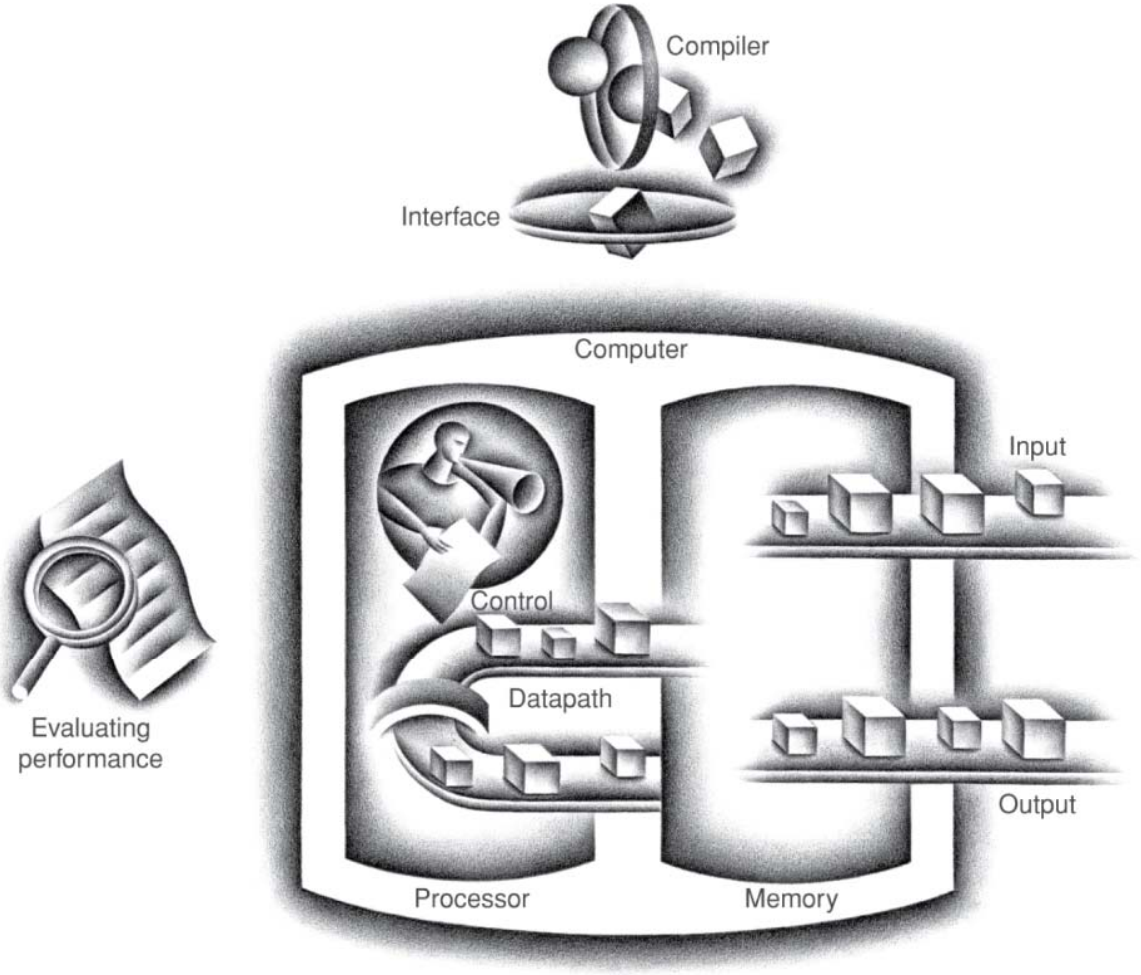
Computer Architecture

Lecture 6

Components of a Computer



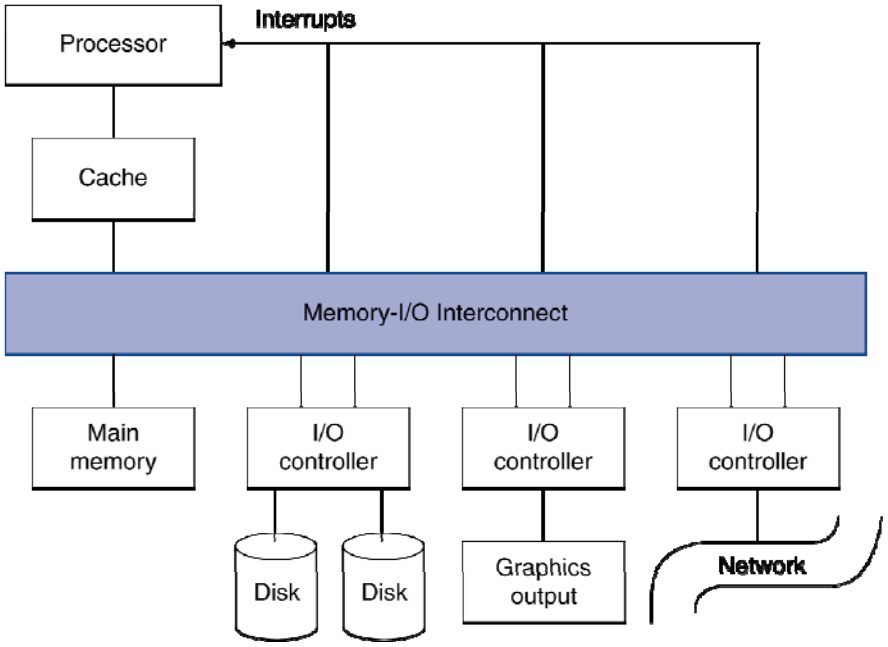
Introduction

- Processors and memory place more focus on performance

- I/O systems place more focus on dependability
  - Networks - communication
  - Disks - storage

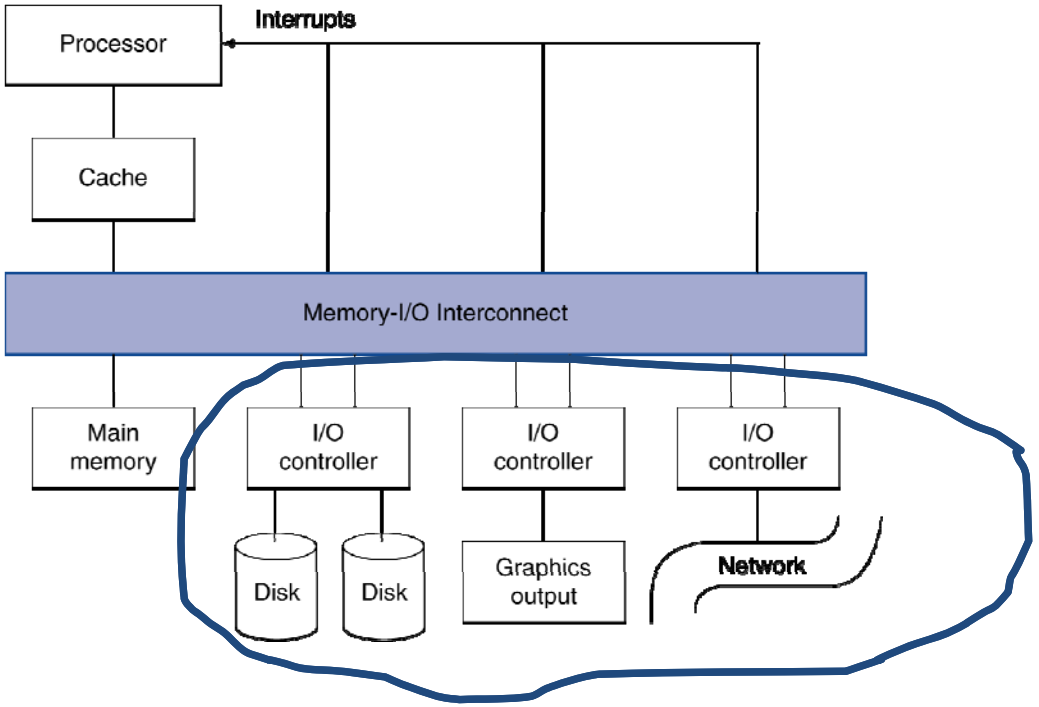
Introduction

- A high-level view of a simple system with I/O



Introduction

- A high-level view of a simple system with I/O



Diverse I/O systems

Introduction

- Performance consideration with I/O is complex because it depends on
  - The device characteristics
  - Connection between the device and the rest of the system
  - Memory hierarchy
  - The OS

Performance measures

- Latency (response time)
  - Regular use of desktop/laptop, the response time for different tasks is more important. Each I/O request is for a small task
- Throughput (bandwidth)
  - In multimedia application, the I/O request is for a long stream of data (e.g., video)

