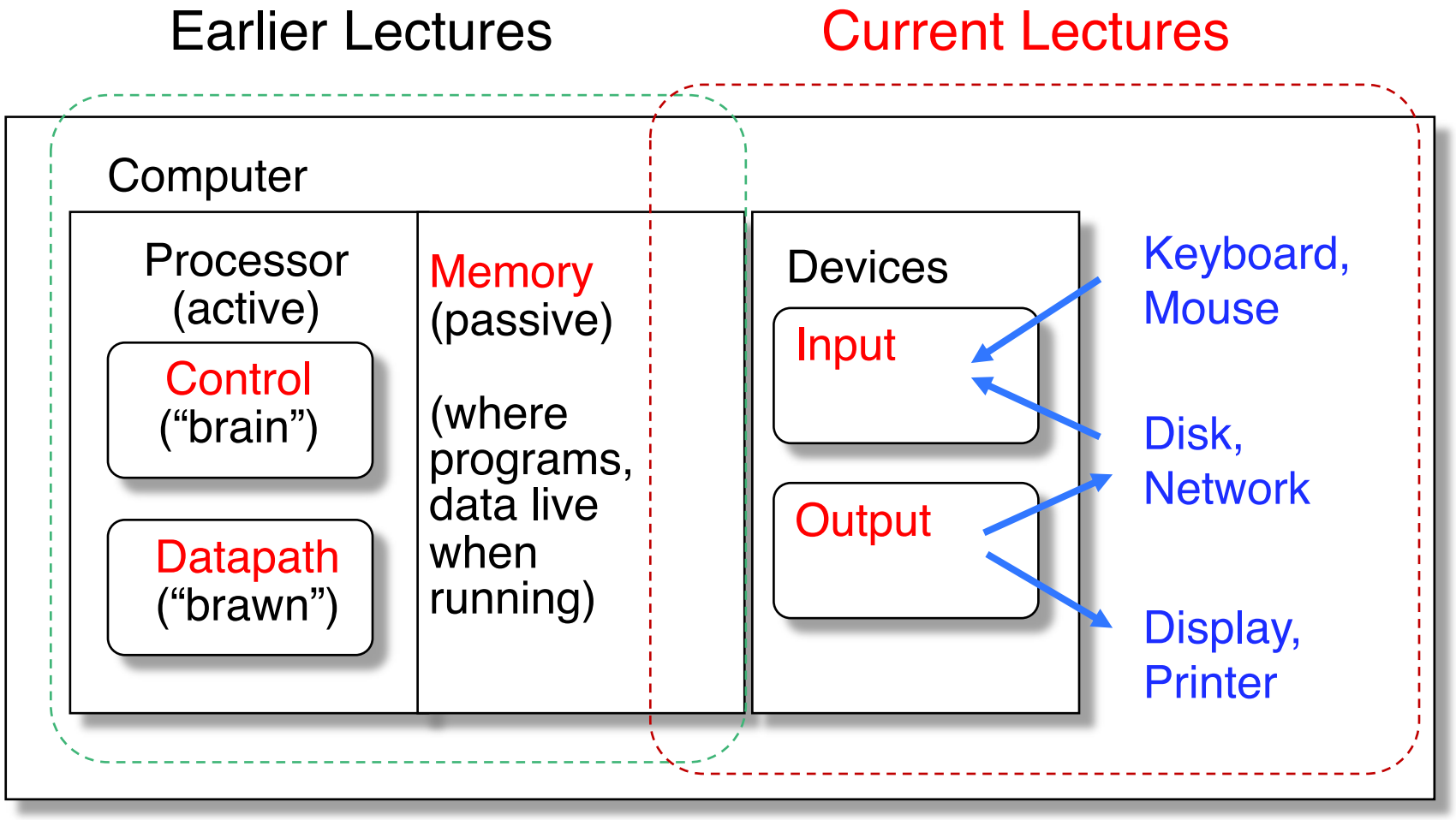
appendix

Input and Output

most slides are adapted from Prof. Yong's slides

School of CS and EE
Handong Global University

Recall : 5 Components of Computer



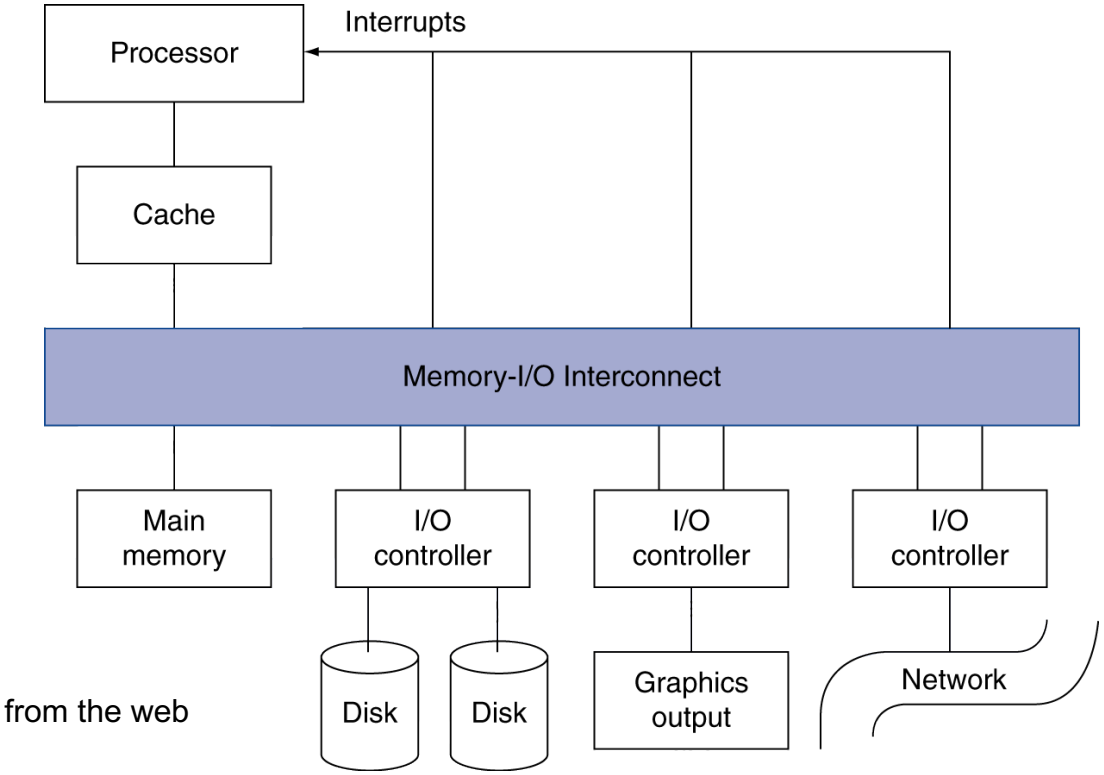
Motivation for I/O Systems

- I/O is how humans interact with computers
- I/O gives computers long-term memory
- I/O lets computers do amazing things:
  - Read pressure of synthetic hand and control synthetic arm (Robot)
  - Control propellers, fins (Drone)
  - Smart car (Unmanned Vehicle)

- Computer without I/O is like a car without wheels
  - great technology, but won't get you anywhere

Introduction

- I/O devices can be characterized by
