



Input Output

Raul Queiroz Feitosa

Parts of these slides are from the support material provided by W. Stallings



Objective

This chapter presents concepts, structures and functions involved in I/O operation.

Outline

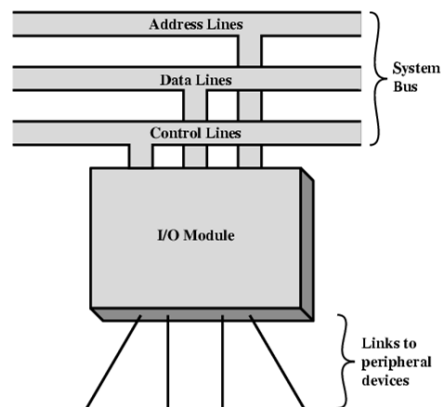
- I/O Modules
- I/O Techniques
 - Programmed I/O
 - Interrupt driven I/O
 - Direct Memory Access (DMA)
- I/O Channels and I/O Processors

Input Output

3

Generic Model of I/O Module

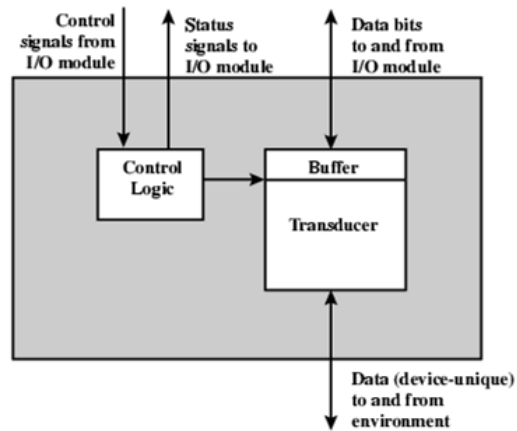
- Wide variety of peripherals
 - Delivering different amounts of data
 - At different speeds
 - In different formats
- Most of them slower than CPU and RAM
- Need I/O modules as **interface**
 - to CPU and Memory via bus or central switch
 - to one or more peripherals



Input Output

4

External Device Block Diagram



Input Output

5

I/O Module Function

1. Control & Timing
2. CPU Communication
3. Device Communication
4. Data Buffering
5. Error Detection

Input Output

6

I/O Module Function

Control & Timing : The data transfer will typically involve the following steps:

- a) CPU **checks** I/O module device **status**
- b) I/O module **returns status**
- c) If ready, CPU **requests** data **transfer**
- d) I/O module **gets data** from device
- e) I/O module **transfers data** to CPU
- d) I/O module **transfers data** from CPU
- e) I/O module **sends data** to device

Input Output

7

I/O Module Function

Processor Communication : involves the following steps:

- a) **Command decoding:** typically word commands are sent through the bus to the I/O module.
- b) **Data:** exchanged between cpu and I/O module through the bus.
- c) **Status reporting:** because peripherals are slow the cpu must know the status of the I/O module – *is it ready to receive one more data unit?*
- d) **Address recognition:** I/O modules are accessed quite in the same way as memory locations

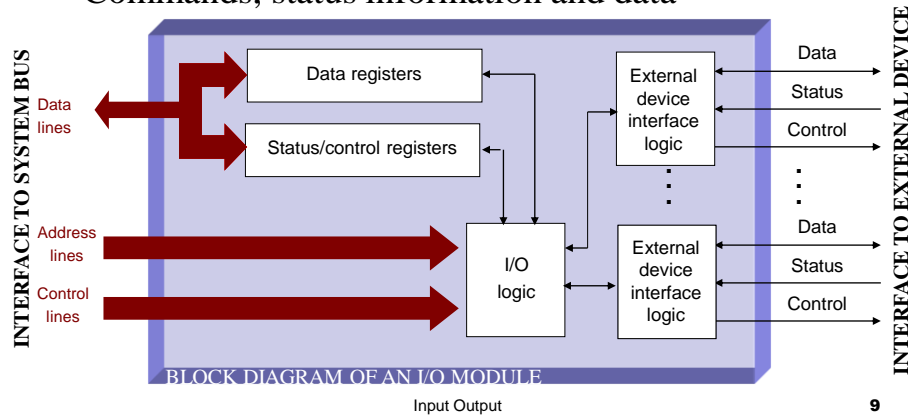
Input Output

8

I/O Module Function

Device Communication

Commands, status information and data



I/O Module Function

Data Buffering

to cope with the different transfer rates between memory and peripheral devices.

