I/O Systems

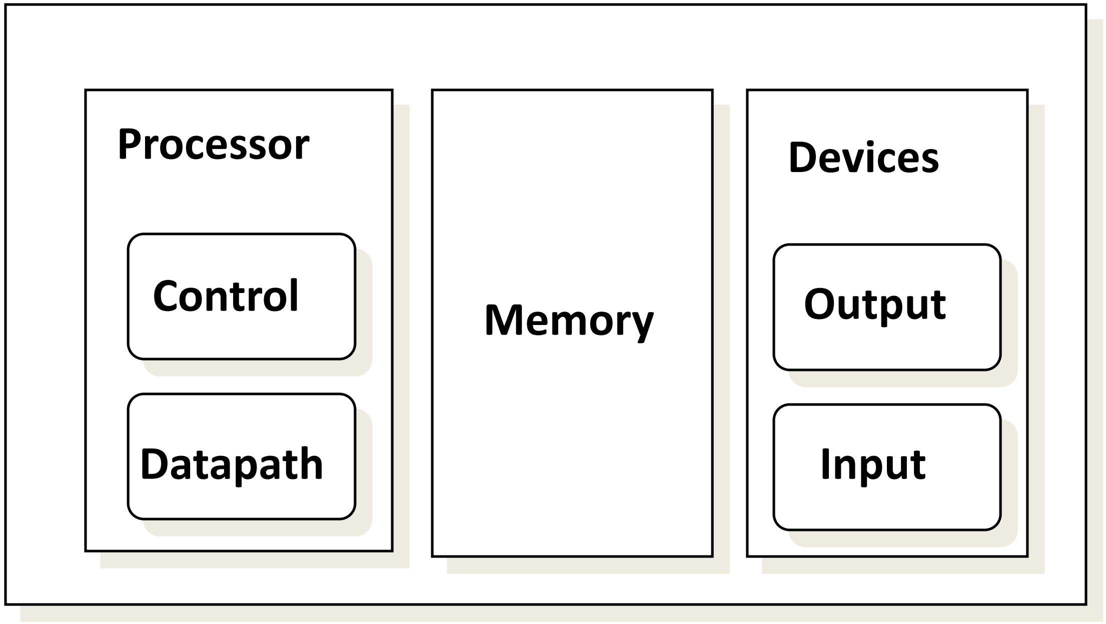
Computer Systems Organization, Spring 2018

Lavanya Ramapantulu
Email: lavanya.r@iiit.ac.in
Office: Vindhya A5-304b

Acknowledgements

- These slides have been adapted from Prof. Mary Jane Irwin’s slides at PSU.

Review: Major Components of a Computer



- Important metrics for an I/O system
  - Performance
  - Expandability
  - Dependability
  - Cost, size, weight

Input and Output Devices

Device	Behavior	Partner	Data rate (Mb/s)
Keyboard	input	human	0.0001
Mouse	input	human	0.0038
Laser printer	output	human	3.2000
Graphics display	output	human	800.0000-8000.0000
Network/LAN	input or output	machine	100.0000-1000.0000
Magnetic disk	storage	machine	240.0000-2560.0000