  - Behaviour: input, output, storage
  - Partner: human or machine
  - Data rate: bytes/sec, transfers/sec
- I/O bus connections



from the web

IO devices

- I/O Speed:
  - bytes transferred per second
    (from mouse to Gigabit LAN: 7 orders of mag!)

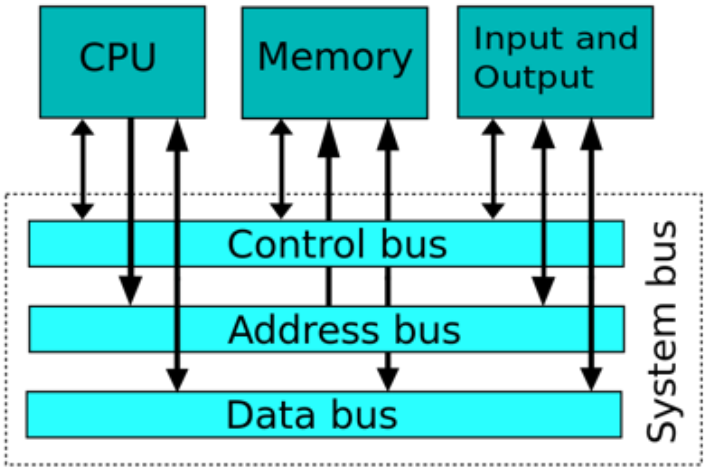
Device	Behavior	Partner	Data rate (KB/sec)
Keyboard	input	human	0.01
Mouse	input	human	0.02
Voice input	input	human	0.02
Scanner	input	human	400.00
Voice output	output	human	0.60
Line printer	output	human	1.00
Laser printer	output	human	200.00
Graphics display	output	human	60,000.00
Modem	input or output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

