Computer Organization & Design

Hardware/Software interface

Lou Xueqing

玉泉校区曹光彪东楼507室

Website:10.214.47.99/index.php
Email: xqlou@zju.edu.cn
hzlou@163.com

浙江大学计算机学院&软件学院

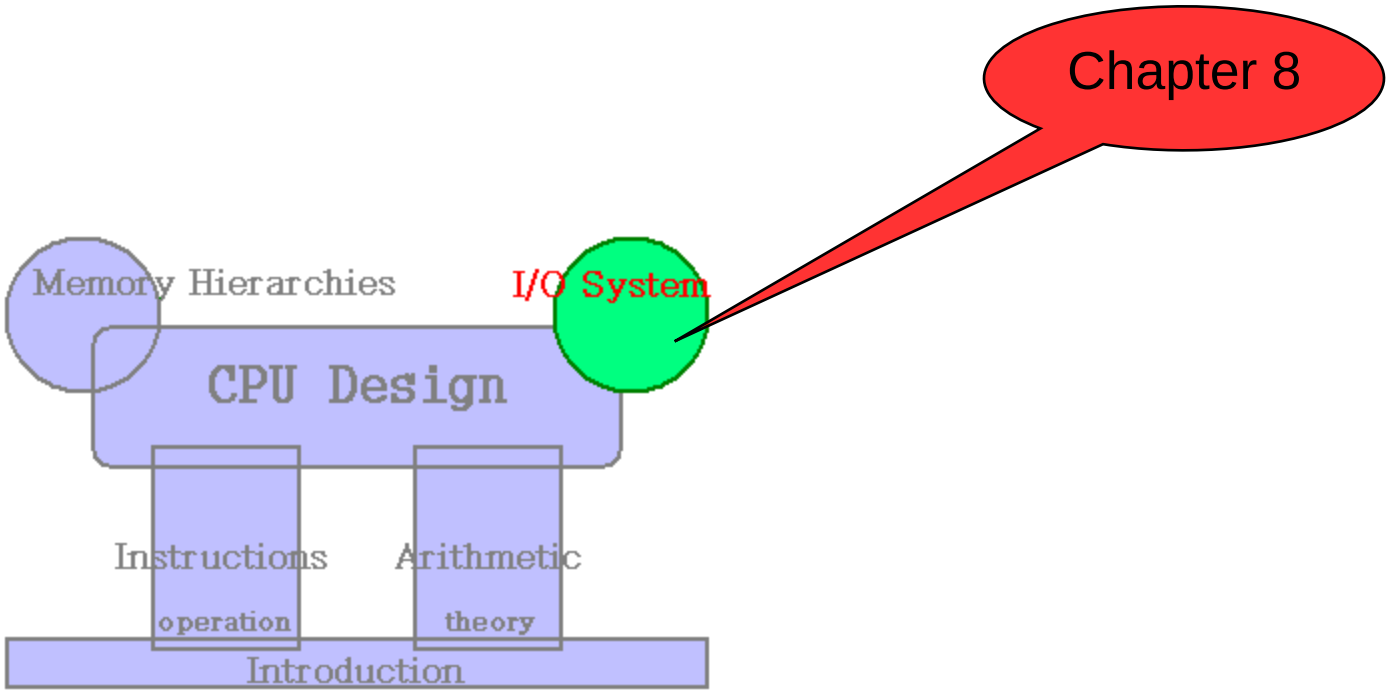
Chapter 8: Interfacing Processors and Peripherals

Lou Xueqing

Computer Organization & Design

Chapter 8

- Interfacing Processors and Peripherals



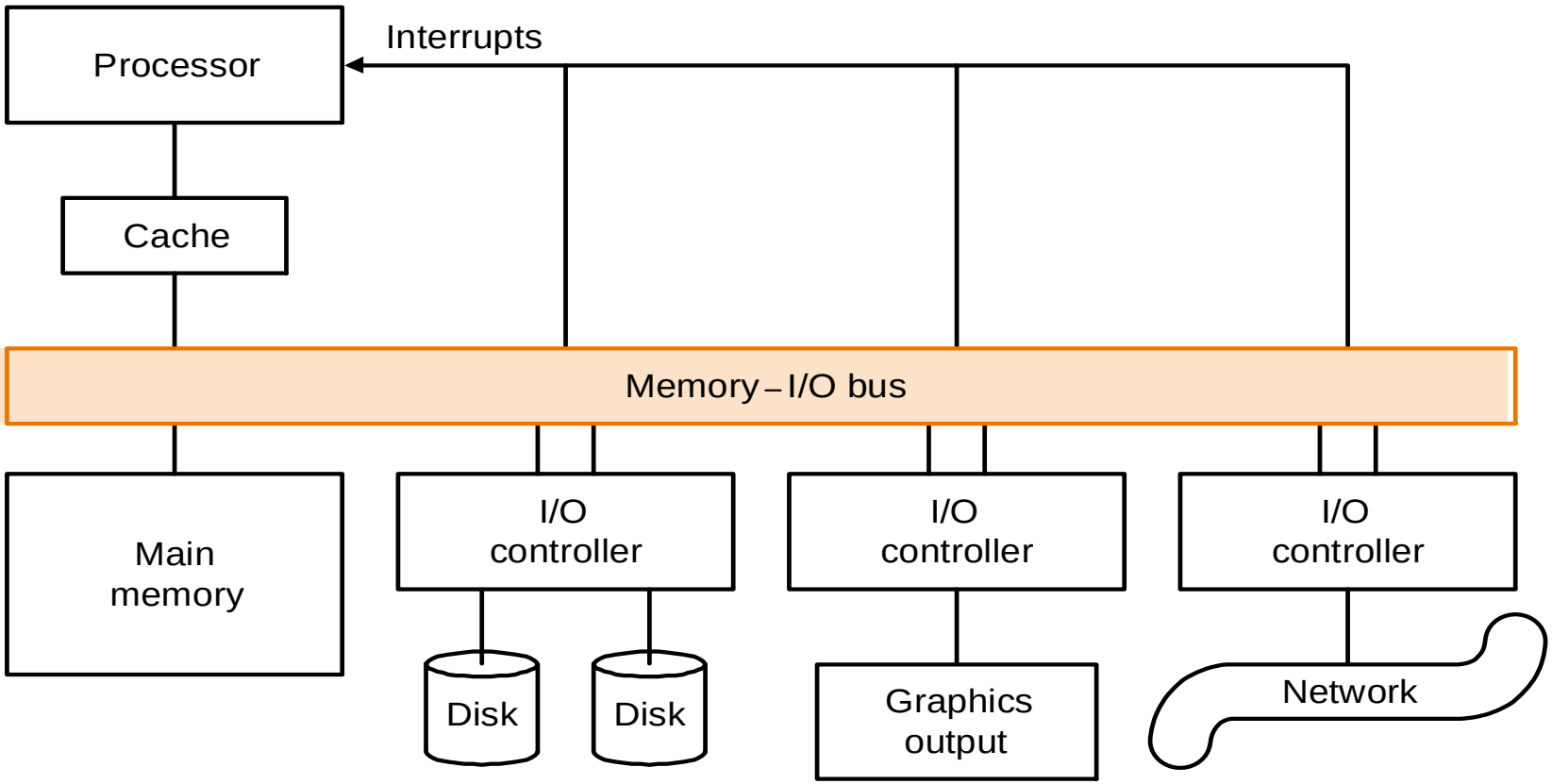
Content

- 8.1 Introduction
- 8.2 I/O Performance Measures: Some Examples from Disk and File Systems
- 8.3 Types and Characteristics of I/O Devices
- 8.4 Buses: Connecting I/O Devices to Processor and Memory
- 8.5 Interfacing I/O Devices to the Memory, Processor, and Operating System
- 8.6 Designing an I/O system
- 8.7 Real Stuff: A Typical Desktop I/O System

8.1 Introduction

- I/O Design affected by many factors (expandability, resilience)
- Performance:
  - access latency
  - throughput
  - connection between devices and the system
  - the memory hierarchy
  - the operating system
- A variety of different users (e.g., banks, engineers, supercomputers)