Error Detection

and reporting to the processor

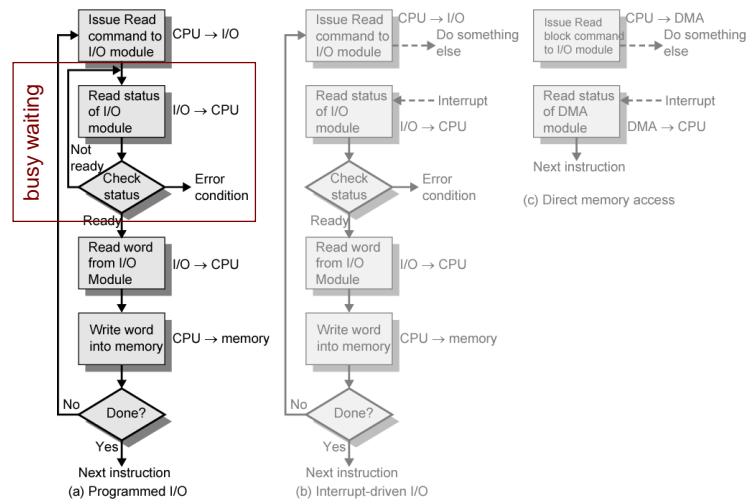
Outline

- I/O Modules
- I/O Techniques
 - Programmed I/O
 - Interrupt driven I/O
 - Direct Memory Access (DMA)
- I/O Channels and I/O Processors

Input Output Techniques

I/O Technique	Task	
	Read Status	Data Transfer
1. Programmed I/O	CPU	CPU
2. Interrupt driven I/O	I/O module	CPU
3. Direct Memory Access (DMA)	DMA module	DMA module

Programmed I/O



13

Programmed I/O

- **CPU has direct control over I/O**
 - Sensing status
 - Read/write commands
 - Transferring data
- **CPU waits for I/O module to complete operation**
- **Wastes CPU time**
- **Simplest implementation**

Input Output

14

I/O Commands

CPU issues address

- Identifies module (& device if >1 per module)
- Very like memory accesses with separated or common address spaces

CPU issues command

- Control - telling module what to do
 - e.g. spin up disk
- Test - check status
 - e.g. power? Error?

CPU reads/writes information

- Module transfers data via buffer from/to device

Input Output

15

I/O Mapping

Memory mapped I/O

- Devices and memory share a common address space
- I/O looks just like memory read/write
- No special commands for I/O
 - Large selection of memory access commands available

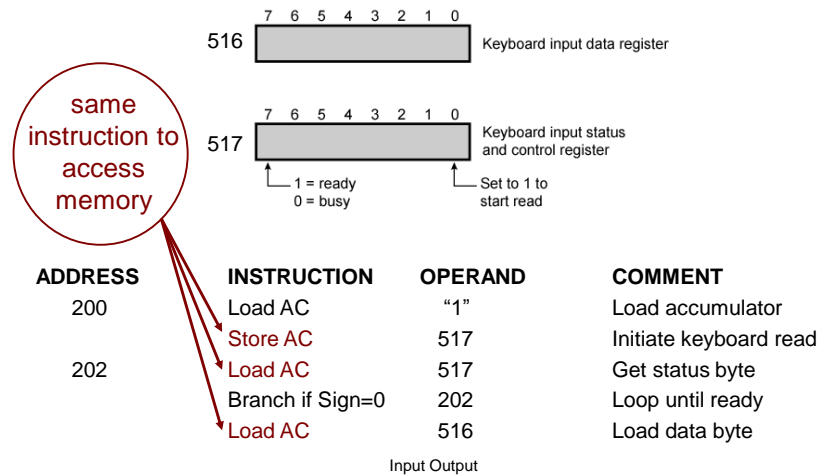
Isolated I/O

- Separate address spaces
- Need I/O or memory select lines
- Special commands for I/O
 - Limited set