8 orders of magnitude range

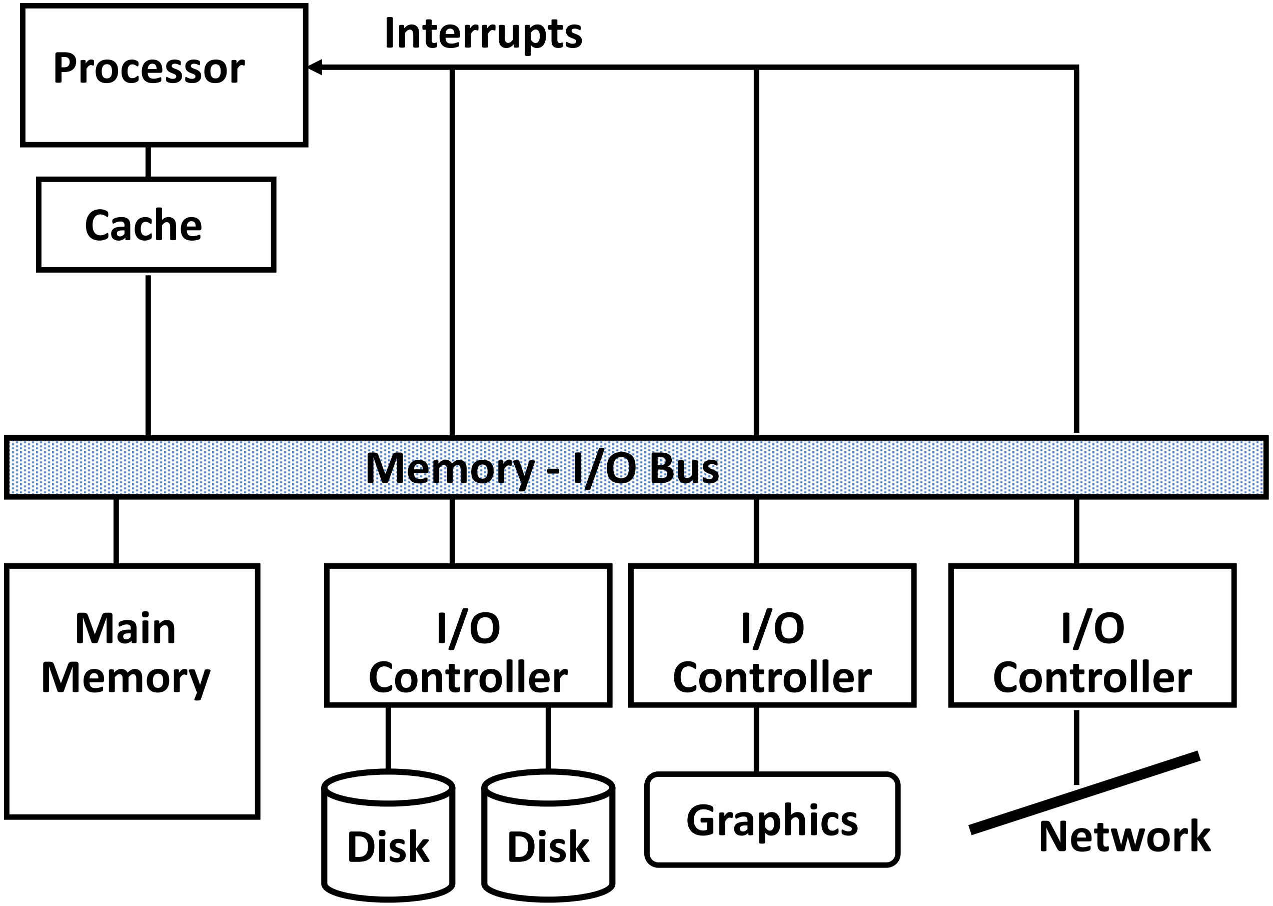
I/O Performance Measures

- I/O bandwidth (throughput) – amount of information that can be input (output) and communicated across an interconnect (e.g., a bus) to the processor/memory (I/O device) per unit time
  1. How much data can we move through the system in a certain time?
  2. How many I/O operations can we do per unit time?

- I/O response time (latency) – the total elapsed time to accomplish an input or output operation
  – An especially important performance metric in real-time systems

- Many applications require both high throughput and short response times

A Typical I/O System



I/O System Performance

- Designing an I/O system to meet a set of bandwidth and/or latency constraints means

1. Finding the weakest link in the I/O system – the component that constrains the design
   - The processor and memory system ?
   - The underlying interconnection (e.g., bus) ?
   - The I/O controllers ?
   - The I/O devices themselves ?
2. (Re)configuring the weakest link to meet the bandwidth and/or latency requirements
3. Determining requirements for the rest of the components and (re)configuring them to support this latency and/or bandwidth

I/O System Performance Example

- A disk workload consisting of 64KB reads and writes where the user program executes 200,000 instructions per disk I/O operation and
  - a processor that sustains 3 billion instr/s and averages 100,000 OS instructions to handle a disk I/O operation