Typical collection of I/O devices



Important but neglected

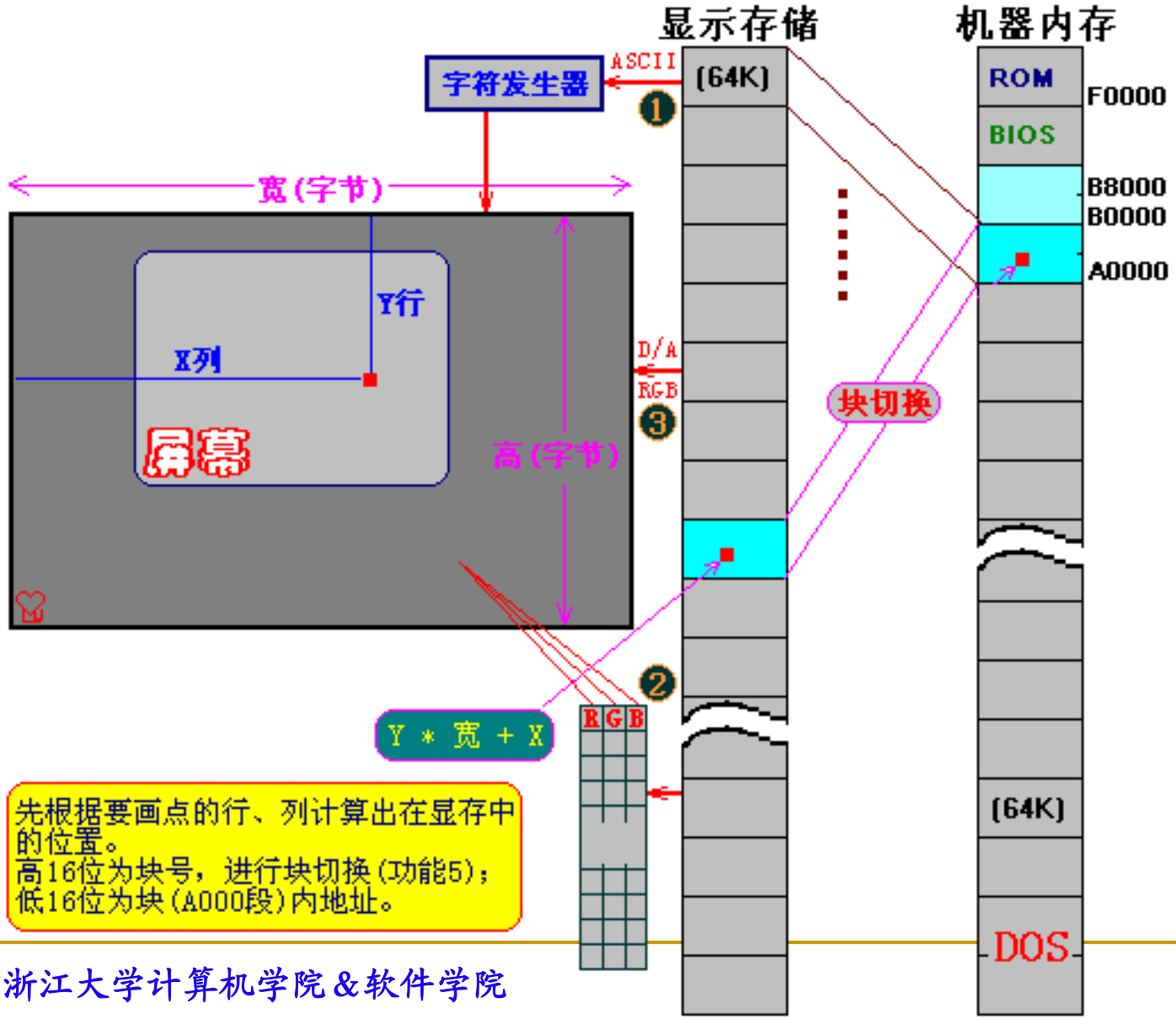
- “The difficulties in assessing and designing I/O systems have often relegated I/O to second class status”
- “courses in every aspect of computing, from programming to computer architecture often ignore I/O or give it scanty coverage”
- “textbooks leave the subject to near the end, making it easier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

- Amdahl's law remind us that ignoring I/O is dangerous
  - **e.g., a bench mark executes in 100 seconds of elapsed time , where 90 seconds is CPU time and the rest is I/O time. CPU time improves by 50% per year but I/O time doesn't improve. After five years, the improvement in CPU performance is 7.5 times, but the improvement in elapsed time is only 4.5 times.**
- measure I/O performance depends on the application
  - *throughput* (e.g., NITS)
  - *response time* (e.g., workstation and PC)
  - both *throughput* and *response time* (e.g., ATM)

8.2 I/O Performance Measures: Some 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

VESA


显示存储
机器内存
字符发生器
ASCII
(64K)
ROM
BIOS
F0000
B8000
B0000
A0000
宽(字节)
Y行
X列
屏幕
高(字节)
D/A
RGB
块切换
R G B
Y * 宽 + X
(64K)
DOS
先根据要画点的行、列计算出在显存中的位置。
高16位为块号，进行块切换(功能5)；
低16位为块(A000段)内地址。

8.3 Types and Characteristics of I/O Devices

- Very diverse devices
  - Behavior: Input, output, or storage.
  - Partner: Either a human or a machine is at the other end of the I/O device.
  - Data rate: The peak rate at which data can be transferred between the I/O device and the main memory or processor.
- e.g., keyboard is an input device used by a human with a peak data rate of about 10 bytes per second.

The diversity of I/O devices

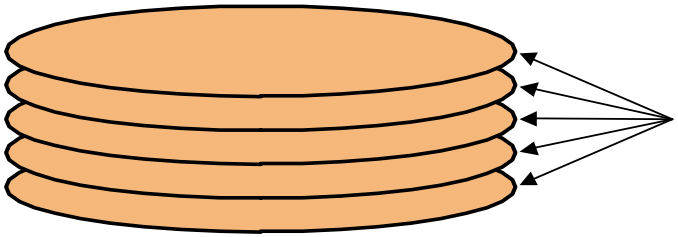
Device	Behavior	Partner	Data rate (KB/sec)
Keyboard	input	human	0.01
Mouse	input	human	0.02
Voice input	input	human	0.02
Scanner	input	human	400.00
Voice output	output	human	0.60
Line printer	output	human	1.00
Laser printer	output	human	200.00
Graphics display	output	human	60,000.00
Modem	input or output	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

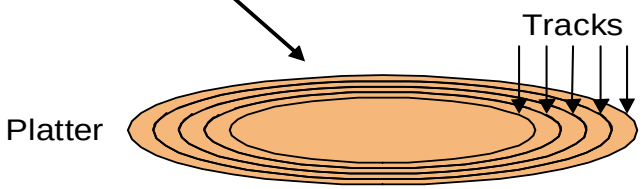
Mouse

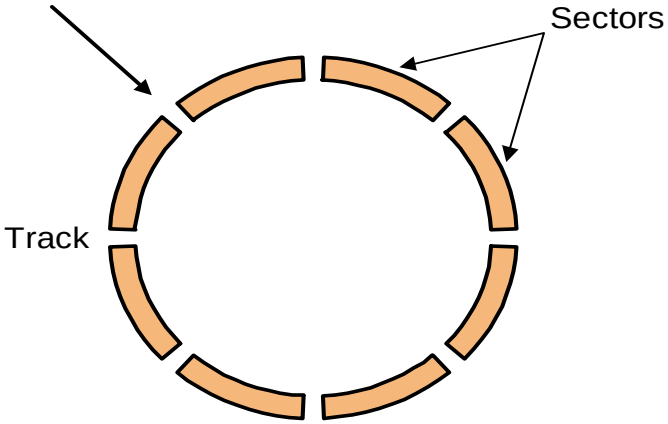
- an input device with a low data rate
- The mechanical version
  - *Moving the mouse rolls the large ball inside*
  - *The ball makes contact with an x-wheel and a y-wheel*
  - *Decide the distance and direction the mouse moves according to the rotation of wheels*
- The photoelectric version
  - *Better orientation and better precision*

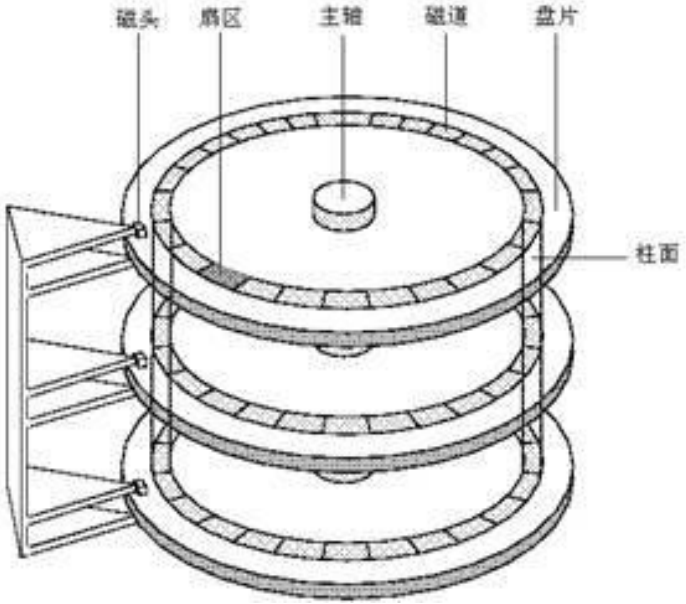
- two major types of magnetic disks
  - floppy disks
  - hard disks
    - **larger**
    - **higher density**
    - **higher data rate**
    - **more than one platter**
- the organization of hard disk
  - platters: disk consists of a collection of *platters*, each of which has two recordable disk surfaces
  - 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









- To access data of disk:
  - Seek: position read/write head over the proper track
    - **minimum seek time**
    - **maximum seek time**
    - **average seek time (8 to 20 ms)**
  - Rotational latency: wait for desired sector
    - **average latency is the half way around the disk.**

$$\text{Average rotational latency} = \frac{0.5 \text{ rotation}}{3600\text{RPM}} = \frac{0.5 \text{ rotation}}{3600\text{RPM} / \left(60 \frac{\text{seconds}}{\text{minute}}\right)}$$
$$= 0.0083 \text{ seconds} = 8.3 \text{ ms}$$