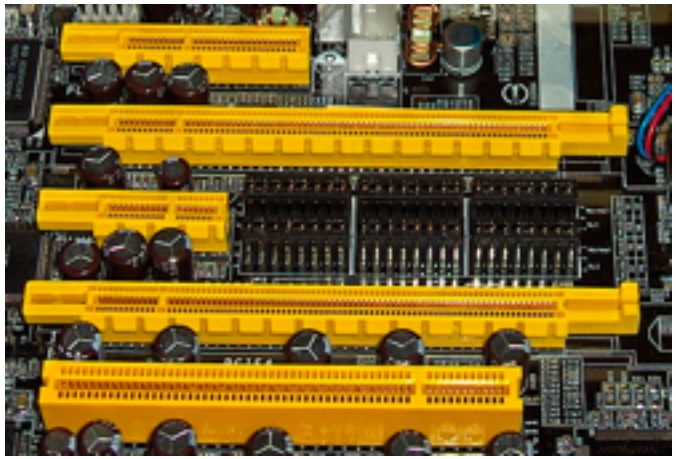
Interconnecting Components

- Need interconnections between
  - CPU, memory, I/O controllers
- Bus: shared communication channel
  - Parallel set of wires for data and synchronization of data transfer
  - Can become a bottleneck
  - Performance limited by physical factors
    - Wire length, number of connections
- More recent alternative
  - high-speed serial connections with switches, like networks

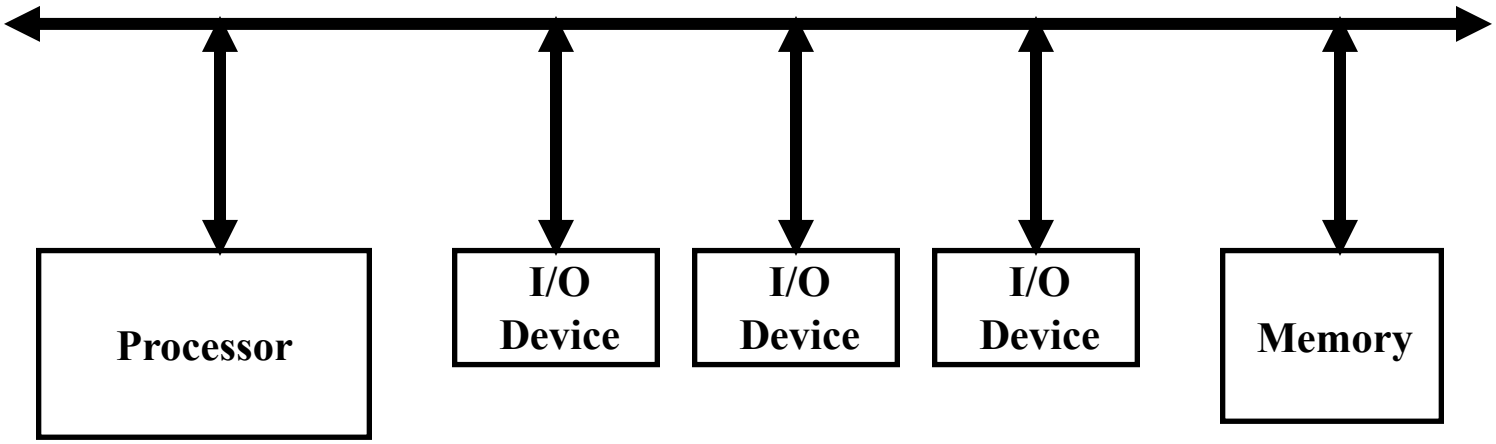
Bus Types

- Processor-Memory buses
  - Short, high speed
  - Design is matched to memory organization
- I/O buses
  - Longer, allowing multiple connections
  - Connect to processor-memory bus through a bridge