Input Output

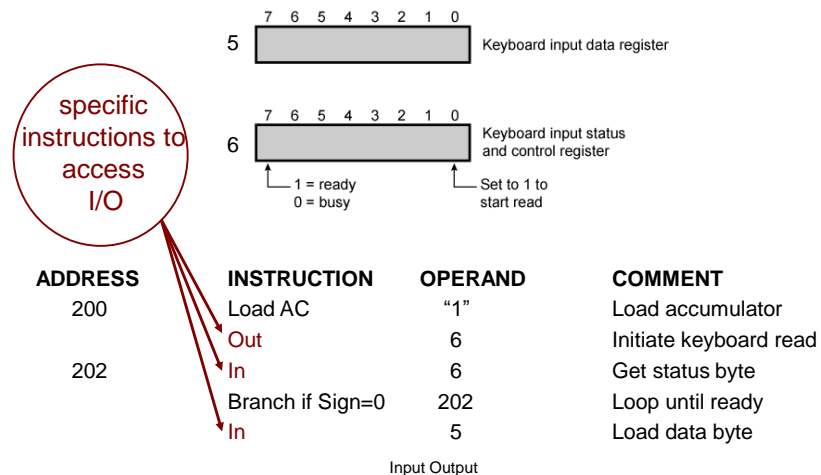
16

Memory Mapped I/O



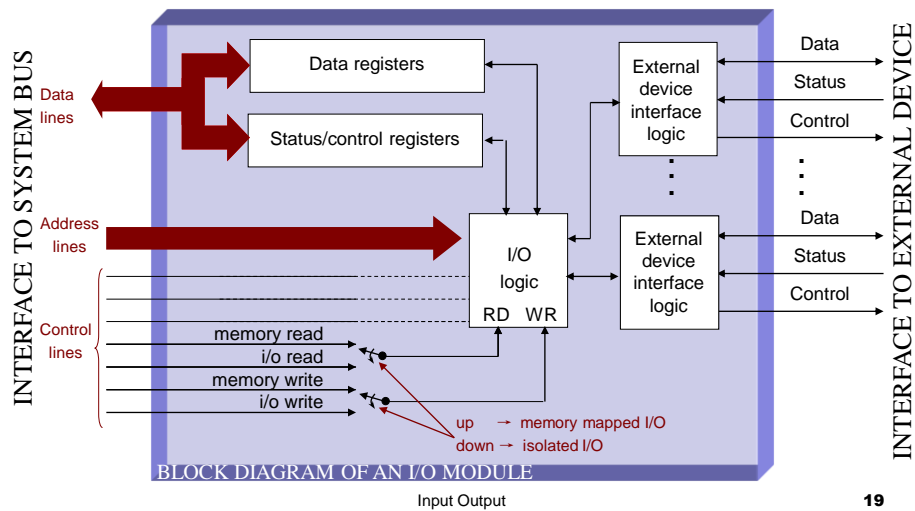
17

Isolated I/O



18

I/O Mapping



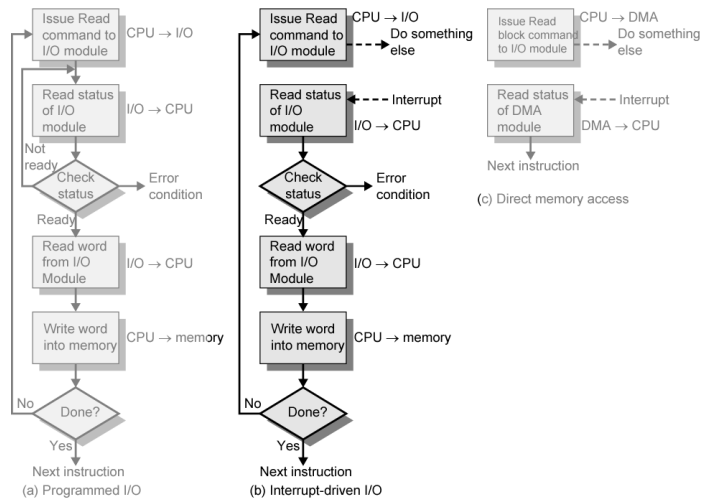
Interrupt Driven I/O

Overcomes CPU waiting

No repeated CPU checking of device

I/O module interrupts when ready

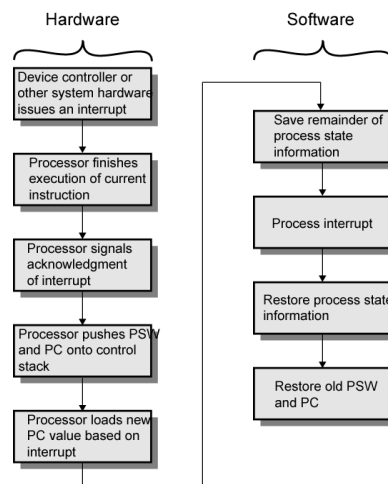
Interrupt Driven I/O



Input Output

21

Simple Interrupt Processing

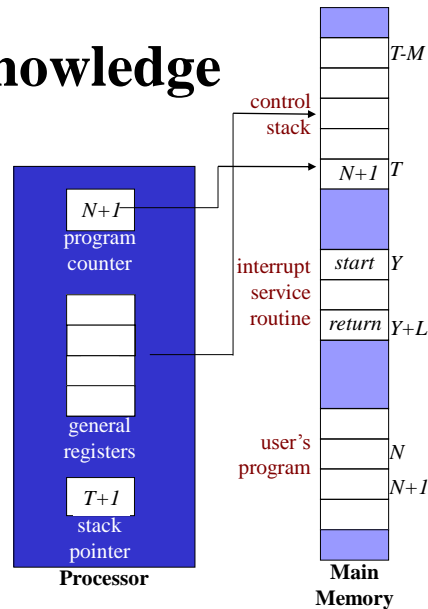


Input Output

22

On Interrupt Acknowledge

1. Processor acknowledges interrupt request, while executing instruction at location N .
2. Processor saves current PC and register contents program.