Performance measures

- Desktops & embedded systems
  - Mainly interested in response time & diversity of devices
- Servers
  - Mainly interested in throughput & expandability of devices

Input/Output

- What do we mean by Input/Output ?
- An input operation is inputting a data from a device to the memory
- An output operation is outputting a data from memory to a device
- This means …

Interconnecting Components

- Need interconnections between
  - CPU, memory, I/O controllers

BUS : Interconnecting Components

- Bus: shared communication channel
  - Parallel set of wires for data and synchronization of data transfer
- Advantages of a bus:
  - Easy to add new devices because the same set of shared wires connect all components
  - Low cost due to the shared wires

BUS: Interconnecting Components

- Bus: disadvantage
  - Bottleneck: performance of the system limited by the bus
- Performance limited by physical factors
  - Wire length, number of connections
- More recent alternative: high-speed serial connections with switches
  - Like networks

Bus Signals

- Data lines
  - Carry address and data
  - Multiplexed or separate
- Control lines
  - Indicate data type, synchronize transactions

Bus Type: Synchronous

- Synchronous
  - Uses a bus clock
  - A fixed protocol for communicating that pre-determines the actions in different clock cycle
  - E.g., transmit address and read command in first cycle, memory transmits data on 5th cycle …
- Advantage : Bus can run fast in a predictable fashion
- Disadvantage : Devices must run on same clock rate as bus and there might be clock skew problems

Bus Type: Asynchronous

- Asynchronous
  - Uses request/acknowledge control lines for handshaking
  - They are not clocked – so it is easier to accommodate a wider range of devices
  - No worries about clock skew problem

I/O Management : OS

- I/O is mediated by the OS. Why ?
- Multiple programs share I/O resources
  - Need protection and scheduling
- Role of the OS:
  - OS guarantees that the user programs access only the portions where it has rights. Recall file write/read rights set by the OS
  - OS schedules access to the shared I/O resources

I/O Management : OS

- I/O is mediated by the OS. Why ?
- I/O causes asynchronous interrupts
  - Interrupts cause transfer to kernel mode, must be handled by the OS
- Role of the OS:
  - OS handles the interrupts just like exceptions

I/O Management : OS

- I/O is mediated by the OS. Why ?
- Low-level I/O programming is complex
- Role of the OS:
  - OS provides supplies subroutines to handle low-level I/O commands

I/O Commands

- There must be a way for the OS to communicate with the I/O devices:
- Memory mapped I/O commands
- I/O instructions

Memory mapped I/O

- Portions of address are assigned to I/O devices
- When processor sends a data to such an address, the OS ignores it
- Instead, the I/O device controller interprets the data as a command
- OS uses address translation mechanism to make them only accessible to kernel, and not to the user programs

I/O Commands

- Apart from commands, processor also needs to check the status of the device
- Status registers
  - Indicate what the device is doing and occurrence of errors
  - E.g, 'done' printer has completed printing, 'error' printer is jammed
- Data registers
  - Write: transfer data to a device
  - Read: transfer data from a device

I/O Instructions

- I/O instructions
  - Separate instructions to access I/O registers
  - Can only be executed in kernel mode
  - Example: x86

Communicating with the processor

- Polling versus Interrupt

Polling

- Periodically check I/O status register
  - If device ready, do operation
  - If error, take action
- Common in small or low-performance real-time embedded systems
  - Predictable timing
- In other systems, wastes CPU time

Interrupts

- When a device is ready or error occurs
  - Controller interrupts CPU
- Interrupt is like an exception
  - But not synchronized to instruction execution, i.e., does not prevent instruction execution
  - Conveys information that often identifies the interrupting device

I/O Data Transfer

- Polling and interrupt-driven I/O
  - CPU transfers data between memory and I/O data registers
  - Time consuming for high-speed devices
- Direct memory access (DMA)
  - OS provides starting address in memory
  - I/O controller transfers to/from memory autonomously without involving the processor
  - Controller interrupts on completion or error

DMA/Cache Interaction

- If DMA writes to a memory block that is cached
  - Cached copy becomes stale
- If write-back cache has dirty block, and DMA reads memory block
  - Reads stale data
- Need to ensure cache coherence
  - Flush blocks from cache if they will be used for DMA
  - Or use non-cacheable memory locations for I/O