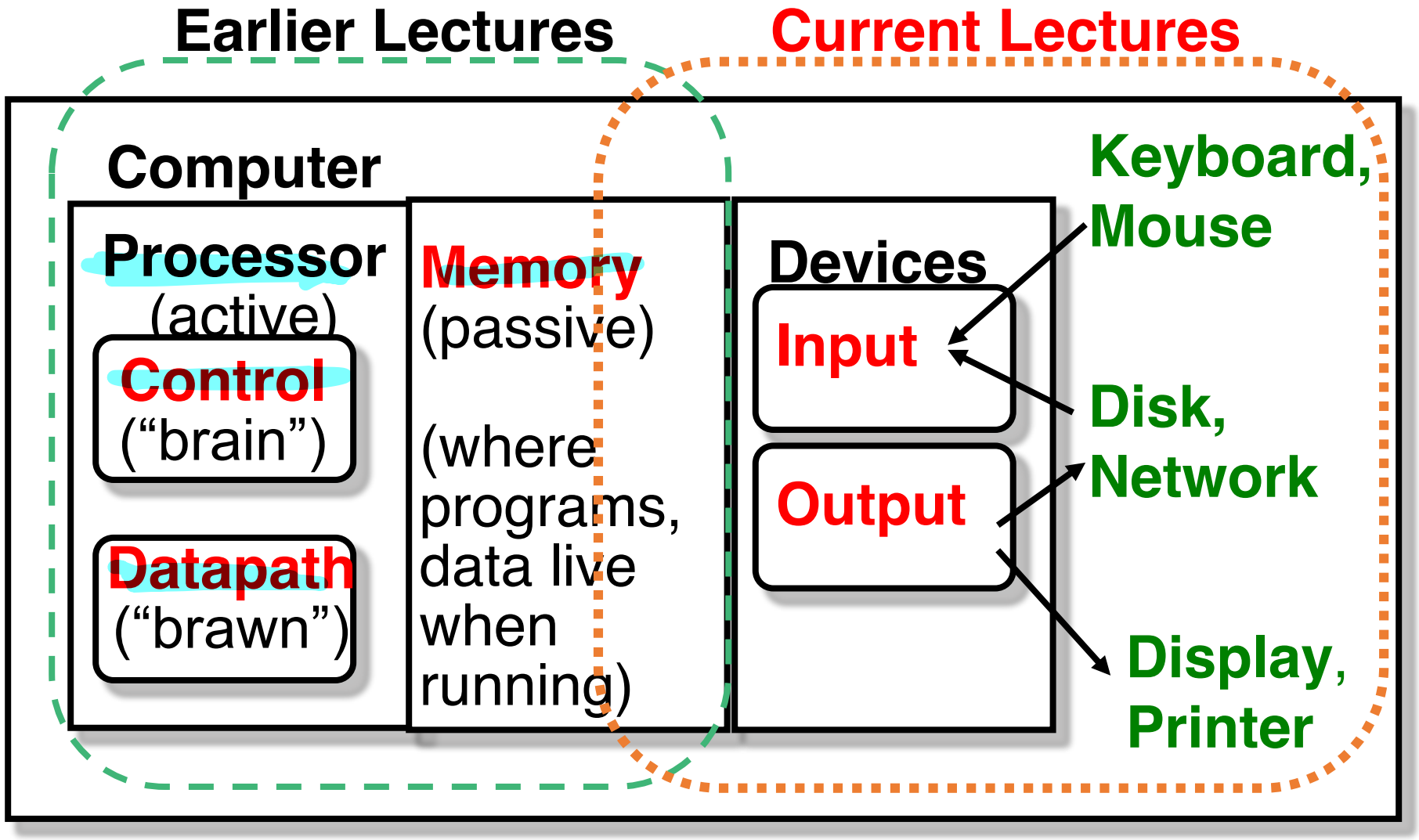
Special Chapter

Input / Output

Computer Architecture and Organization

School of CSEE

Components of Computer



Motivation for I/O Systems

Input & output

- 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 like a car without wheels; great technology, but won't get you anywhere