  - Transfer: grab the data (one or more sectors)
  - Disk controller, which control the transfer between the disk and the memory

Disk Read Time

Example: **What is the average time to read or write a 512-byte sector for a typical disk rotating at 5400 RPM? The advertised average seek time is 12 ms, the transfer rate is 5 MB/sec, and the controller overhead is 2 ms. Assume that the disk is idle so that there is no waiting time.**

Answer: **Average disk time is equal to *average seek time* + *average rotation delay* + *transfer time* + *controller overhead*. Using the advertised average seek time ,the answer is**

12ms + 5.6ms + 0.5KB/(5MB/sec) + 2ms = 19.7ms

If the measured average seek time is 25% of the advertised **average time, the answer is**

3ms + 5.6ms + 0.1ms + 2ms = 10.7ms

软盘结构

引导记录分布：

偏移	字长	描述
00	3	JMP到引导代码
03	8	OEM名字及版本
0B	2	每扇区字节数
0D	1	每簇扇区数（2的幂）
0E	2	保留扇区（为目录、FAT等）
10	1	FAT的拷贝数
11	2	根目录中最大允许目录项
13	2	总扇区数
15	1	磁盘介质描述
16	2	FAT的扇区数
18	2	每磁道的扇区数
1A	2	磁头数
1C	2	隐藏扇区数

扇区号	内容
1	BOOT区
2	FAT－1(9个扇区)
11	FAT－2(9个扇区)
20	目录区(12个扇区)
32～	（数据区）

文件分区表FAT：

- FAT表：每个盘有相同的两个FAT表。软盘一般为12位；硬盘以前为16位，现多为32位。FAT表的头两项为硬盘类型。

12位FAT表项值	16位FAT表项值	意义
000	0000	未用簇
001～FEF	0001～FFEF	下一簇号
FF0～FF6	FFF0～FFF6	保留
FF7	FFF7	坏簇
FF8～FFF	FFF8～FFFF	最后簇（文件结束）

文件目录：

FAT

- 目录项

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
文件名								扩展名			属性	**保留**										时间		日期		首簇		文件长				

- 属性字段

7	6	5	4	3	2	1	0	=1	=0
							X	只读文件	读/写文件
						X		隐藏文件	可显文件
					X			系统文件	常规文件
				X				卷名	
			X					子目录名	
		X						自上次备份后，文件有改动	文件无改动
X	X							**保留**	

Networks

- 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**
- Switched network ,which are common in long-haul networks

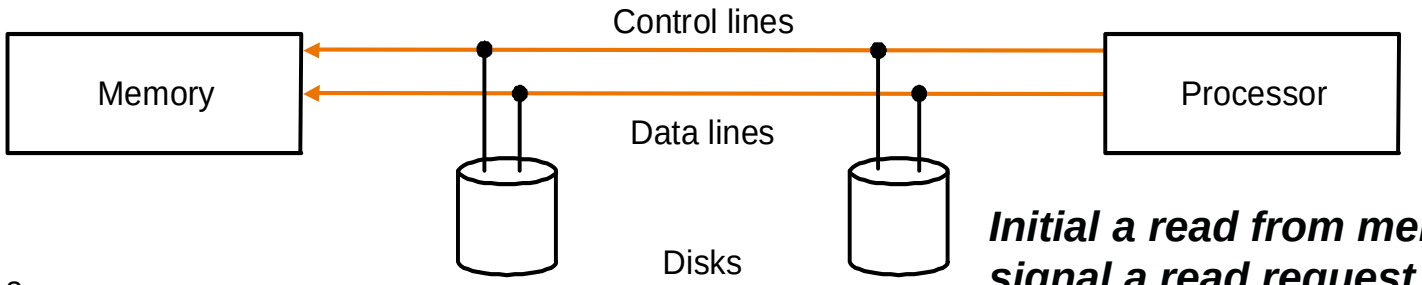
  e.g., ARPANET
- TCP/IP is the key to interconnecting different networks
- The bandwidths of networks are probably growing faster than the bandwidth of any other type of device at present.

8.4 Buses: Connecting I/O Devices to Processor and Memory

- Shared communication link (one or more wires)
- Difficult design:
  - may be bottleneck
  - length of the bus
  - number of devices
  - tradeoffs (buffers for higher bandwidth increases latency)
  - support for many different devices
  - cost

- A buts contain 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**

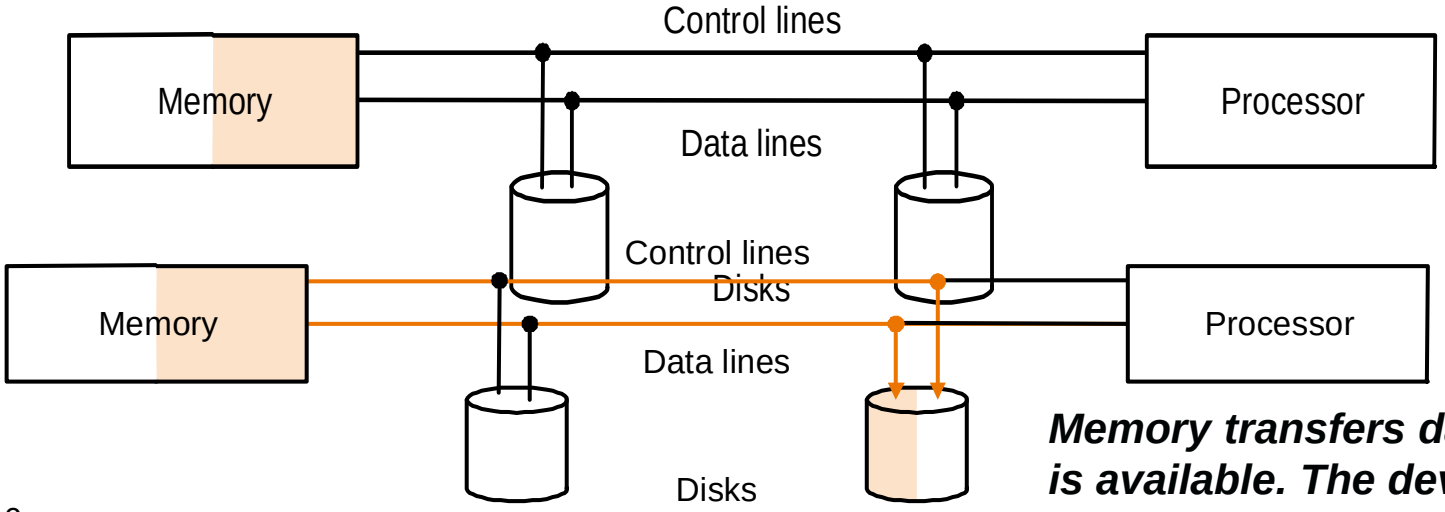
The steps of an output operation.



a.

Initial a read from memory. Control lines signal a read request to memory, while the data lines contain the address

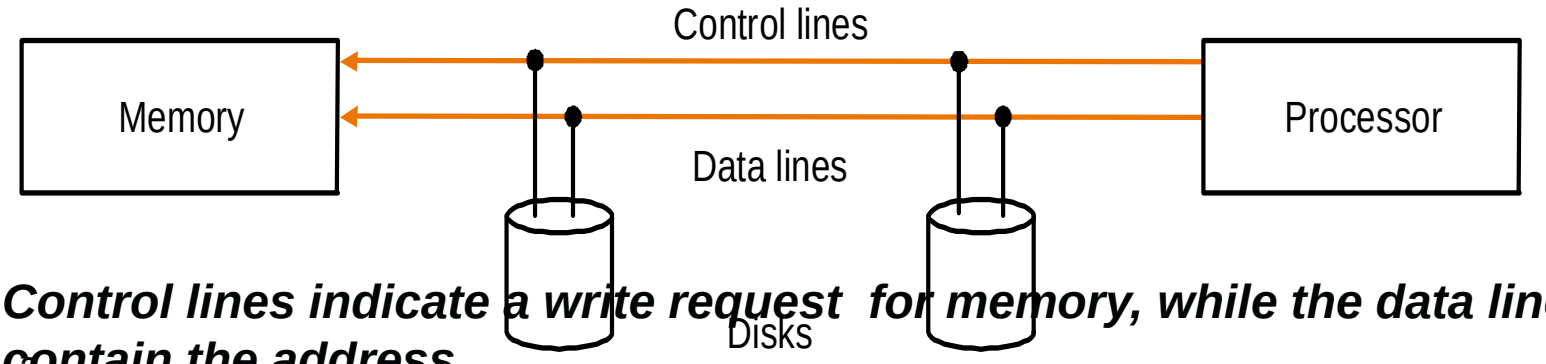
Memory access the data.



c.