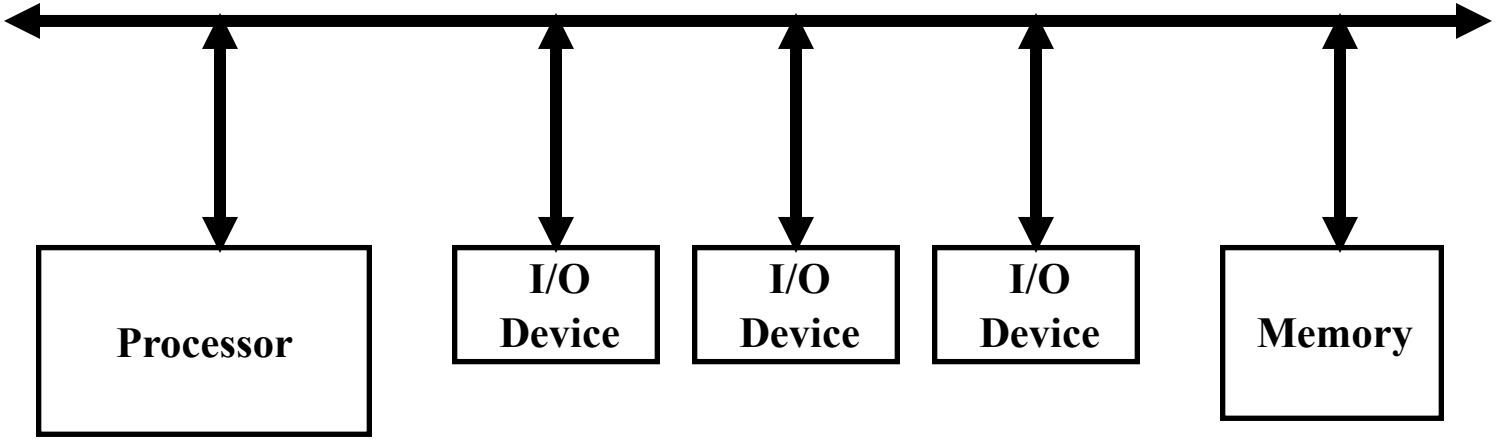


Advantages of Buses



- Versatility:
  - New devices can be added easily
  - Peripherals can be moved between computer systems that use the same bus standard
- Low Cost:
  - A single set of wires is shared in multiple ways

Disadvantage of Buses



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

Bus Basics

- Bus consists of a set of Control Lines and Data Lines.
  - Control Lines :

    Signal requests and acknowledgments

    Indicate what type of information is on the data lines
  - Data Lines : Data / Address / Complex commands
- Bus Transaction
  - A Sequence of Bus Operations that includes a Request and may include a Response.
  - A Transaction is initiated by a single request and may take many individual Bus Operations
  - Two Parts :