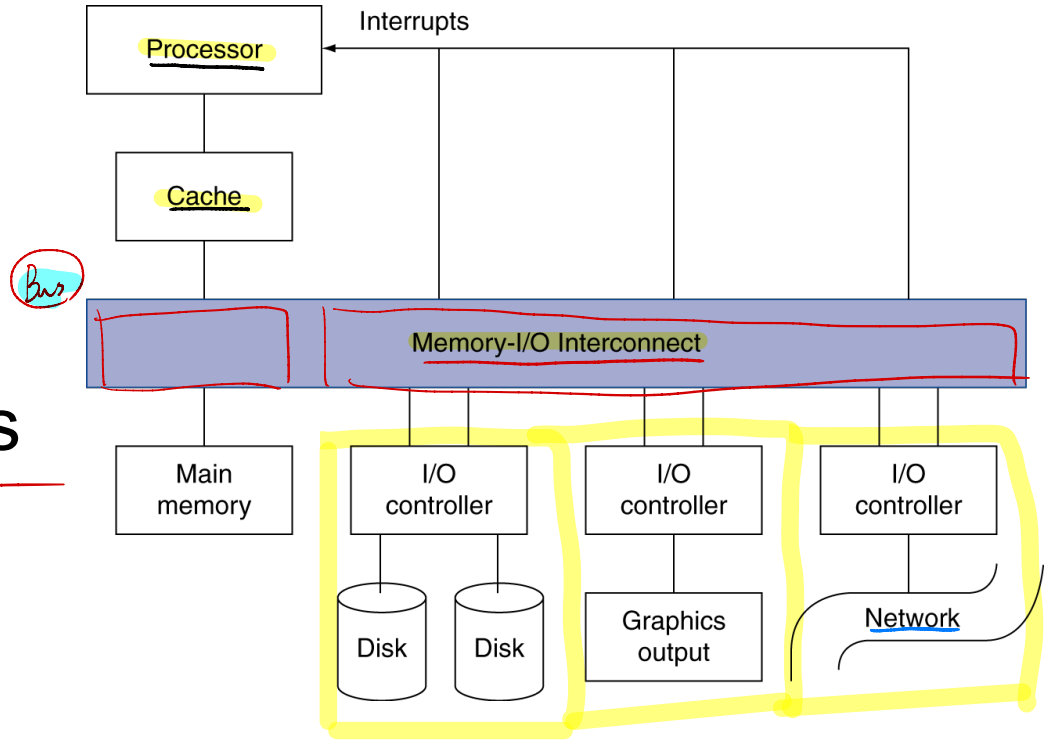
Introduction

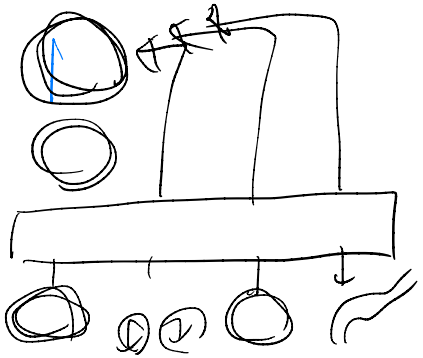
- I/O devices can be characterized by
  - Behavior: input, output, storage
  - Partner: human or machine
  - Data rate: bytes/sec, transfers/sec

Device	Behavior	Partner	Data Rate (KB/s)
Keyboard	Input	Human	0.01
Mouse	Input	Human	0.02
Voice output	Output	Human	5.00
Floppy disk	Storage	Machine	50.00
Laser printer	Output	Human	100.00
Magnetic disk	Storage	Machine	10,000.00
Wireless network	Input or Output	Machine	10,000.00
Graphics display	Output	Human	30,000.00
Wired LAN network	Input or Output	Machine	125,000.00