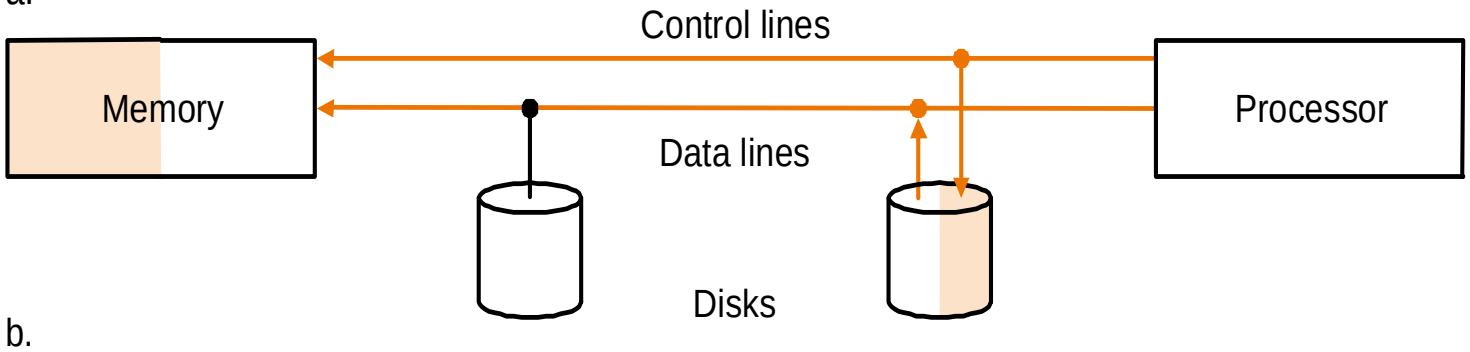
Memory transfers data and signal data is available. The device stores data as it appears on the bus.

The steps of an input operation.



a.

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

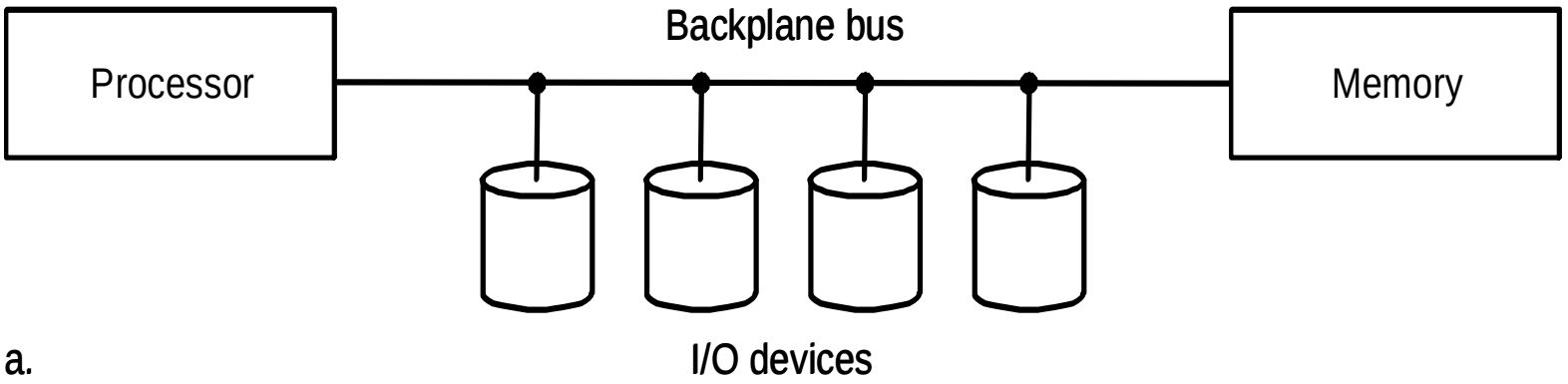


b.

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.

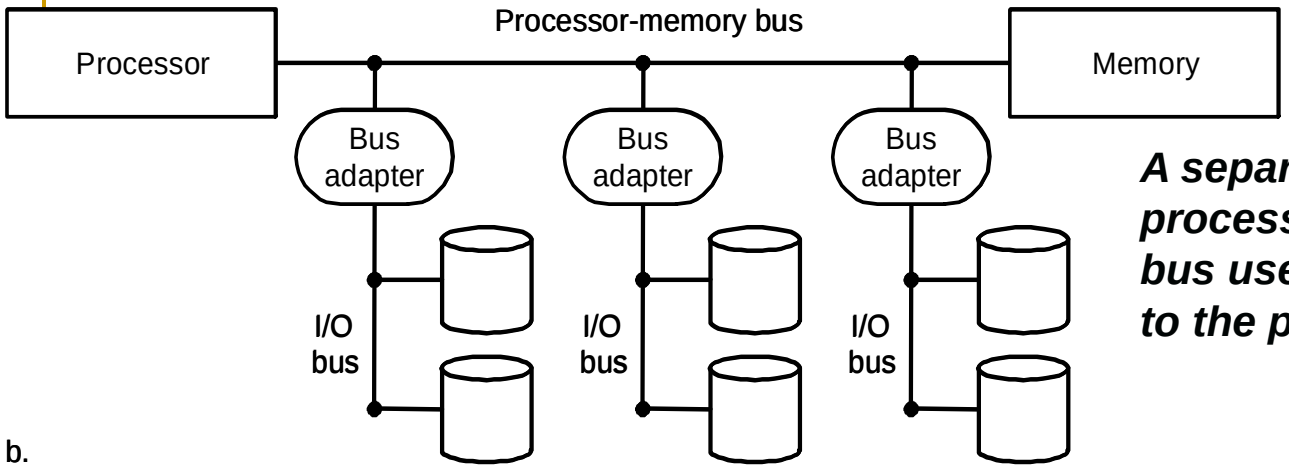
Types of buses:

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



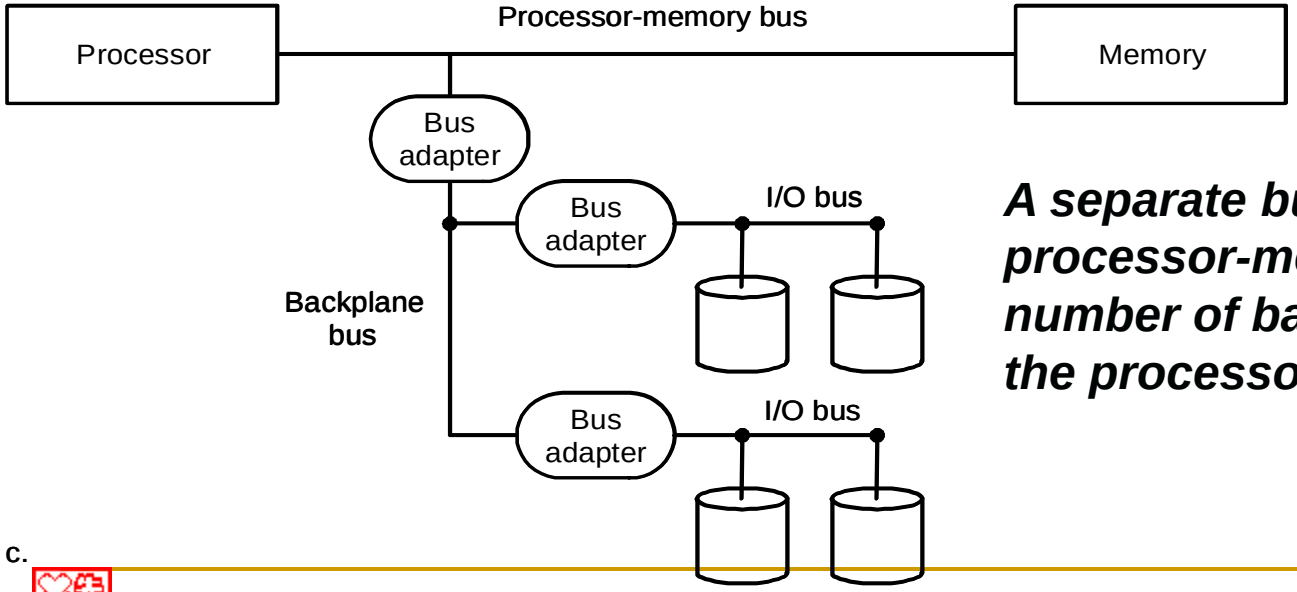
a.