3. Processor loads the entry location of the interrupt-handling program.

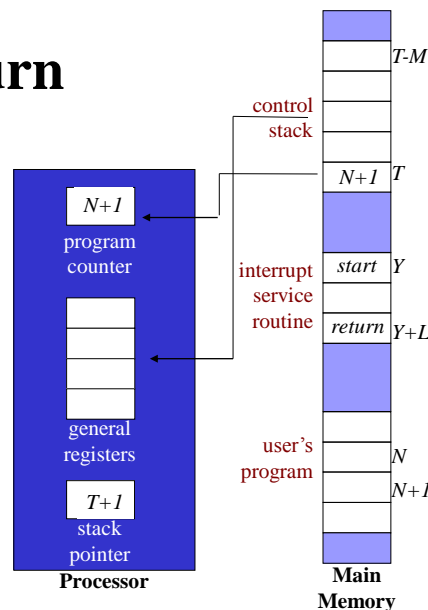


Input Output

23

On Interrupt Return

1. Processor retrieves saved register contents from the stack.
2. Processor retrieves saved PC and PSW from the stack.



Input Output

24

Design Issues

1. **How do you identify the module issuing the interrupt?**
2. **How do you deal with multiple interrupts?**
 - ☐ i.e. an interrupt handler being interrupted

Input Output

25

Identifying Interrupting Module (1)

Different line for each module

- ☐ PC
- ☐ Limits number of devices

Software poll

- ☐ CPU asks each module in turn → Slow

Input Output

26

Identifying Interrupting Module (2)