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

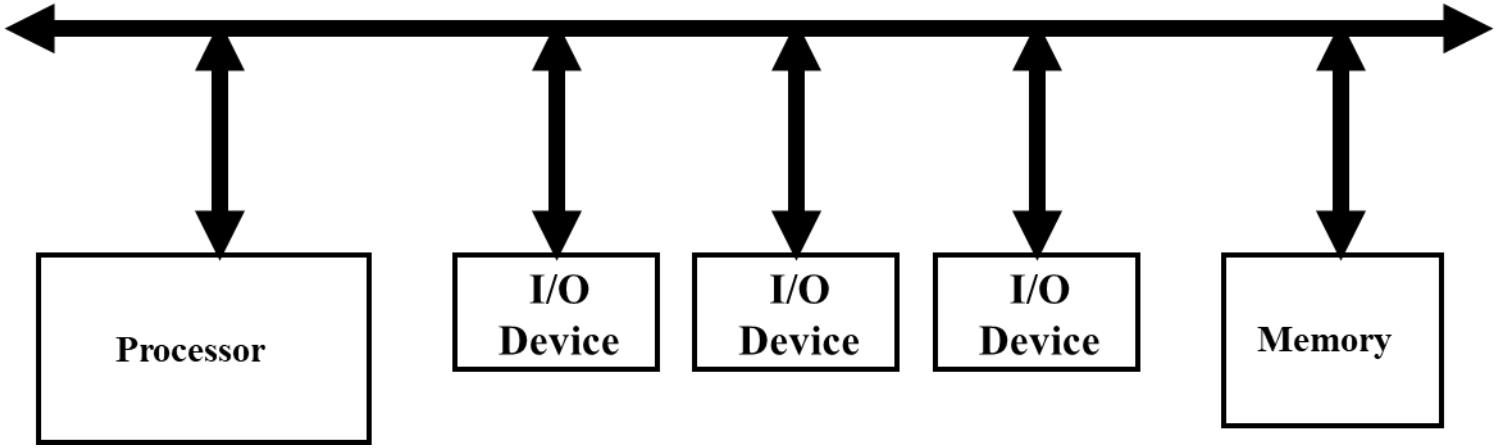




Bus Types

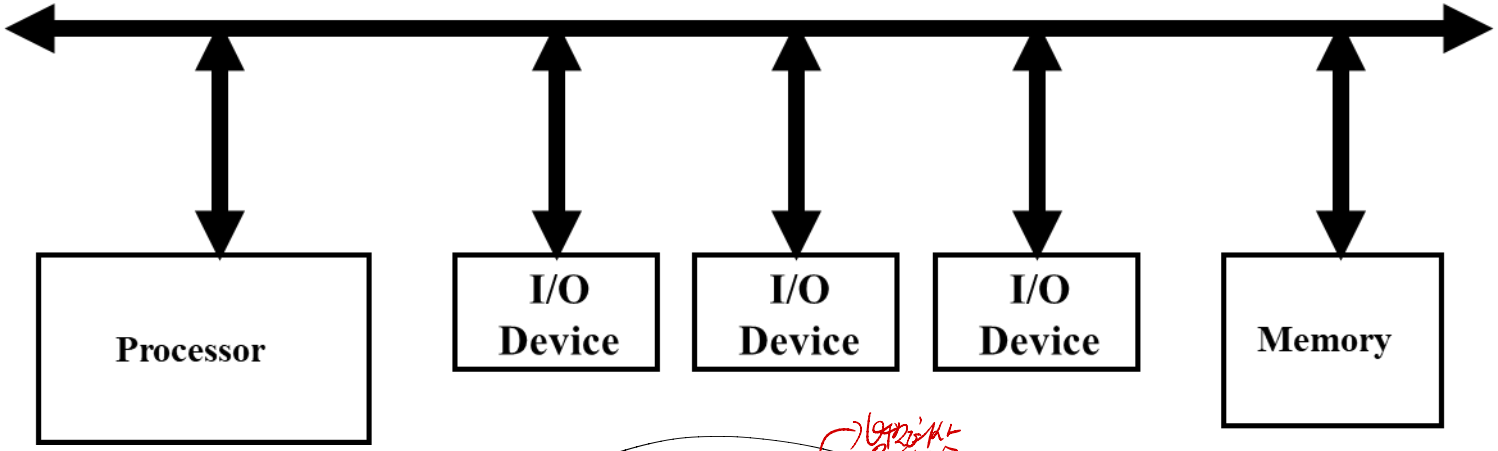
- Processor-Memory buses
  - Short, high speed
  - Design is matched to memory organization
- I/O buses
  - Longer, allowing multiple connections
  - Connect to processor-memory bus through a bridge