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



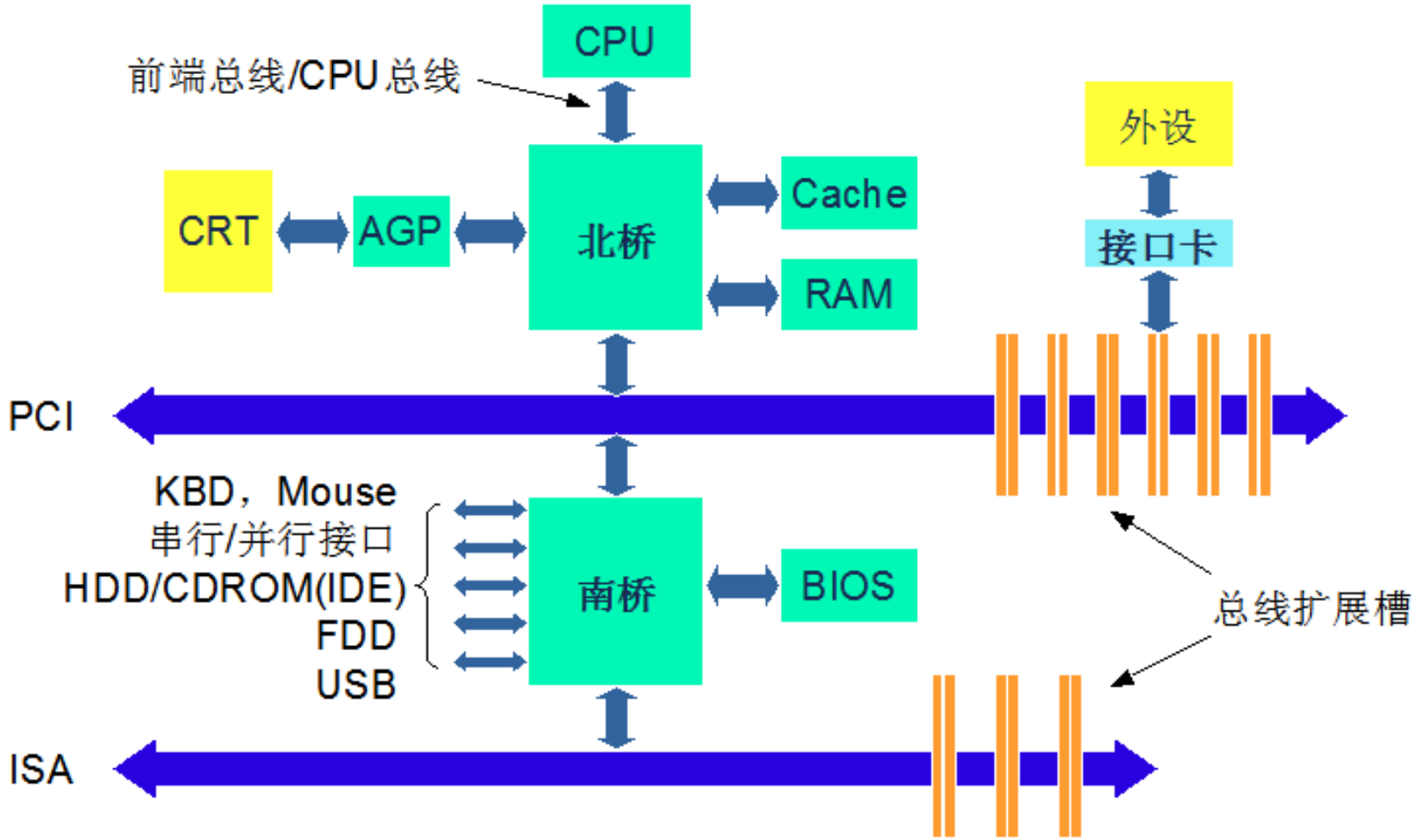
b.

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



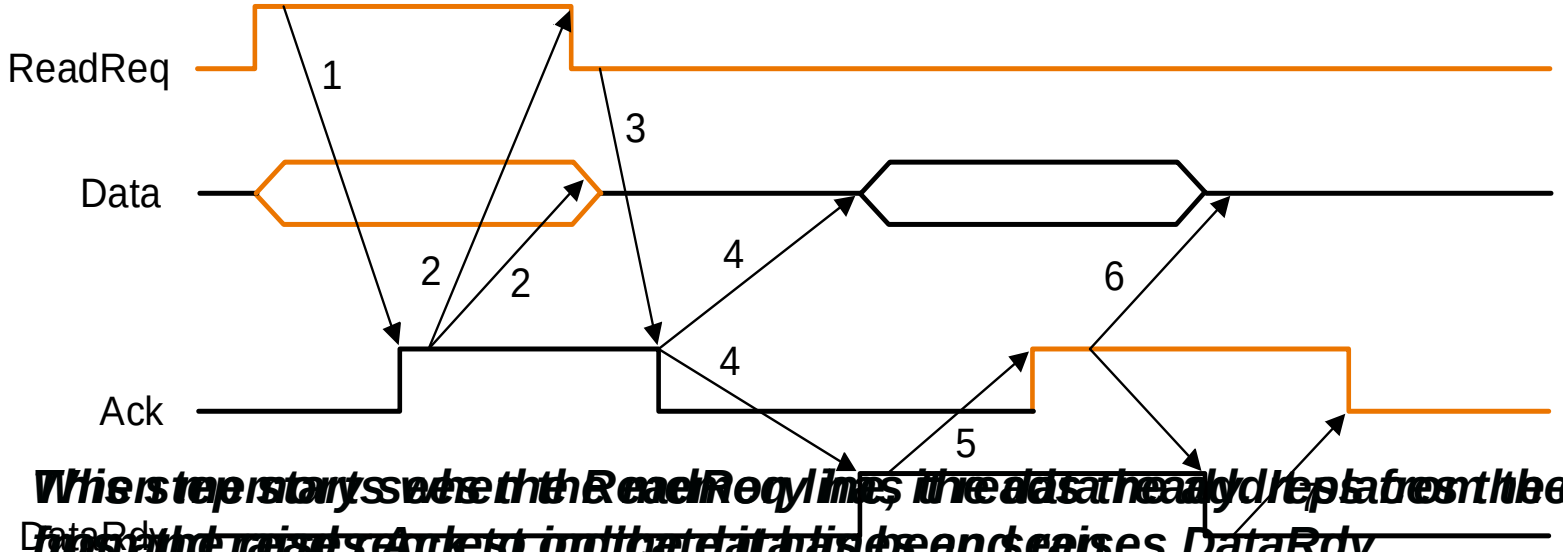
c.

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


CPU
前端总线/CPU总线
外设
CRT
AGP
北桥
Cache
RAM
接口卡
PCI
KBD，Mouse
串行/并行接口
HDD/CDROM(IDE)
FDD
USB
南桥
BIOS
总线扩展槽
ISA

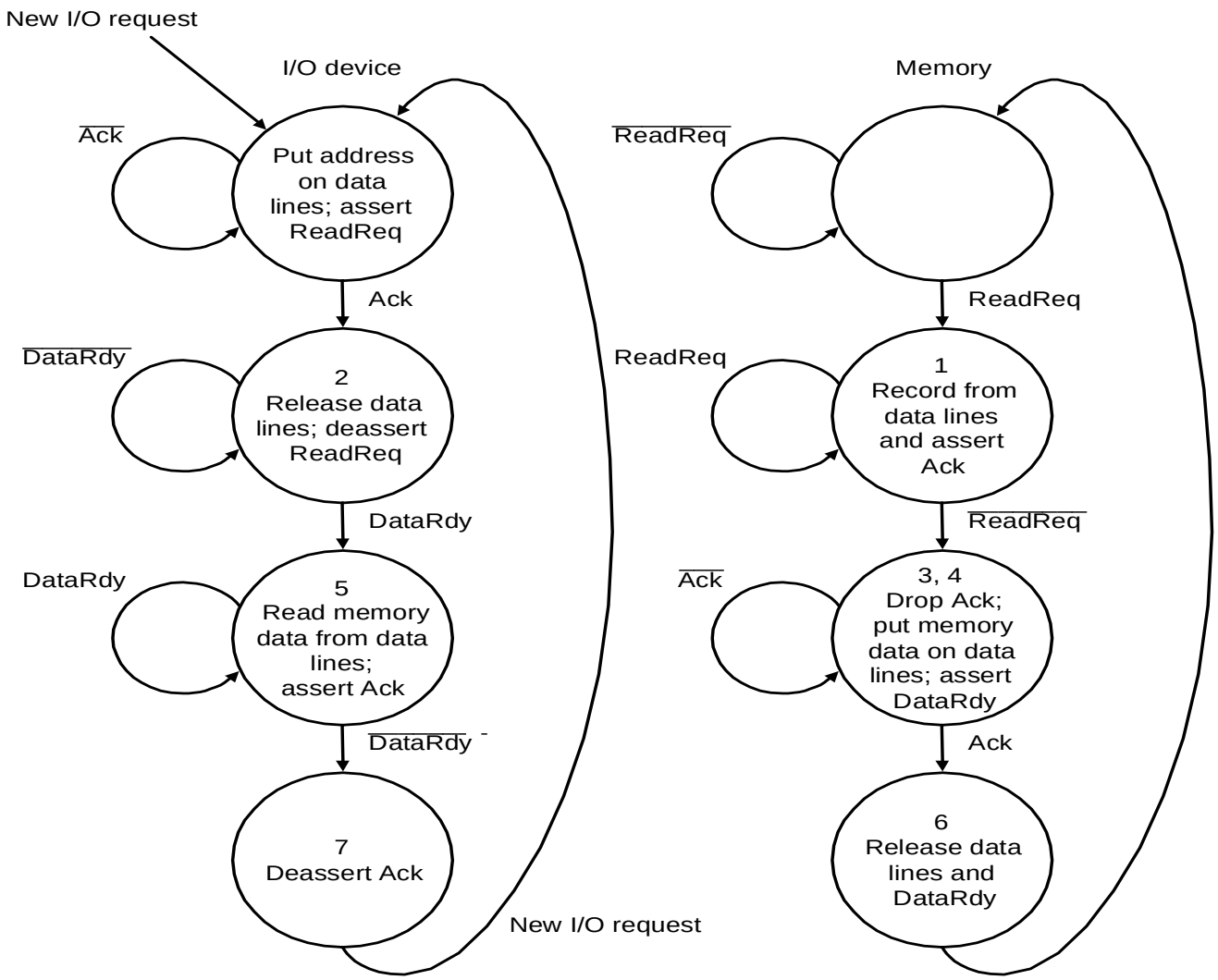
- Synchronous vs. Asynchronous
  - Synchronous bus use a clock and a synchronous protocol, fast and small but every device must operate at same rate and clock skew requires the bus to be short
  - Asynchronous bus don't use a clock and instead use *handshaking*
- Handshaking protocol
  - Our example ,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 the I/O device sees the DataRdy line high, it raises the ReadReq signal to inform memory that it wants to read a word.*
2. *The I/O device sees the DataRdy, reads the data from the data lines, and signals that it has the data by raising ACK.*
3. *Memory sees that ReadReq is low and drops the Ack line to acknowledge the ReadReq signal.*
6. *The ReadReq and 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.