The maximum disk I/O rate (# I/O's/s) of the processor is

$$\frac{\text{Instr execution rate}}{\text{Instr per I/O}} = \frac{3 \times 10^{9}}{(200 + 100) \times 10^{3}} = \boxed{10{,}000 \text{ I/O's/s}}$$

- a memory-I/O bus that sustains a transfer rate of 1000 MB/s

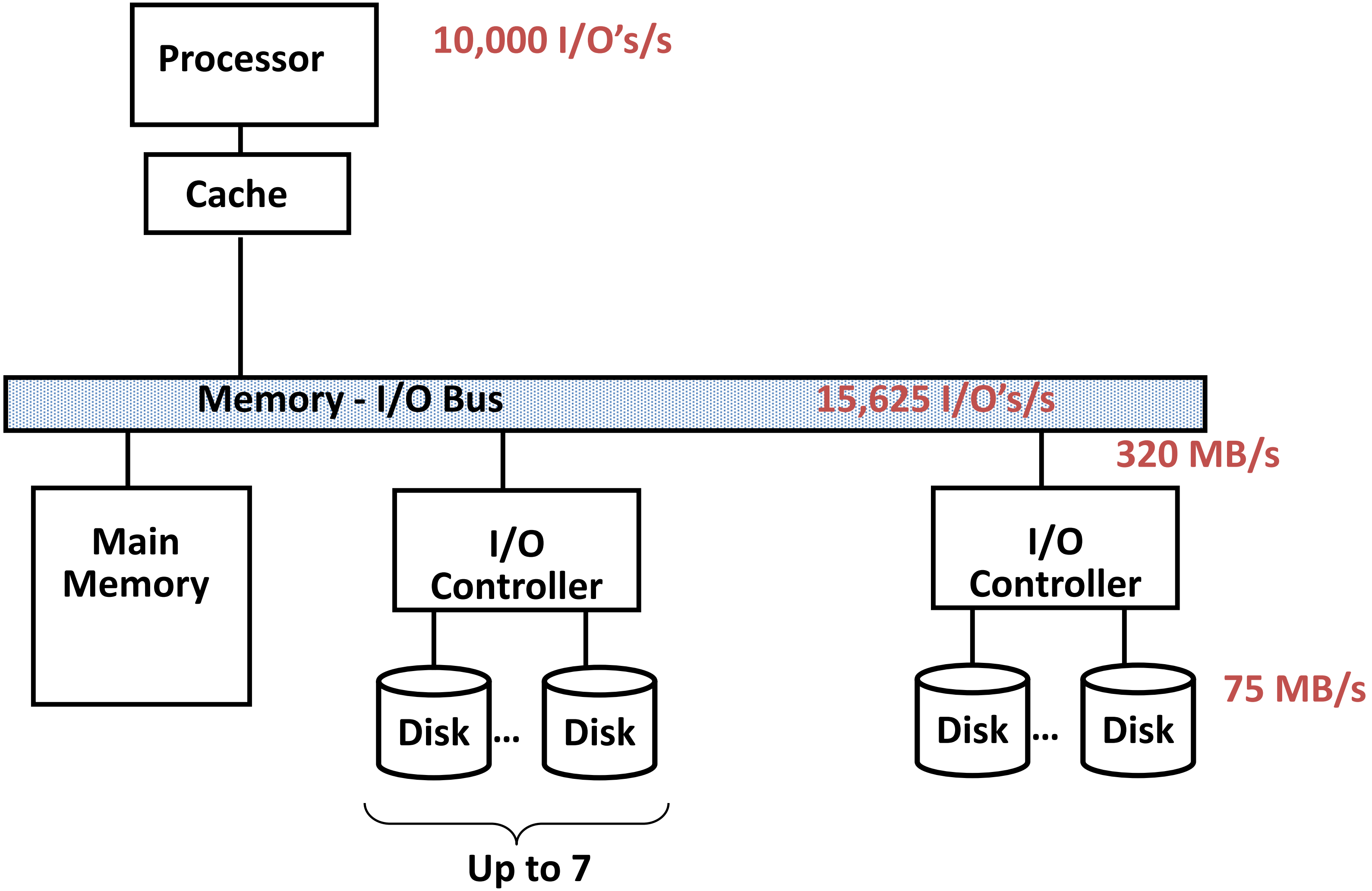
Each disk I/O reads/writes 64 KB so the maximum I/O rate of the bus is

$$\frac{\text{Bus bandwidth}}{\text{Bytes per I/O}} = \frac{1000 \times 10^{6}}{64 \times 10^{3}} = \boxed{15{,}625 \text{ I/O's/s}}$$