Daisy Chain or Hardware poll

- ☐ Interrupt Acknowledge (\approx read access)
 - sent down a chain, or
 - parallel to all devices.
- ☐ Module responsible places vector on bus
- ☐ CPU uses vector to identify handler routine

Bus Master

- ☐ Module must claim the bus before it can raise interrupt e.g. PCI & SCSI

Input Output

27

Direct Memory Access

Drawbacks with previous I/O strategies :

- ☐ both require active CPU intervention
 - Transfer rate is limited
 - CPU is tied up

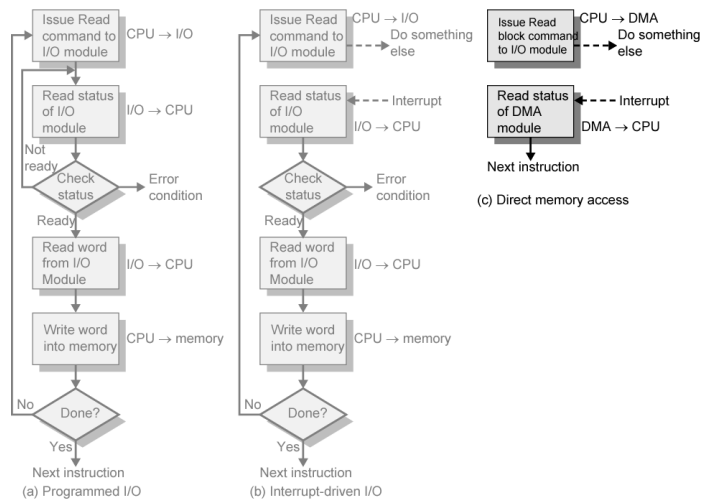
DMA is the answer!

- ☐ Additional Module (hardware) on bus
- ☐ DMA controller takes over from CPU for I/O

Input Output

28

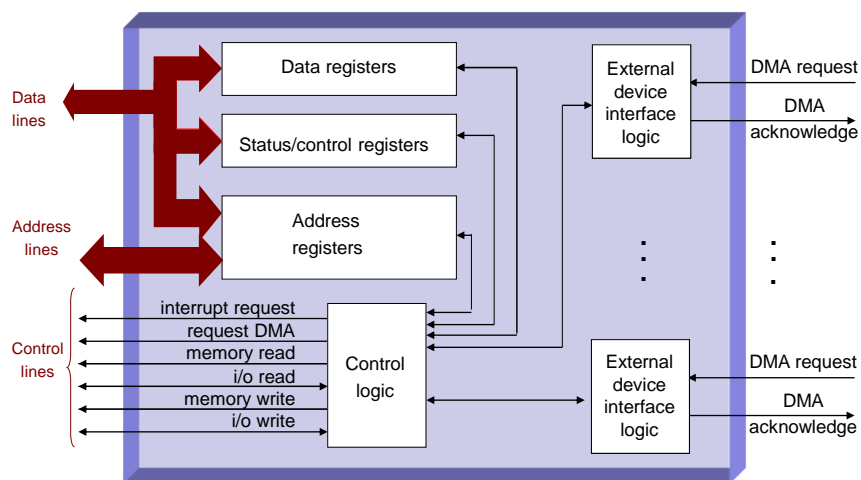
Direct Memory Access



Input Output

29

DMA controller diagram

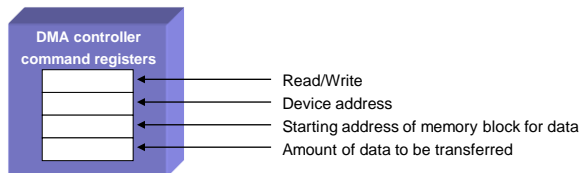


Input Output

30

DMA Operation

- CPU tells DMA controller:

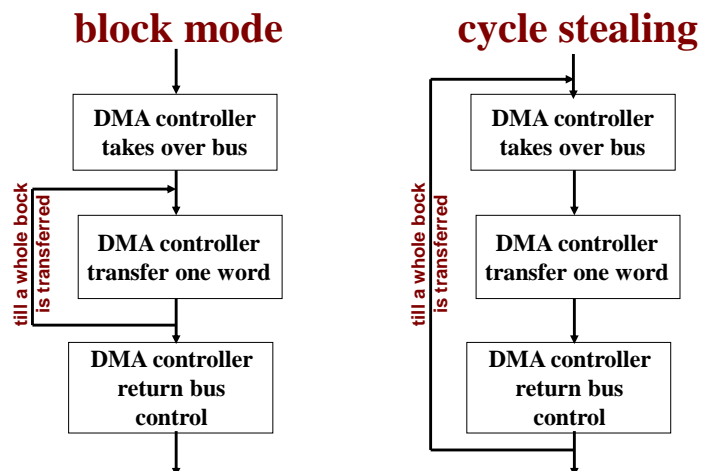


- CPU carries on with **other work**
- DMA controller deals with **transfer**
- DMA controller **sends interrupt** when finished

Input Output

31

DMA transfer modes



Input Output

32

DMA transfer modes

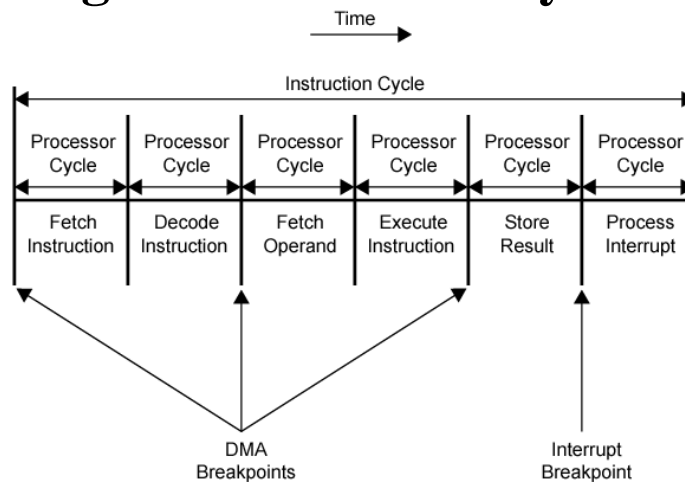
In *block mode* as well as in *cycle stealing*

- CPU is **suspended, not interrupted**
 - CPU does not switch context
- CPU **suspended just before it accesses bus**
 - i.e. before an operand or data fetch or a data write
- Slows down CPU but not as much as CPU doing transfer.

Input Output

33

DMA and Interrupt Breakpoints During an Instruction Cycle



Input Output

34

DMA Configurations (1)

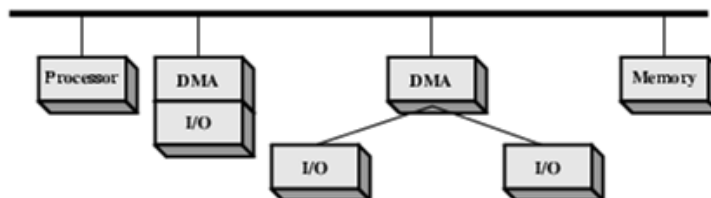


- Single Bus, Detached DMA controller

Input Output

35

DMA Configurations (2)

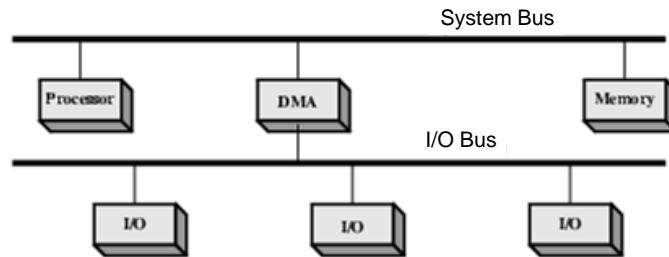


- Single Bus, Integrated DMA –I/Ocontroller
- Controller may support >1 device

Input Output

36

DMA Configurations (3)



- ☐ Separate I/O Bus
- ☐ Bus supports all DMA enabled devices

Input Output

37

Outline

- I/O Modules
- I/O Techniques
 - ☐ Programmed I/O
 - ☐ Interrupt driven I/O
 - ☐ Direct Memory Access (DMA)
- I/O Channels and I/O Processors

Input Output

38

I/O Channels and I/O Processors

- Enhanced I/O modules with a **built-in programmable processor**
 - if it shares the system memory → **I/O channel**
 - if it has its own memory → **I/O processor**
- CPU instructs I/O channel/processor to do transfer
- I/O channel/processor does entire transfer and interrupts CPU at the end.
- Improves overall speed.
- e.g. 3D graphics cards.