Performance analysis of Synchronous versus Asynchronous buses

Example: **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. 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 Figure 8.10 in the text, 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 1and

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(3×40ns,200ns)=200ns

step5,6,7: 3×40ns=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.

Increasing the Bus Bandwidth

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

Performance Analysis of Two Bus Schemes

Examples: **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. Assume the bus is idle before an access.**
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: ❑ *the 4-word block transfers:*

each block takes

1. *1 clock cycle to send the address to memory*
2. *200ns/(5ns/cycle) = 40 clock cycles to read memory*
3. *2 clock cycles to send the data from the memory*
4. *2 idle clock cycles between this transfer and the next*

This is a total of *45cycles*, and *256/4 = 64transactions* are needed, so the entire transfer takes $45 \times 64 = 2880$ *clock cycles* The latency is *2880 cycles* $\times$ *5ns/cycle = 14,400ns*.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 send the data of the block, during which time the read of the four words in the next block is started*
4. *2 idle cycles between transfers and during which the read of the next block is completed*

Each of the three remaining 4-word blocks requires repeating only the last two steps.

Thus, the total number of cycles for each 16- word block is *1+40+4*$\times$*(2+2)=57 cycles*, and 256/16 transactions are needed, so the entire transfer takes *57*$\times$*16=912 cycles*. Thus the latency is *912cycles*$\times$*5ns/cycles = 4560ns.*

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

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

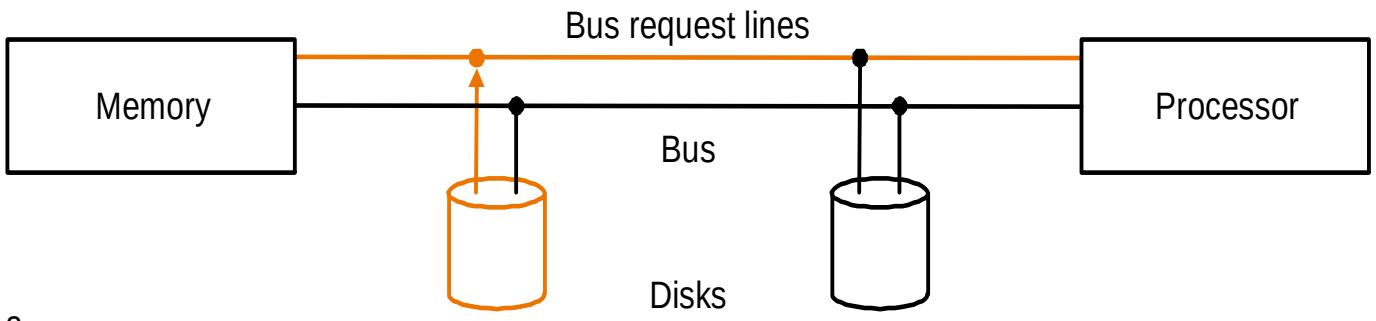
The bus bandwidth with 16-word blocks is

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

Which is 3.16 times higher than for the 4-word blocks. The advantage of using large block transfers is clear.

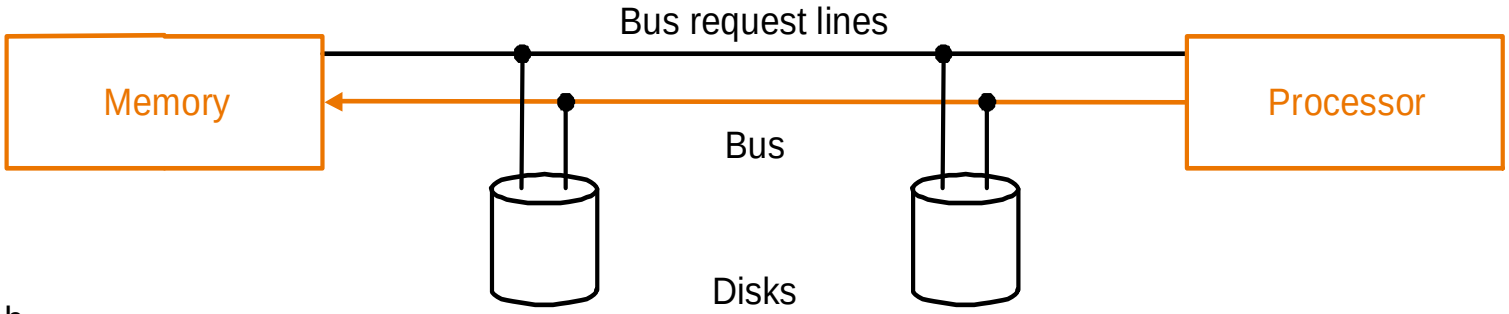
Obtaining Access to the Bus

- "Without any control, multiple device desiring to communicate could each try to assert the control and data lines for different transfers!"
  - 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).



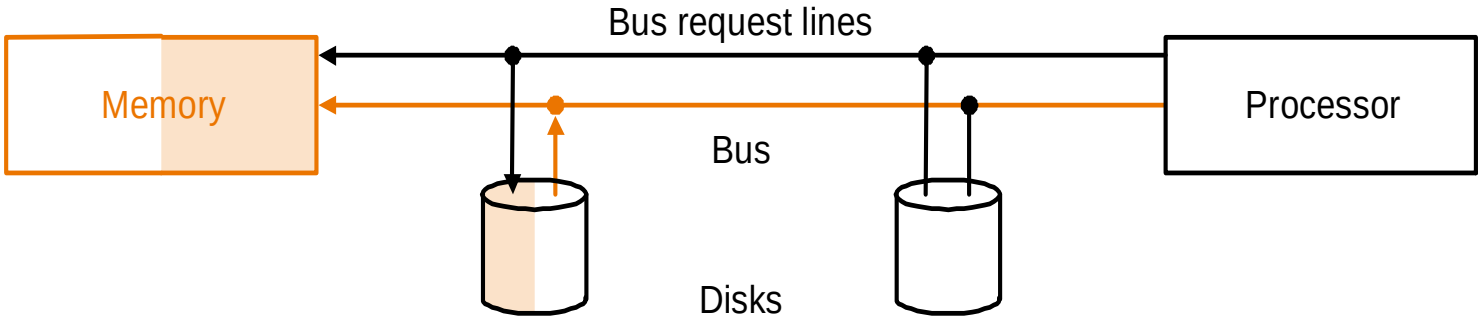
a.

First, the device generates a bus request to indicate to the processor that it wants to use the bus.



b.

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



c.

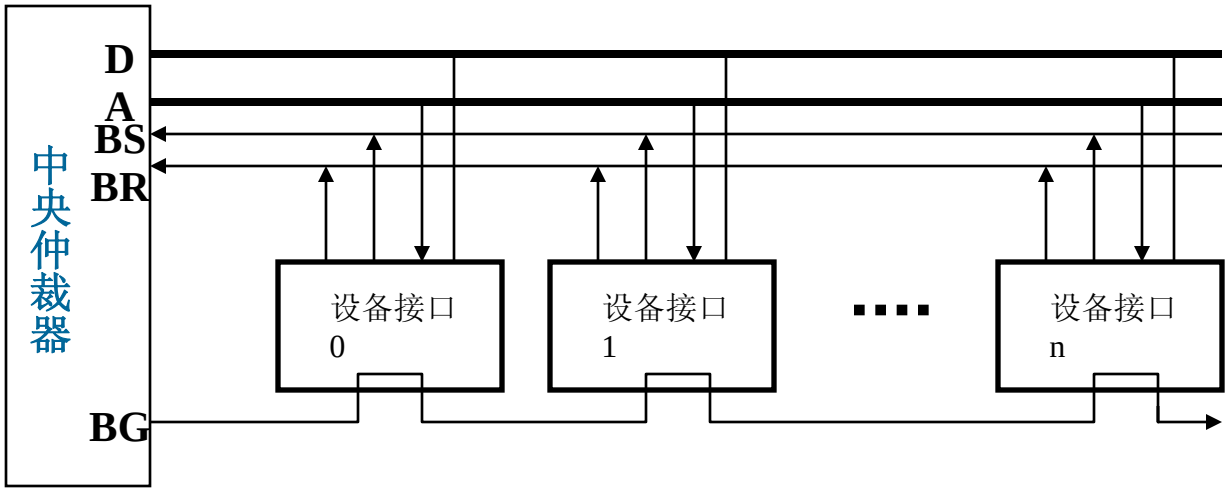
The processor also notifies the device that its bus request is being processed; as a result, the device knows it can use the bus and places the address for the request on the bus.