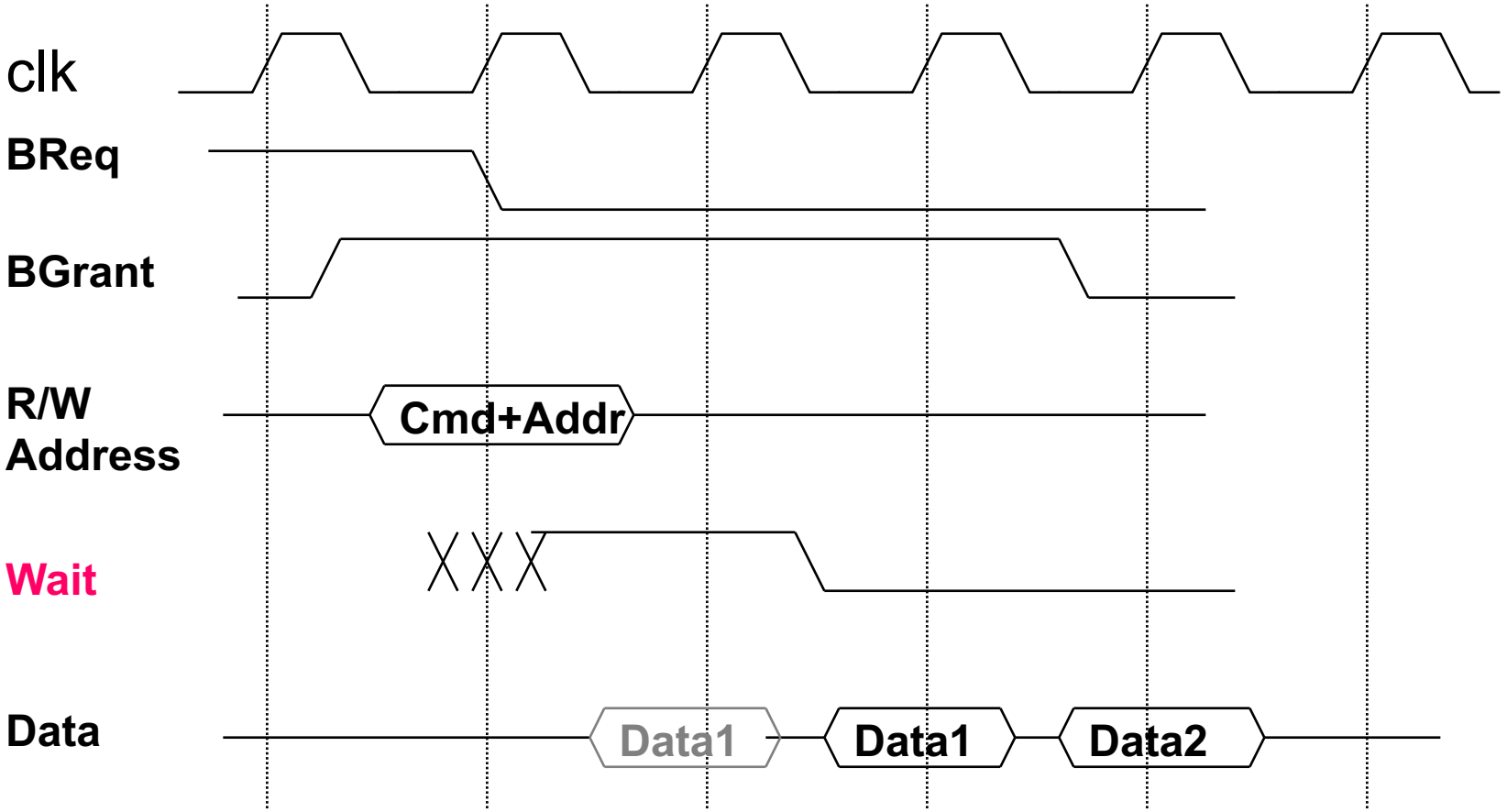
    (1) Sending Address

    (2) Receiving or Sending Data

Synchronous vs. Asynchronous Bus

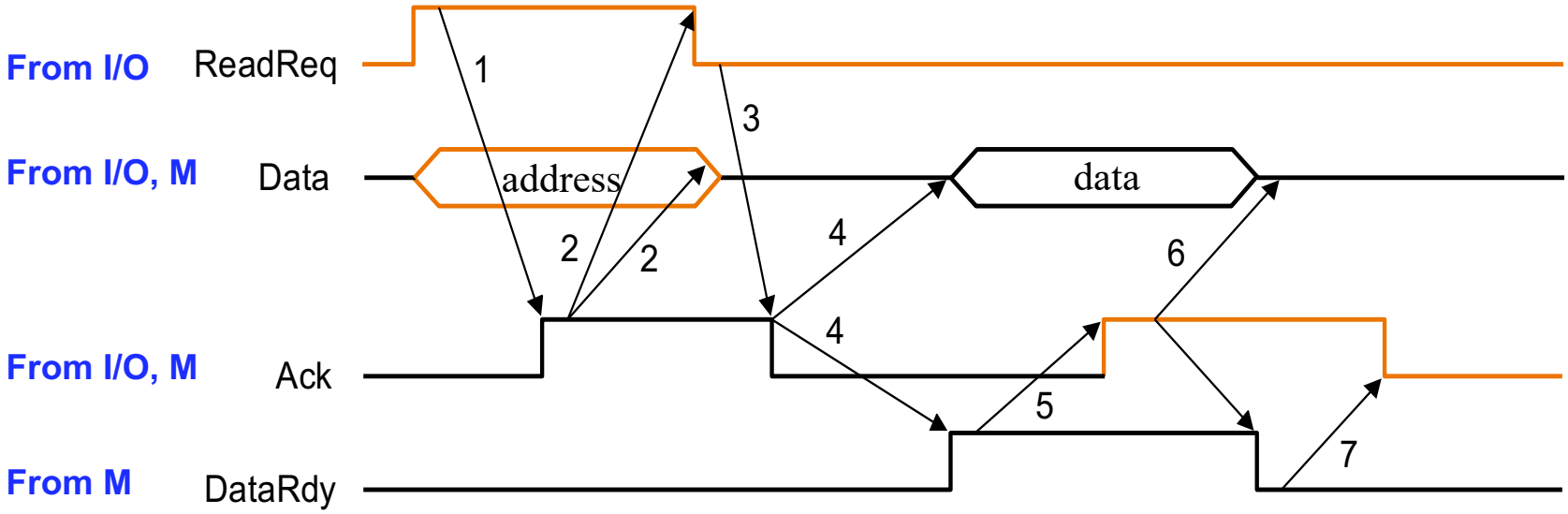
- Synchronous Bus:
  - Includes a clock in the control lines
  - A fixed protocol for communication that is relative to the clock
  - Advantage: involves very little logic and can run very fast
  - Disadvantages:
    - Every device on the bus must run at the same clock rate
    - To avoid clock skew, bus cannot be long if the clock is fast
- Asynchronous Bus:
  - It is not clocked
  - It can accommodate a wide range of devices
  - It can be lengthened without worrying about clock skew
  - It requires a handshaking protocol

Typical Synchronous Protocol



- Slave indicates when it is prepared for data transfer
- Actual transfer goes at bus rate

Asynchronous Bus Example



1. 'ReadReq' from I/O to Mem. I/O sends address.
2. Mem acknowledges 'ReadReq' and I/O releases 'ReadReq' and 'data(address)'.
3. Mem drops 'Ack'.
4. Mem sends 'data' and 'DataRdy'.
5. I/O acknowledge 'data' and 'DataRdy'.
6. Mem releases 'data' and 'DataRdy'.
7. I/O drops 'Ack'.

I/O Management

- Issues :
  - How is a user I/O request transformed into a device command and communicated to the device?
    - Memory Mapped I/O vs. Special I/O Instructions
    - Polling vs. Interrupt
  - How is data actually transferred to or from a memory location?
    - DMA (Direct Memory Access)
  - What is the Role of Operating Systems ?