- SCSI disk I/O controllers with a DMA transfer rate of 320 MB/s that can accommodate up to 7 disks per controller
- disk drives with a read/write bandwidth of 75 MB/s and an average seek plus rotational latency of 6 ms

what is the maximum sustainable I/O rate and what is the number of disks and SCSI controllers required to achieve that rate?

Disk I/O System Example



I/O System Performance Example, Con't

So the processor is the bottleneck, not the bus

- disk drives with a read/write bandwidth of 75 MB/s and an average seek plus rotational latency of 6 ms

Disk I/O read/write time = seek + rotational time + transfer time =
6ms + 64KB/(75MB/s) = 6.9ms

Thus each disk can complete 1000ms/6.9ms or 146 I/O's per second. To saturate the processor requires 10,000 I/O's per second or
10,000/146 = 69 disks

To calculate the number of SCSI disk controllers, we need to know the average transfer rate per disk to ensure we can put the maximum of 7 disks per SCSI controller and that a disk controller won't saturate the memory-I/O bus during a DMA transfer

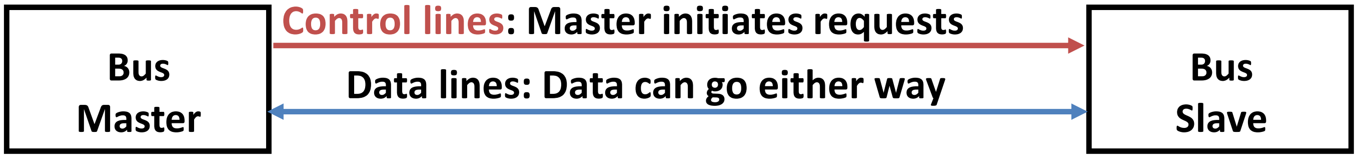
Disk transfer rate = (transfer size)/(transfer time) = 64KB/6.9ms = 9.56 MB/s

Thus 7 disks won't saturate either the SCSI controller (with a maximum transfer rate of 320 MB/s) or the memory-I/O bus (1000 MB/s). This means we will need 69/7 or 10 SCSI controllers.

I/O System Interconnect Issues

- A bus is a shared communication link (a single set of wires used to connect multiple subsystems) that needs to support a range of devices with widely varying latencies and data transfer rates
  - Advantages
    - Versatile – new devices can be added easily and can be moved between computer systems that use the same bus standard
    - Low cost – a single set of wires is shared in multiple ways
  - Disadvantages
    - Creates a communication bottleneck – bus bandwidth limits the maximum I/O throughput
- The maximum bus speed is largely limited by
  - The length of the bus
  - The number of devices on the bus

Bus Characteristics


Bus
Master
Control lines: Master initiates requests
Data lines: Data can go either way
Bus
Slave

- Control lines
  - Signal requests and acknowledgments
  - Indicate what type of information is on the data lines
- Data lines
  - Data, addresses, and complex commands
- Bus transaction consists of
  - Master issuing the command (and address) – request
  - Slave receiving (or sending) the data – action
  - Defined by what the transaction does to memory
    - Input – inputs data from the I/O device to the memory
    - Output – outputs data from the memory to the I/O device

Types of Buses

- Processor-memory bus (proprietary)
  - Short and high speed
  - Matched to the memory system to maximize the memory-processor bandwidth
  - Optimized for cache block transfers

- I/O bus (industry standard, e.g., SCSI, USB, Firewire)
  - Usually is lengthy and slower
  - Needs to accommodate a wide range of I/O devices
  - Connects to the processor-memory bus or backplane bus