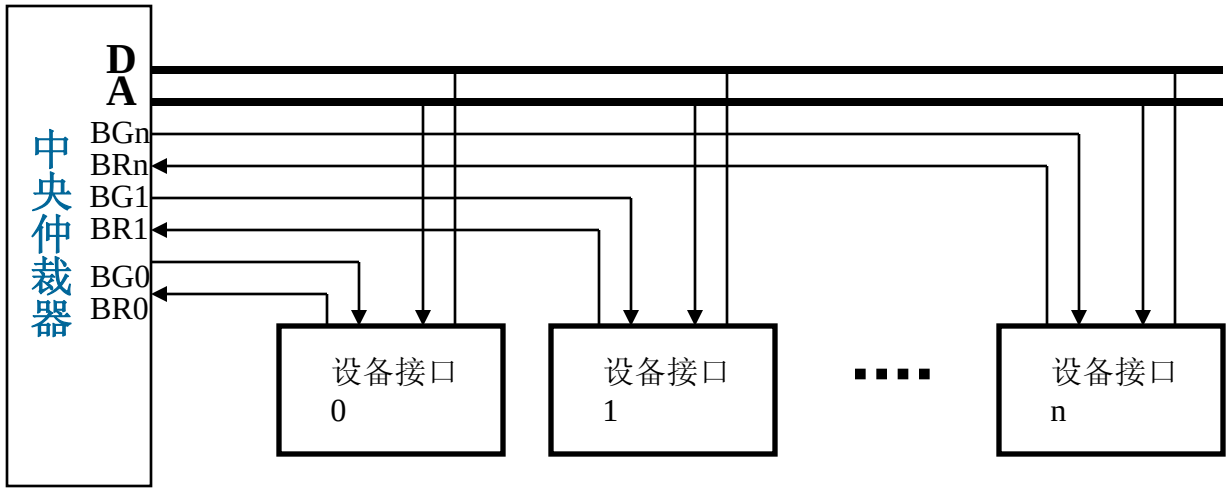
Bus Arbitration

- Deciding which bus master 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)***
  - ***centralized, parallel arbitration (requires an arbiter), e.g., PCI***
  - ***self selection, e.g., NuBus used in Macintosh***
  - ***collision detection, e.g., Ethernet***

Bus Standards

- SCSI *(small computer system interface)*
- PCI *(peripheral component interconnect)*


中央仲裁器
D
A
BS
BR
BG
设备接口
0
设备接口
1
设备接口
n

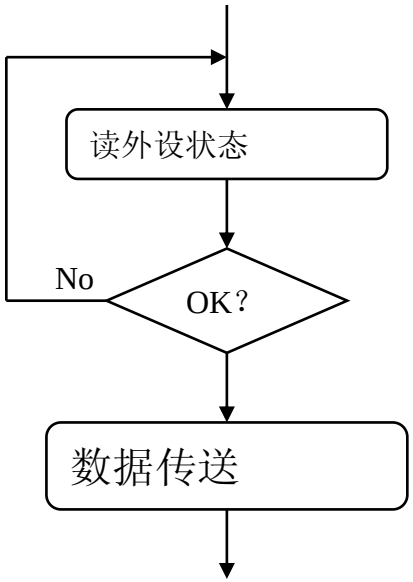

D
A
中央仲裁器
BGn
BRn
BG1
BR1
BG0
BR0
设备接口
0
设备接口
1
设备接口
n

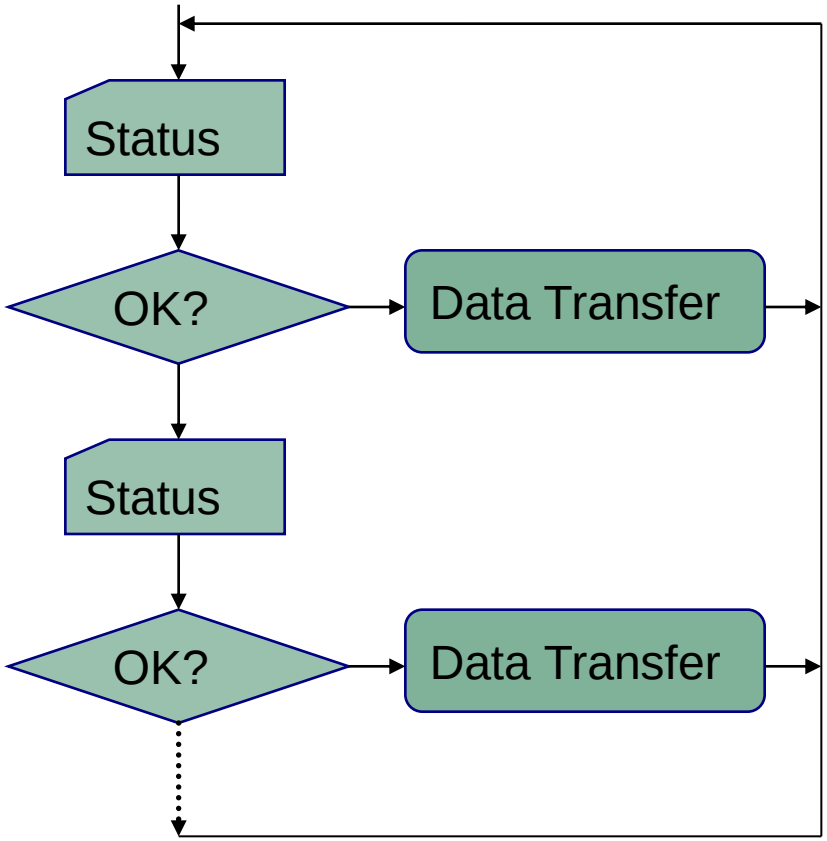
8.5 Interfacing I/O Devices to the Memory, Processor, and Operating System

- 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 an 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***

- Two methods used to address the device:
  - memory-mapped I/O, portions of the address space are assigned to I/O devices.
  - special I/O instructions
- Communication with the Processor
  - 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