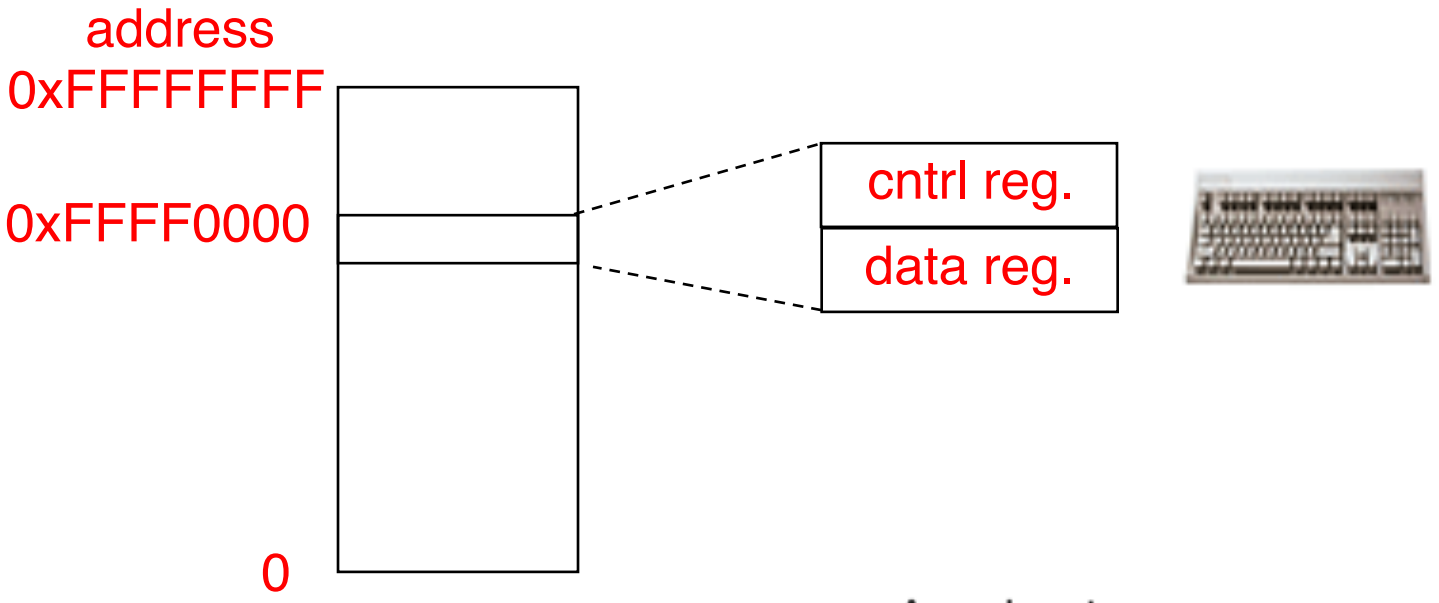
    : The operating system acts as the interface between the I/O hardware and the program that requests I/O

Instruction Set Architecture for I/O

- What must the processor do for I/O?
  - Input: reads a sequence of bytes
  - Output: writes a sequence of bytes
- Some processors have Special Input/Output Instructions
- Alternative model (used by MIPS):
  - Use loads for input, stores for output
  - Called "Memory Mapped Input/Output"
  - A portion of the address space dedicated to communication paths to Input or Output devices (no memory there)

Memory-Mapped IO

- Certain addresses are not regular memory address
- Instead, they correspond to registers in I/O devices

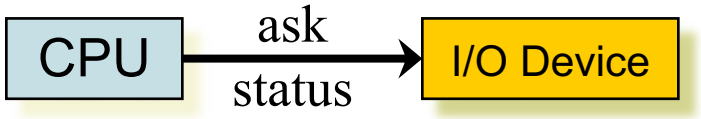


A sample system memory map

Address range (hexadecimal)	Size	Device
0000–7FFF	32 KiB	RAM
8000–80FF	256 bytes	General-purpose I/O
9000–90FF	256 bytes	Sound controller
A000–A7FF	2 KiB	Video controller/text-mapped display RAM
C000–FFFF	16 KiB	ROM