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

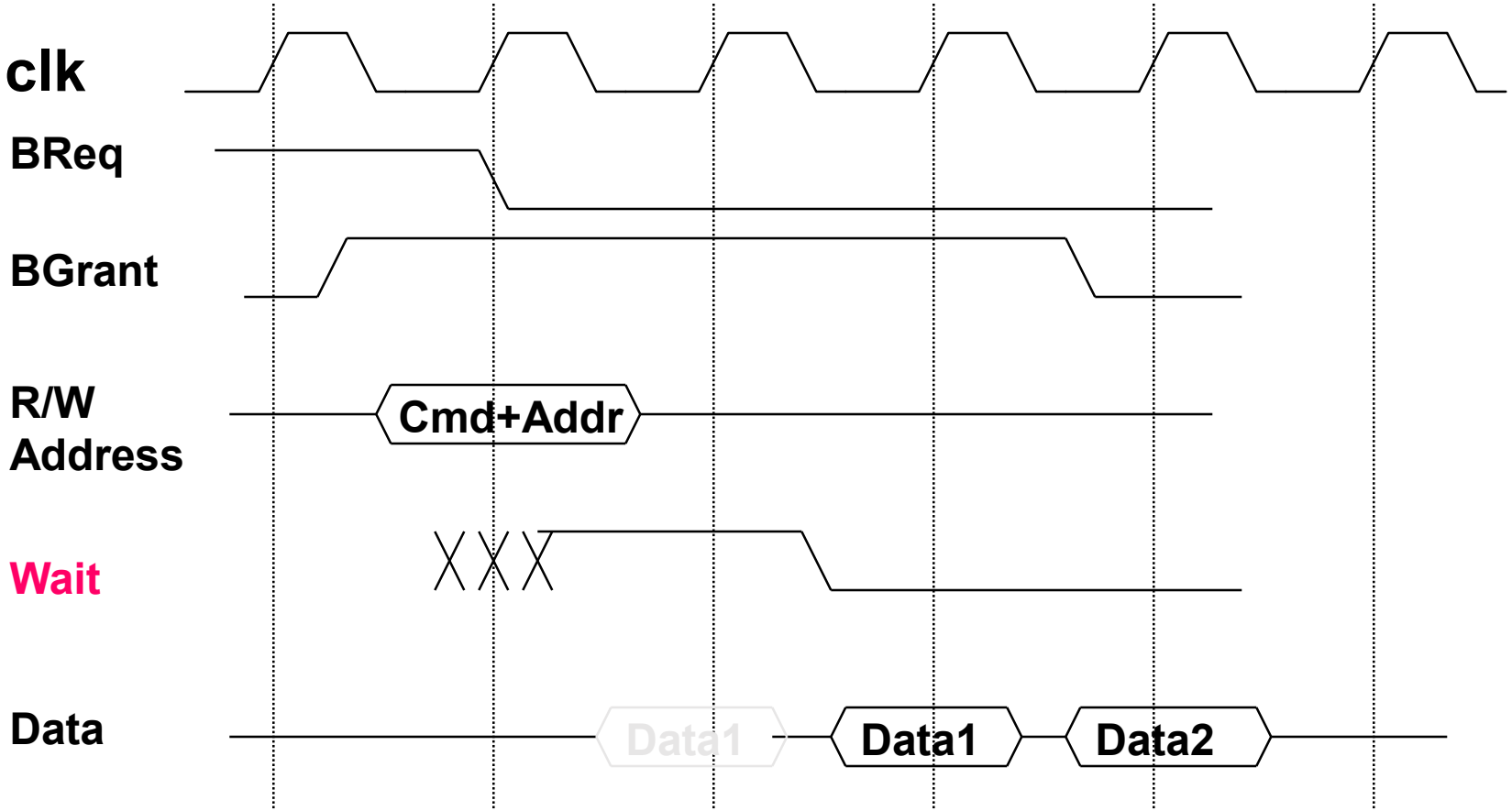
Bus Basics

- 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

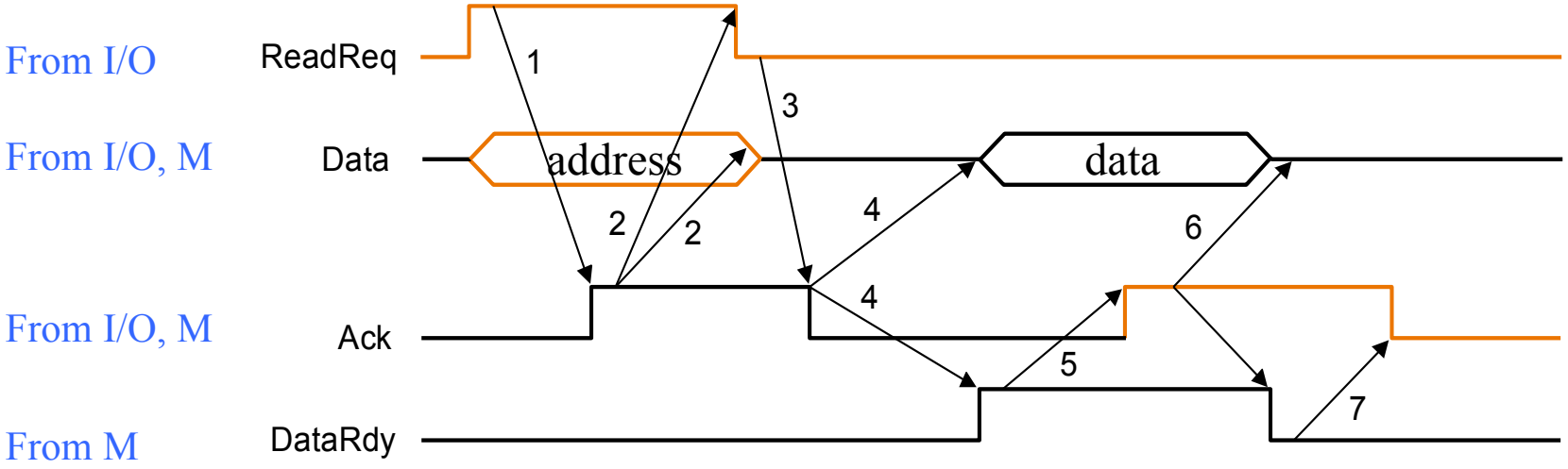
- Synchronous Bus:
  - Includes a clock in the control lines
  - A fixed protocol for communication that is relative to the clock
  - Advantage: involves very little logic and can run very fast
  - Disadvantages:
    - Every device on the bus must run at the same clock rate
    - To avoid clock skew, they cannot be long if they are 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) ⇒ I/O 모듈이 main memory에 직접 접근하고, 이후 CPU에 Interrupt가 발생했음을 알림.
  - 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