- Backplane bus (industry standard, e.g., ATA, PCIexpress)
  - The backplane is an interconnection structure within the chassis
  - Used as an intermediary bus connecting I/O busses to the processor-memory bus

Bus Bandwidth Determinates

- The bandwidth of a bus is determined by
  - Whether its is synchronous or asynchronous and the timing characteristics of the protocol used
  - The data bus width
  - Whether the bus supports block transfers or only word at a time transfers

	Firewire	USB 2.0
Type	I/O	I/O
Data lines	4	2
Clocking	Asynchronous	Synchronous
Max # devices	63	127
Max length	4.5 meters	5 meters
Peak bandwidth	50 MB/s (400 Mbps) 100 MB/s (800 Mbps)	0.2 MB/s (low) 1.5 MB/s (full) 60 MB/s (high)

I/O Devices and Processor Communication

- How the processor directs the I/O devices
  - Special I/O instructions
    - Must specify both the device and the command
  - Memory-mapped I/O
    - Portions of the high-order memory address space are assigned to each I/O device
    - Read and writes to those memory addresses are interpreted as commands to the I/O devices
    - Load/stores to the I/O address space can only be done by the OS
- How the I/O device communicates with the processor
  - Polling – the processor periodically checks the status of an I/O device to determine its need for service
    - Processor is totally in control – but does all the work
    - Can waste a lot of processor time due to speed differences
  - Interrupt-driven I/O – the I/O device issues an interrupts to the processor to indicate that it needs attention

Interrupt-Driven I/O

- An I/O interrupt is asynchronous wrt instruction execution
  - Is not associated with any instruction so doesn’t prevent any instruction from completing
    - You can pick your own convenient point to handle the interrupt
- With I/O interrupts
  - Need a way to identify the device generating the interrupt
  - Can have different urgencies (so may need to be prioritized)
- Advantages of using interrupts
  - Relieves the processor from having to continuously poll for an I/O event; user program progress is only suspended during the actual transfer of I/O data to/from user memory space
- Disadvantage – special hardware is needed to
  - Cause an interrupt (I/O device) and detect an interrupt and save the necessary information to resume normal processing after servicing the interrupt (processor)

Direct Memory Access (DMA)

- For high-bandwidth devices (like disks) interrupt-driven I/O would consume a *lot* of processor cycles
- DMA – the I/O controller has the ability to transfer data directly to/from the memory without involving the processor
    1. The processor initiates the DMA transfer by supplying the I/O device address, the operation to be performed, the memory address destination/source, the number of bytes to transfer
    2. The I/O DMA controller manages the entire transfer (possibly thousand of bytes in length), arbitrating for the bus
    3. When the DMA transfer is complete, the I/O controller interrupts the processor to let it know that the transfer is complete
- There may be multiple DMA devices in one system
    - Processor and I/O controllers contend for bus cycles and for memory

The DMA Stale Data Problem

- In systems with caches, there can be two copies of a data item, one in the cache and one in the main memory
  - For a DMA read (from disk to memory) – the processor will be using stale data if that location is also in the cache
  - For a DMA write (from memory to disk) and a write-back cache – the I/O device will receive stale data if the data is in the cache and has not yet been written back to the memory
- The coherency problem is solved by
  1. Routing all I/O activity through the cache – expensive and a large negative performance impact
  2. Having the OS selectively invalidate the cache for an I/O read or force write-backs for an I/O write (flushing)
  3. Providing hardware to selectively invalidate or flush the cache – need a hardware snooper

I/O and the Operating System

- The operating system acts as the interface between the I/O hardware and the program requesting I/O
  - To protect the shared I/O resources, the user program is not allowed to communicate directly with the I/O device

- Thus OS must be able to give commands to I/O devices, handle interrupts generated by I/O devices, provide equitable access to the shared I/O resources, and schedule I/O requests to enhance system throughput
  - I/O interrupts result in a transfer of processor control to the supervisor (OS) process

Reference

- Computer Organization and Design, The Hardware/Software Interface by Patterson and Hennessy