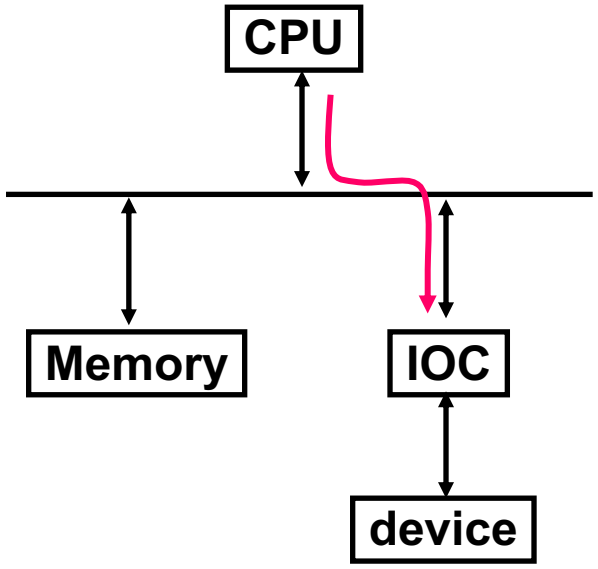
I/O Device Notifying the OS

- The OS needs to know when:
  - The I/O device has completed an operation
  - The I/O operation has encountered an error
- This can be accomplished in two different ways:
  - **Polling**: checking if it is time for next I/O operation.
    - The I/O device put information in a status register
    - The OS periodically check the status register



  - **I/O Interrupt**:
    - Whenever an I/O device completed some operations or needs attention from the processor, it interrupts the processor.

Polling: Programmed I/O



- Advantage:
  - Simple: the processor is totally in control and does all the work
- Disadvantage:
  - Polling overhead can consume a lot of CPU time

Processor Checks Status before Acting

- Path to device generally has 2 registers:
  - Control Register, says it's OK to read/write (I/O ready)
    : think of a flagman on a road
  - Data Register, contains data
- Processor reads from Control Register, waiting for device to set Ready bit in Control reg
  (0 $\Rightarrow$ 1) to say its OK
- Processor then loads from (input) or writes to (output) data register
  - Load from or Store into Data Register resets Ready bit
    (1 $\Rightarrow$ 0) of Control Register

Alternative to Polling

- Wasteful to have processor spend most of its time “spin-waiting” for I/O to be ready
- Would like an unplanned procedure call that would be invoked only when I/O device is ready
- Solution:
  - Use exception mechanism to help I/O.
  - Interrupt program when I/O ready, return when done with data transfer

Exceptions and Interrupts

- “Unexpected” events requiring change in flow of control
  - Different ISAs use the terms differently
- Exception
  - Arises within the CPU
    - e.g., undefined opcode, overflow, syscall, …
- Interrupt
  - From an external I/O controller
- Dealing with them without sacrificing performance is hard

Interrupt (including exception)

- The system needs special attention.
- The transfer of control from a currently running program to an interrupt service routine.
  - To handle CPU effectively (I/O) : compared to Polling
  - When error has occurred.
- Interrupt Handling Procedure

  1) Leave the running procedure
     and save the state of CPU, such as PC
  2) Take necessary actions in response to the particular interrupt.
  3) Restore the state of the CPU and return to the running program.

Handling Exceptions

- In MIPS, exceptions managed by a System Control Coprocessor (CP0)
- Save PC of offending (or interrupted) instruction
  - In MIPS: Exception Program Counter (EPC)
- Save indication of the problem
  - In MIPS: Cause register
  - We'll assume 1-bit
    - 0 for undefined opcode, 1 for overflow