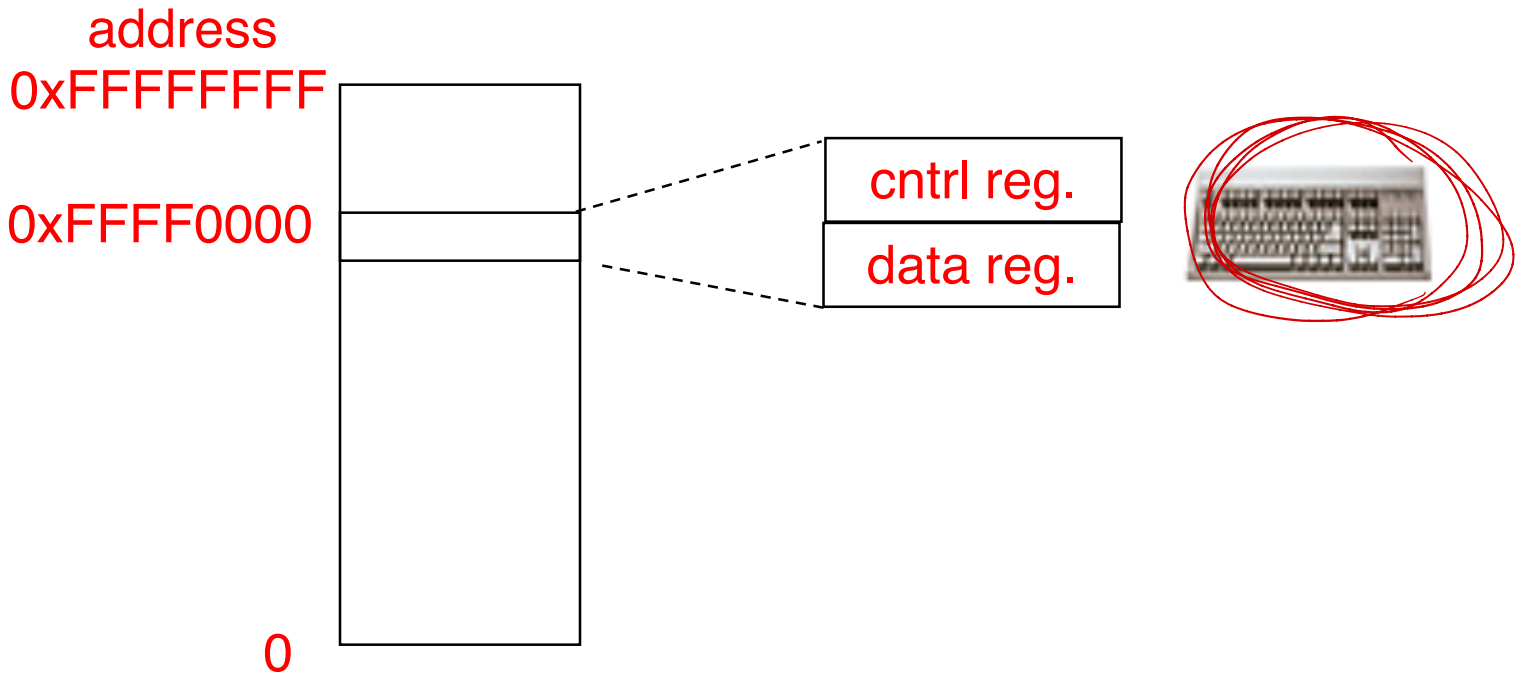
⇒ 현재 I/O 모듈을 담당하는 Processor 명시존재.

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)

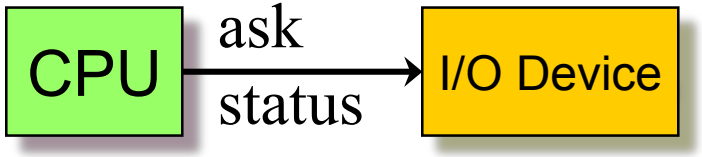
Memory-Mapped IO

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

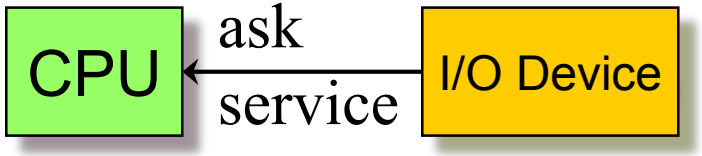


I/O Device Notifying the OS

- 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



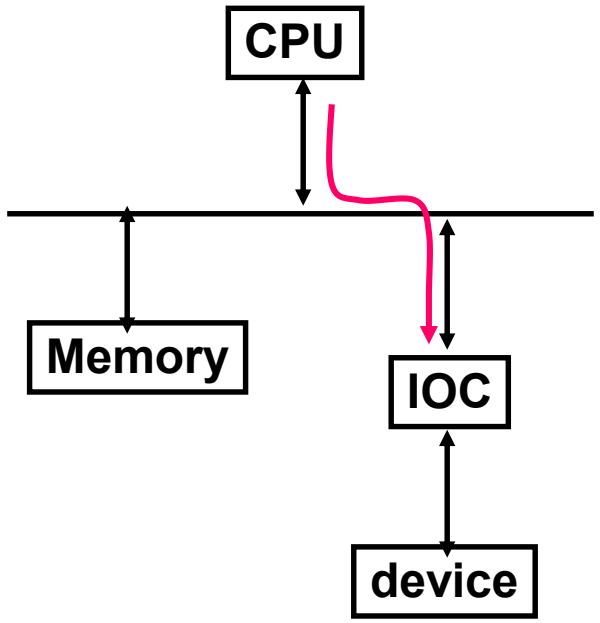
  - I/O Interrupt:
    - Whenever an I/O device completed some operation or needs attention from the processor, it interrupts the processor.