Input Output

39

I/O Channels and I/O Processors

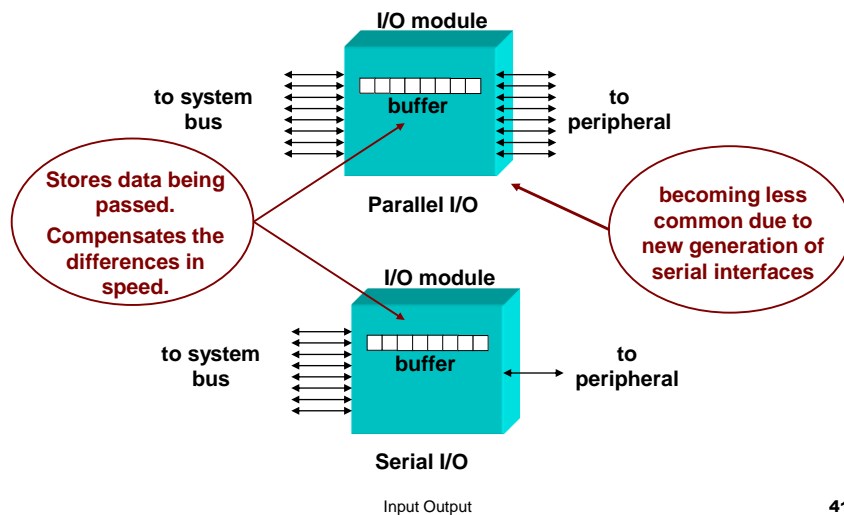
Types

- **Selector channel**
 - Controls multiple high-speed devices
 - And is dedicated to **one of them at a time**
 - **Each** (or a small set of) **device is handled by a controller** or I/O module
 - The **IO/Channel** serves **in place of** the **CPU** to control these I/O controllers
- **Multiplexor channel**
 - Handle multiple devices **at the same time**.

Input Output

40

Interfacing



Firewire versus USB

Firewire:

- Firewire 400: 400 Mbps = 50 MB/s
- Firewire 800: 800 Mbps = 100 MB/s

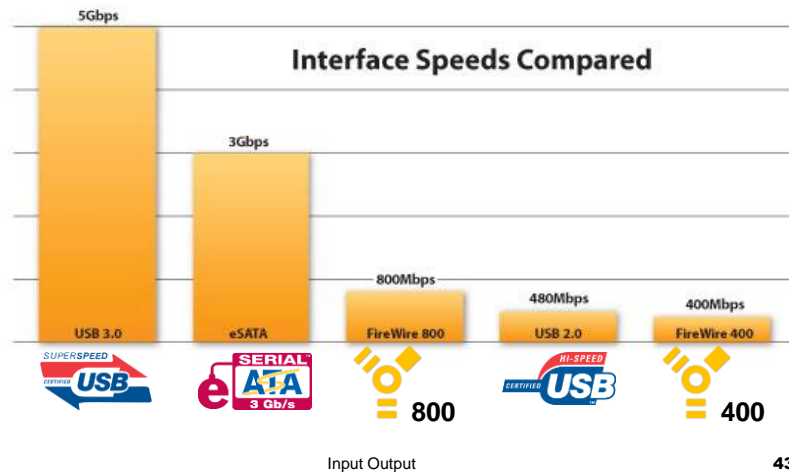


USB:

- USB 1.0: 1.5 Mbps = 0.188 MB/s
- USB 1.1: 12 Mbps = 1.5 MB/s
- USB 2.0: 480 Mbps = 60 MB/s
- USB 3.0: 5000 Gbps = 640 MB/s (rarely in practice)



Interface Speeds



Exercise 7.6

For a programmed I/O the processor is stuck in a wait loop doing status checking of an I/O device. To increase efficiency, the I/O software could be written so that the processor periodically checks the stats of the device. If the device is not ready, the process can jump to other tasks. After some timed interval, the processor comes back to check status again

- Consider the above scheme for outputting data one character at a time to a printer that operates at 10 characters per second (cps). What will happen if the status is scanned every 200ms?
- Next consider a keyboard with a single character buffer. On average, characters are entered at a rate of 10 cps. However, the time interval between two consecutive key depressions can be as short as 80 ms. At what frequency should the keyboard be scanned by the I/O programm?

Exercise 7.8

In this presentation one advantage and one disadvantage of memory-mapped I/O compared with isolated I/O were listed. List two more advantages and two more disadvantages.

Exercise 7.10

Consider a system employing interrupt-driven I/O for a particular device that transfers data at an average of 8 KB/s on a continuous basis.

- a) Assume that interrupt processing takes about $100\ \mu\text{s}$ (i.e., the time to jump to the interrupt service routine (ISR), execute it, and return to the main program). Determine what fraction of processor time is consumed by this I/O device if it interrupts for every byte.
- b) Now assume that the device has two 16-byte buffers and interrupts the processor when one of the buffers is full. Naturally, interrupt processing takes longer, because the ISR must transfer 16 bytes. While executing the ISR, the processor takes about $8\ \mu\text{s}$ for the transfer of each byte. Determine what fraction of processor time is consumed by this I/O device in this case.

Exercise 7.12

A DMA module is transferring characters to memory using cycle stealing, from a device transmitting at 9600 bps. The processor is fetching instructions at the rate of 1 million instruction per second (1 MIPS). By how much will the processor be slowed down due to the DMA activity?

Exercise 7.13

Consider a system in which bus cycles take 500 ns. Transfer of bus control in either direction, from processor to I/O device or vice-versa, takes 250 ns. One of the I/O devices has a data transfer rate of 50 KB/s and employs DMA. Data are transferred one byte at a time.

- a) Suppose we employ DMA at a burst mode. That is, the DMA interface gains bus mastership prior to the start of a block transfer and maintains control of the bus until the whole block is transferred. For how long would the device tie up the bus when transferring a block of 128 bytes?
- b) Repeat the calculation for cycle-stealing mode.

Question of a prior G2

At t_0 a disk controller receives from processor a command to transfer a 512 byte long sector to the main memory. These bytes become available in the controller's buffer 5,000 μ s after t_0 . The byte transfer from buffer to main memory may be executed at any time from then on. The controller may adopt three management strategies:

- a) **Programmed I/O:** Starting at $t_0 + 500 \mu$ s a program is executed by the processor every 500 μ s to check if the buffer is full. If the buffer is still not full, the execution time of this program is equal to 50 μ s. If the buffer is full, 5 μ s is added to this execution time for each transferred byte. The symbol t_f denotes the time when the transfer of all bytes is completed. Compute the time between t_0 and t_f , available for the processor to execute tasks other than I/O management.
- b) **Interrupt:** As soon as the buffer is full, an interrupt service routine is executed. Its execution time is the same as the program of question a) when the buffer is full. In this case the transfer is completed at t_2 . Compute the time between t_0 and t_2 , available for the processor to execute tasks other than I/O management.
- c) **DMA:** When the buffer is full the DMA controller transfers the data in *block mode*, one byte per μ s. Assume that the time to switch bus ownership between processor and DMA controller and vice-versa are negligible. In this case the transfer is finished at t_3 . Compute the time between t_0 and t_3 , available for the processor to execute tasks other than I/O management.

Text Book References

These topics are covered in

- ☐ Stallings - chapter 7
- ☐ Tanenbaum -
- ☐ Parhami - chapter 22



Input Output

END

Input Output

51