Handler Actions

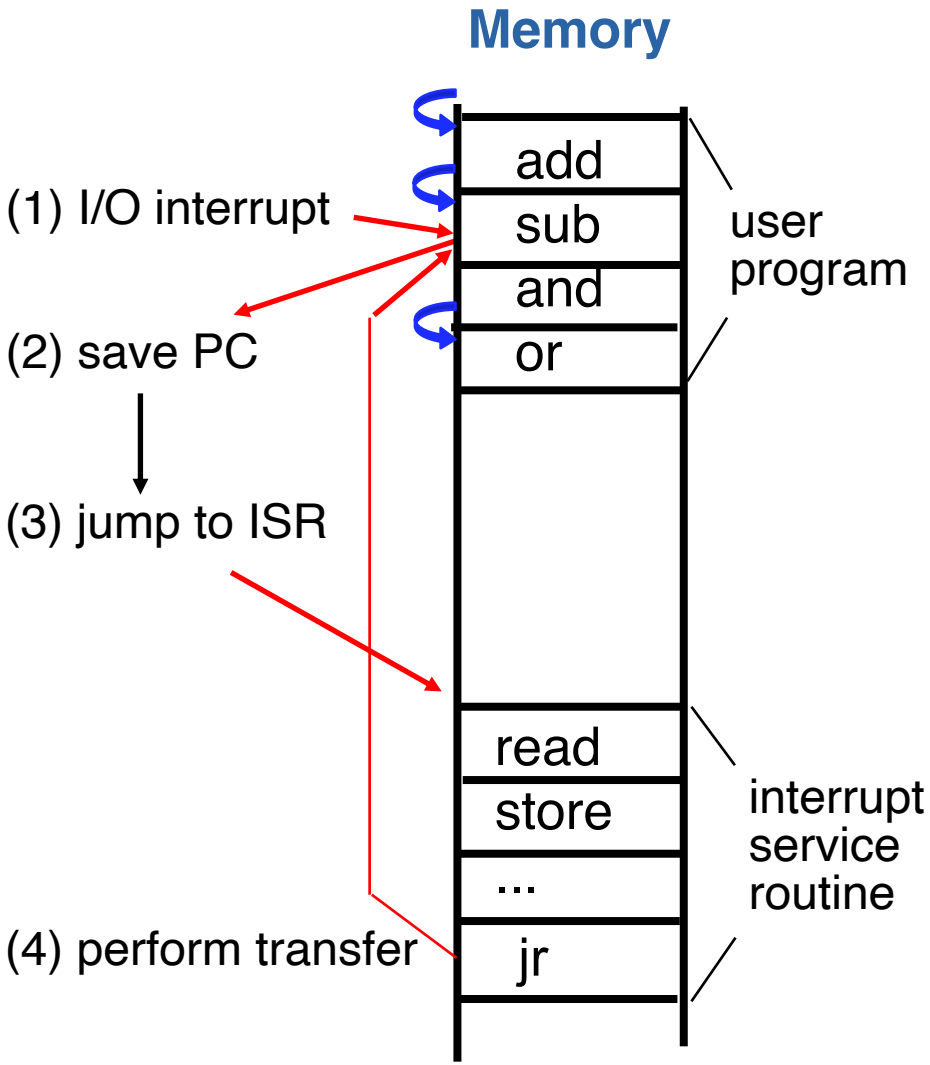
- Read cause, and transfer to relevant handler
- Determine action required
- If restartable
  - Take corrective action
  - use EPC to return to program
- Otherwise
  - Terminate program
  - Report error using EPC, cause, …

I/O Interrupt

- Interrupt is like an exception
  - But not synchronized to instruction execution
  - Can invoke handler between instructions
  - Cause information often identifies the interrupting device

Interrupt Driven Data Transfer

- Advantage:
  - User program progress is only halted during actual transfer
- Disadvantage:
  - special hardware is needed to:
    - Cause an interrupt (I/O device)
    - Detect an interrupt (processor)
    - Save the proper states to resume after the interrupt (processor)



ISR: interrupt service routine

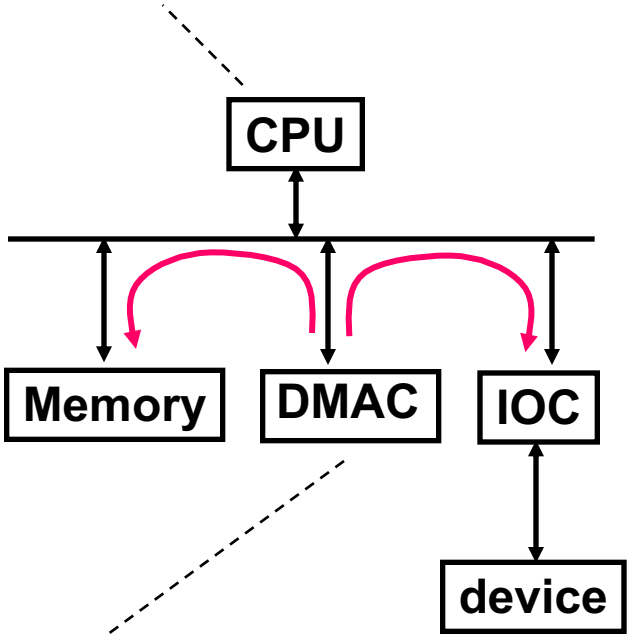
Delegating I/O Responsibility from the CPU: DMA

- In interrupt driven I/O, when transferring large block of data, processors should be involved in transferring the data.

  ➔ DMA is the solution.

- Direct Memory Access (DMA):
  - External to the CPU
  - Act as a master on the bus
  - Transfer blocks of data to or from memory without CPU intervention
  - Appropriate for Block Transfer of High Bandwidth I/O Devices like hard disk

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



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

Summary

- I/O performance is limited by weakest link in chain between OS and device
- I/O Performance Measure
  - Latency : it depends device latency together with latency imposed by memory system or buses
  - throughput : bandwidth
- I/O device notifying the operating system:
  - Polling: it wastes a lot of processor time
  - I/O interrupt: similar to exception except it is asynchronous
- Delegating I/O responsibility from the CPU:
  - DMA
- wide range of devices
  - multimedia and high speed networking poise important challenges

Thanks!