CPU가 열심히 프로그램을 수행하는 과정에서
예외나 I/O 장치등 우선순위가 높은 사건이 발생하면 하던 걸 잠시 멈추고 접수받은 내용을 처리하는 과정

출처: https://ssoonidev.tistory.com/13?category=623915 [심심해서 하는 블로그]

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 ➔ 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 ➔ 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

Type of event	From where?	MIPS terminology
I/O device request	External	Interrupt
Invoke the operating system from user program	Internal	Exception
Arithmetic overflow	Internal	Exception
Using an undefined instruction	Internal	Exception
Hardware malfunctions	Either	Exception or interrupt

Interrupt (including exception)

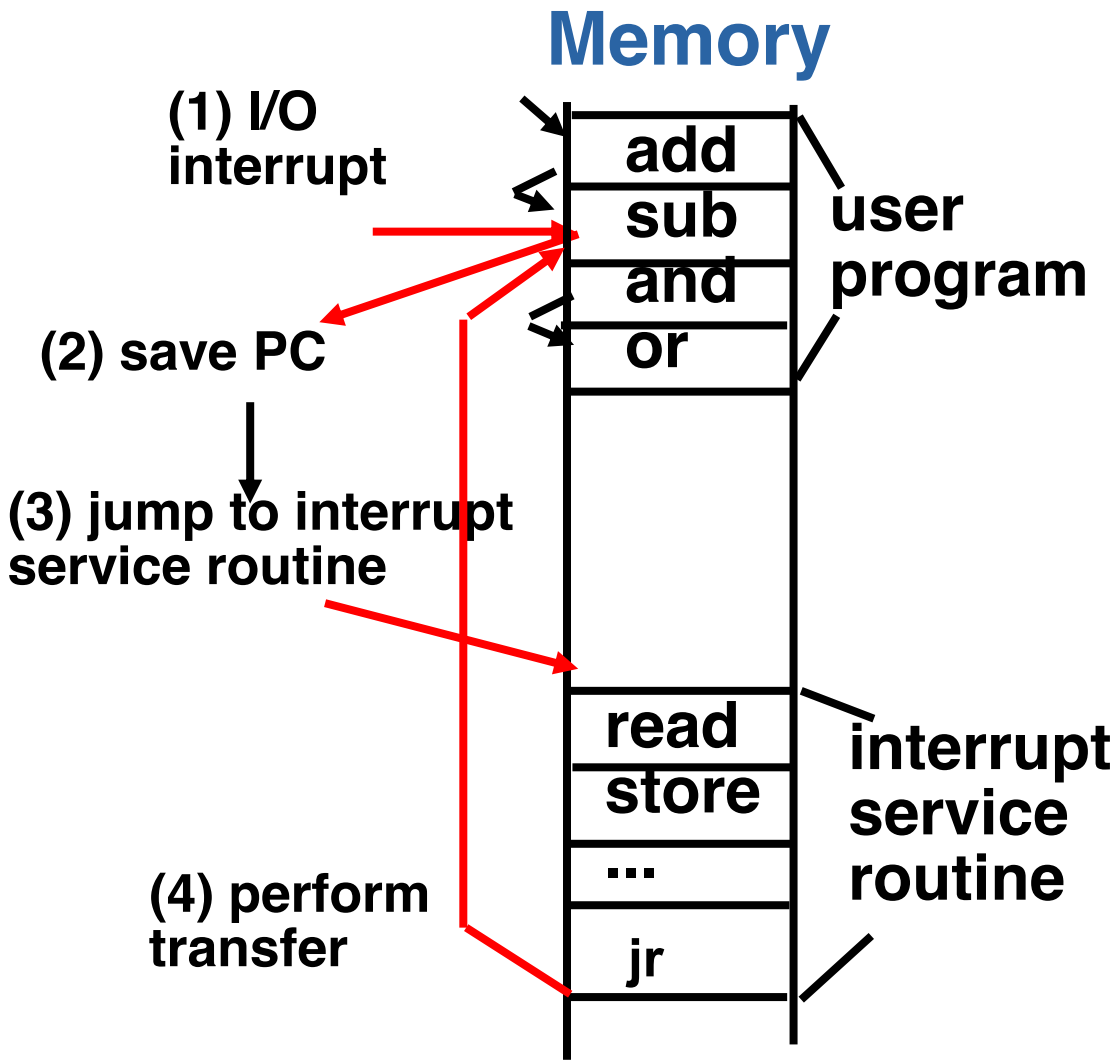
- The system needs special attention. The transfer of control from a currently running program to a interrupt service routine.
  - To handle CPU effectively (I/O) : compared to Polling
  - When error has occurred.
- Interrupt Handling Procedure
  - 1) Leave the running procedure and save the state of CPU, such as PC
  - 2) Take necessary action in response to the particular interrupt.
  - 3) Restore the state of the CPU and return to the running program.