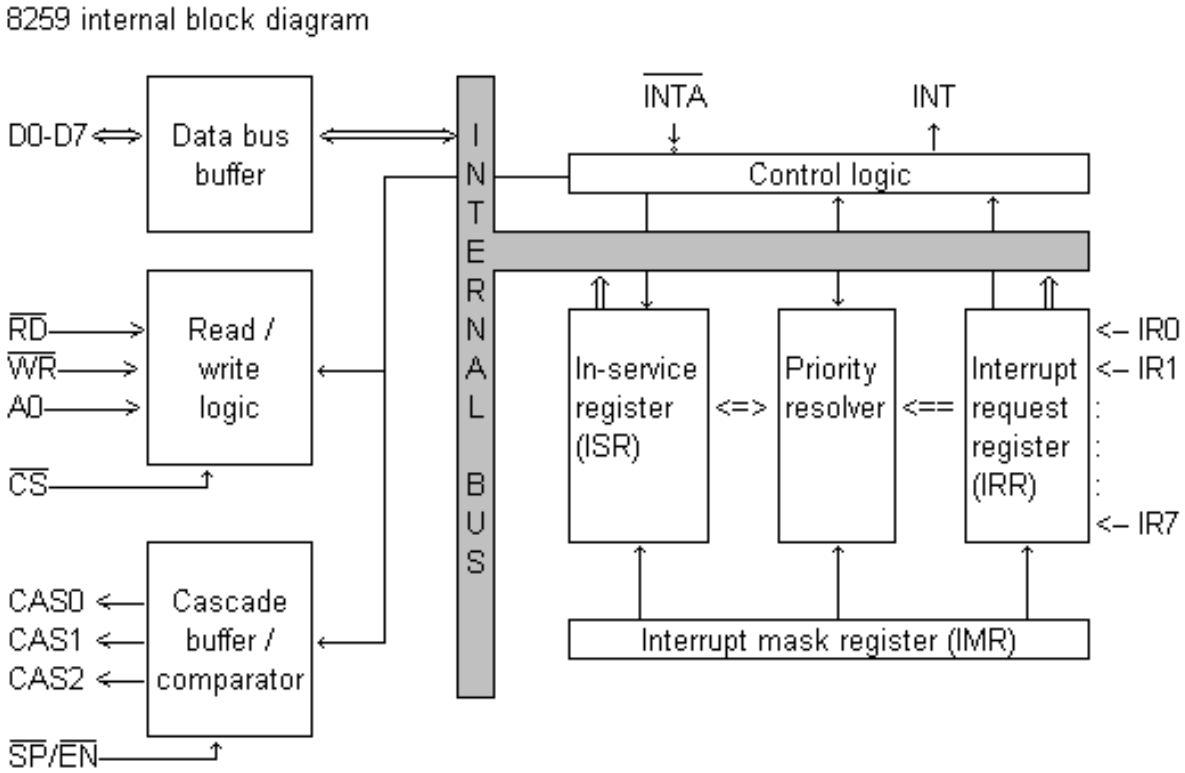
Polling

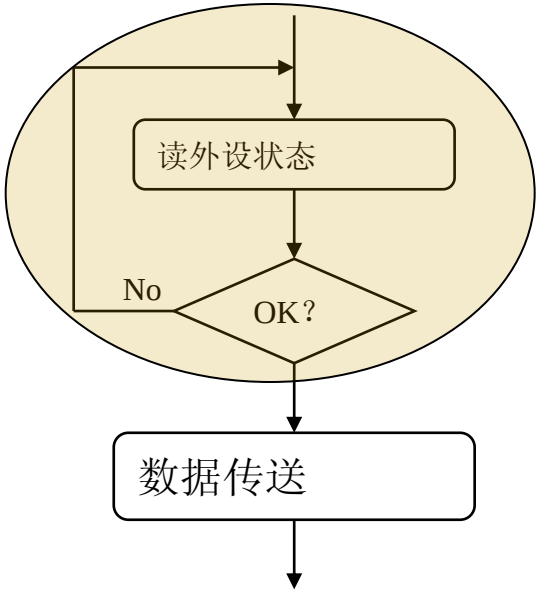

读外设状态
No
OK?
数据传送

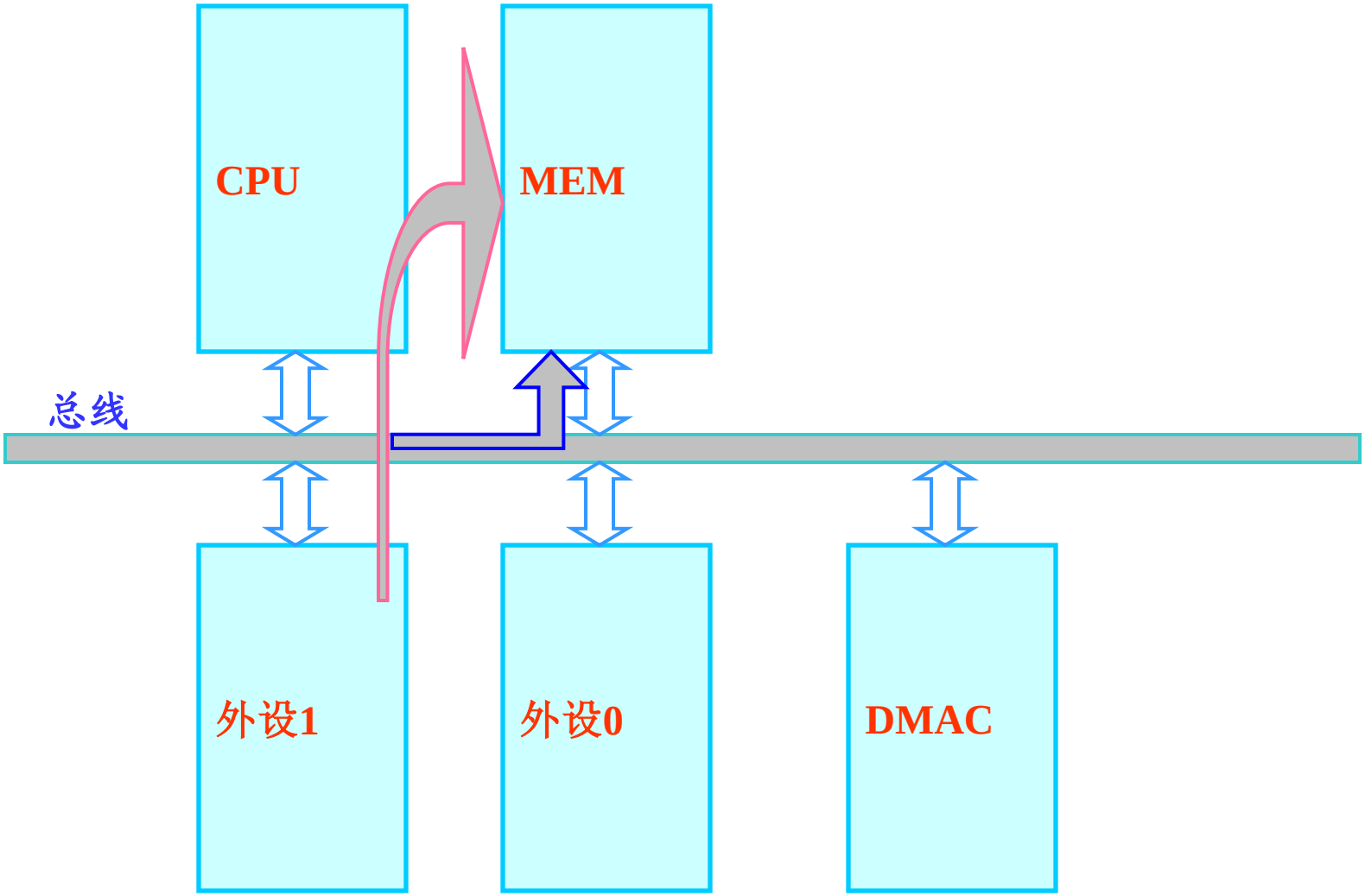

Status
OK?
Data Transfer
Status
OK?
Data Transfer

Interrupt

- 8259 internal block diagram






CPU
MEM
总线
外设1
外设0
DMAC

Compare polling, interrupts, DMA

- The disadvantage of polling is that it wastes a lot of processor time because processors are so much faster than I/O devices.
- 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.

- 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.

Overhead of Polling in an I/O System

Examples: **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 following three cases, assuming that you poll often enough so that no data is ever lost and assuming that devices are potentially always busy:

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 × 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 is 50K/2 = 25K

clock cycles per second for polling = 25K × 400

Fraction of the processor clock cycles consumed is

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

- *the hard disk:*

the number of polling access per second is 4M/16 = 250K

clock cycles per second for polling = 250K × 400

Fraction of the processor clock cycles consumed is

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

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

Examples: **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: **The interrupt rate when the disk is busy is the same as the polling rate. Hence,**

cycles per second for disk = 250K $\times$ ***500=125*** $\times$ ***10^6 cycles per second***

Fraction of the processor consumed during a transfer is

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

Assuming that the disk is only transferring data 5% of the time.

Fraction of the processor consumed on average is

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

As we can see, the absence of overhead when an I/O device is not actually transferring is the major advantage of an interrupt-driven interface versus polling.

Overhead of I/O Using DMA

Examples: **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. If a average transfer from disk is 8 KB, what fraction of the 500-Mhz processor is consumed if the disk is actively transferring 100% of the time? Ignore any impact from bus contention between the processor and DMA controller.**

Answer: **Each DMA transfer take** $8KB/(4MB/second)=2\times10^{-3}seconds$

so if the disk is constantly transferring, it requires

$$\frac{1000+500\ \frac{cycles}{transfer}}{2\times10^{-3}\ \frac{seconds}{transfer}}=750\times10^{3}\ \frac{clock\ cycles}{second}$$

Fraction of processor consumed= $\frac{750\times10^{3}}{500\times10^{6}}=0.2\%$

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.

8.6 Designing an I/O system

- Performance Analysis techniques:
  - *queuing theory*
  - *simulation*
  - *analysis, i.e., find the weakest link*
- The general approach 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 that sustains 300 million instructions per second and averages 50,000 instructions in the operating system per I/O operation***
2. ***A memory backplane bus capable of sustaining a transfer rate of 100 MB/sec***
3. ***SCSI-2 controllers with a transfer rate of 20 MB/sec and accommodating up to seven disks***
4. ***Disk drives with a read/write bandwidth of 5 MB/sec and an average seek plus rotational latency of 10 ms***

If the workload consists of 64-KB reads (where the block is sequential on a track) and the user program need 100,000 instructions per I/O operation, find the maximum sustainable I/O rate and the number of disks and SCSI controllers required. Assume that the reads can always be done on an idle disk if one exists (i.e., ignore disk conflicts).

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**.

$$\text{Maximum I/O rate of CPU} = \frac{\text{Instruction execution rate}}{\text{Instruction per I/O}} = \frac{300 \times 10^6}{(50+100) \times 10^3} = 2000 \frac{\text{I/Os}}{\text{seconds}}$$

$$\text{Maximum I/O rate of bus} = \frac{\text{Bus bandwidth}}{\text{Bytes per I/O}} = \frac{100 \times 10^6}{64 \times 10^3} = 1562 \frac{\text{I/Os}}{\text{seconds}}$$

The bus is the bottleneck, so we can now configure the rest of the system to perform at the level dictated by the bus, 1562 I/Os per second.

Now, let's determine how many disks we need to be able to Accommodate 1562 I/Os per second. To find the number of disks, we first find the time per I/O operation at the disk:

$$\textit{Time per I/O at disk} = \text{Seek/rotational time} + \text{Transfer time}$$

$$= 10\ \text{ms} + \frac{64\text{KB}}{5\text{MB/sec}} = 22.8\ \text{ms}$$

This means each disk can complete 43.9 I/Os per second. To saturate the bus requires 1562 I/Os per second, or 1562/43.9≈36 disks.

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

$$\text{Transfer rate} = \frac{\text{Transfer size}}{\text{Transfer time}} = \frac{64\text{KB}}{22.8\text{ms}} \approx 2.74\text{MB/sec}$$

Assuming the disk accesses are not clustered so that we can use all the bus bandwidth, we can place seven disks per SCSI bus and controller. This means we will need 36/7, or six buses and controllers.

8.7 Real Stuff: A Typical Desktop I/O System