率为99%，L2访问时间为2cc，命中率为95%。虚拟内存管理，内存页与物理扇区一样大。物理内存访问速度为20cc，辅存2M存放在C盘与放在D盘第一个磁道，虚拟内存命中率99.999%。

问7) C盘虚拟内存与D盘虚拟内存对CPI有多少影响

辅存2M=8个磁道；内存页4K

C盘读写时间=控制时间+ 寻道时间+半圈时间+传输时间

$$=0.5+4+0.002+4K/133M=0.5ms+4ms+0.002ms+0.03ms$$

$$=4.532ms=4.532*10^6ns$$

D盘读写时间=控制时间+ 寻道时间+半圈时间+传输时间

$$=0.5+6+0.002+4K/133M=0.5ms+6ms+0.002ms+0.03ms$$

$$=6.532ms=6.532*10^6ns$$

$$\text{虚拟内存在C盘的CPI}=1+1\%*(2+5\%*(20+0.001\%*4.532*10^6))=1+1\%*(2+5\%*(20+45.32))=1+1\%*(2+5\%*65.32)=1+1\%*(2+3.266)=1.05266$$

$$\text{虚拟内存在D盘的CPI}=1+1\%*(2+5\%*(20+0.001\%*6.532*10^6))=1+1\%*(2+5\%*(20+65.32))=1+1\%*(2+5\%*85.32)=1+1\%*(2+4.266)=1.06266$$

结论：虚拟内存放在C盘与D盘，对CPI影响差距为0.01（绝对值）

计算题2

已知一个软件流媒体广播软件，利用硬件上的**DMA**系统，把内存的流媒体数据送到网卡向外发送，数据量为**256MB**，时间长度为**90**分钟。

DMA通过中断与**CPU**通讯。**DMA**内含**32**位地址寄存器、**16**位大小寄存器以及其他控制器，初始化一次需要**1cc**。

内存一次可以读取**128bit**，读取一次花费**20cc**，分**4**次发送到总线上。

总线位宽为**32bit**。

问：这段视频发送期间对总线占有率

解答：计算总线占有率？

计算如果全速运行，把所有数据发送到网卡上，需要花费多少时间。然后计算时间比例。

流程：

- 1) **CPU初始化DMA→**
- 2) **DMA发送读取地址给内存→**
- 3) **内存读取128bit →**
- 4) **分4次发送到总线传输给网卡→**
- 5) **检查DMA大小是否完成，如果没有完成， goto 3**
- 6) **检查流媒体有没有传输完毕，如果没有完毕， goto 1**

因此首先计算**1-6**的循环次数

一次**DMA**初始化能传输多少数据

因为**DMA**内含**16**位大小寄存器，因此最大尺寸为 **$2^{16}=64\text{KB}$**

视频大小为**256MB**，因此需要做 **$256\text{M}/64\text{K}=4\text{K}$** 次**DMA**

然后计算**3-5**的循环次数：

一次**DMA**传输**64KB**，而内存一次**128bit=16B**

则（内存读，传输**4**次）在一次**DMA**内，循环
64KB/16B=4K次

最后做计算时间

=DMA次数* (CPU初始化DMA 时间+ DMA发送读取地址给内存时间+DMA内的循环次数* (内存读取时间+数据发送时间)

=4K* (1cc+1cc+4k* (20cc+4cc))

再用这个数据除以流媒体时间90分钟，就是这90分钟内总线的利用率

注意：

CPU初始化DMA 时间=1cc

DMA发送读取地址给内存时间=总线发送一次时间
=1cc

内存读取时间=读取128bit时间，根据题干为20cc

数据发送时间=发送数据除以32bit/cc (总线32bit位宽)
=4cc

题目变化

1、增加数据块**block**概念

要求每次**DMA**操作以数据块为单位进行。

例如，数据块为**512B**

DMA内的循环次数为 **$64\text{KB}/512\text{B}=128$** 次

每次循环内：

内存读取时间变成**4**次读**128bit**

每次数据传输变成**512cc**

显然，仅仅增加**block**不会改变传输时间

题目变化

2、增加总线位宽

例如：增加位宽到**128bit**

后果：

1) 传输地址时间有变化么？

无

2) 读内存速度有变化么？

无

3) 循环次数有变化么？

无

4) 每次数据传输变化么？

有，每次内存读**128bit**，之后**1cc**就发送到总线上了

题目变化

3、增加overlaped概念

一般做法是在一个**block**操作内，内存读出与数据发送到总线这两个操作，可以重叠执行。

要求**block**大于内存一次读出的数据，意味着一次**block**操作，会有多次内存读与内存发送到总线。

后果：

1) 传输地址时间有变化么？

无

2) 循环次数有变化么？

无

3) 每次数据传输变化么？

有，第一单位时间读内存；第二及以后单位时间做{发送第一单位的数据到总线，同时读取第二次内存}；最后一个单位时间要注意，没有读取内存操作

题目变化

4、增加目标设备的信息

例如：增加网卡的数据，网卡上有个缓冲区，大小为**16bit**，写入缓冲区就能马上发送到网络上。

后果：

每次数据发送到网卡，要等待一个**cc**，因为网卡**1cc**只能从**32bit**总线中选取**16bit**写入缓冲，下**1cc**写另外的**16bit**。因此需要通知发送方（内存），发送**32bit**后休息**1cc**

例如：网卡缓冲区**64bit**

没有上述问题，内存发送无需等待

例如：网卡缓冲区满，需要**1cc**发送到网络上
对应通知总线发送方

Problem 1

- Assume there is a disk which has one platter with 2048 tracks, and each track is divided into 64 sectors, the capacity of each sector is 4KB.
- Average seek time is 6ms, maximum seek time is 8ms and minimum seek time is 4ms. Controller time is 0.5ms. If the disk rotates at 15000RPM and IDE interface has a transfer rate of 133MB/s
- Question 1: disk capacity
 - Capacity of each track = $64 * 4\text{KB} = 256\text{KB}$
 - Disk capacity = $2048 * 64 * 4\text{KB} = 512\text{MB}$

There is a game program with size of 1.78 MB. It is stored at a consecutive addresses starting from track0 in a disk. It will be loaded into memory when the program is clicked.

Question 2) compute disk access time、bandwidth and throughput

1.78M/4KB=445 sectors, occupies 445/64=7 tracks, among which 6 tracks are full (384 sectors in total), and 61 sectors are occupied in the last track

Seek time for neighbor track=(maximum-minimum)/#tracks=(8ms-4ms)/2K=0.002ms

Rotation latency=60s/15000RPM/2=0.002ms;

Disk access time=**controller time**+(track0 seek time+rotation latency+disk read time for all sectors of track0)+(seek time from track0 to track1+rotation latency+disk read time for all sectors of track1) +...+(seek time from track5 to track6+ rotation latency+disk read time for 61 sectors of track6)+transfer time

$$\begin{aligned} &=0.5\text{ms}+(4\text{ms}+0.002\text{ms}+0.004\text{ms})+ \\ &\quad (0.002\text{ms}+0.002\text{ms}+0.004\text{ms})\times 5+(0.002\text{ms}+0.002\text{ms}+0.004\text{ms}\times 61/64)+1.78/133 \\ &=0.5\text{ms}+4.006\text{ms}+0.040\text{ms}+0.008+13.383\text{ms}=17.937\text{ms} \end{aligned}$$

read 1.78MB data in 17.937ms, **bandwidth**=1.78MB/17.937ms=99.2MBps=793.9Mbps

It takes 17.937ms to read a file, which is one IO operation, so

吞吐量throughput=1/17.937=55.75(transactions per second)