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

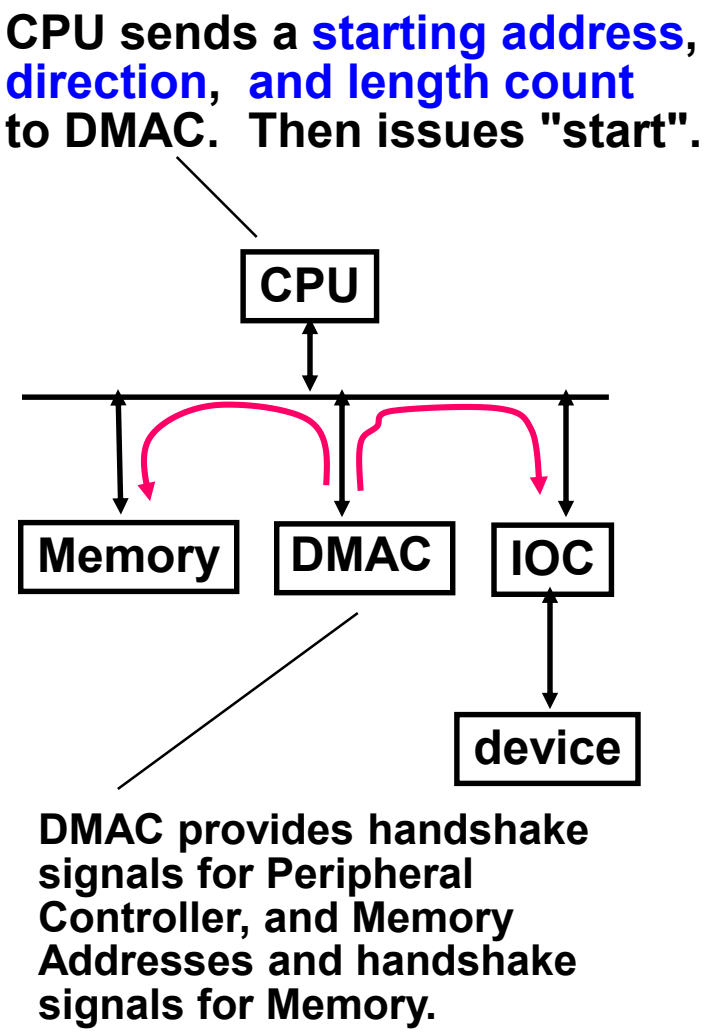
Interrupt Driven Data Transfer

- 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)



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



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 can waste 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

Additional Slides (Revisit)

Handling Exceptions

Revisit

- 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
- Jump to handler $8000\ 0180_{hex}$

An Alternate Mechanism

- Vectored Interrupts
  - Handler address determined by the cause
- Example:
  - Undefined opcode: $8000\ 0000_{hex}$
  - Overflow: $8000\ 0180_{hex}$

- Instructions either
  - Deal with the interrupt, or
  - Jump to real handler

Revisit

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, …