Announcements

- Programming Assignment #1 is on the web page
- Reading for next week:
  – Chapter 3 (sections 3.2 to 3.9)
- Programming Assignment #0
  – files are available on the web page

Why Study Operating Systems?

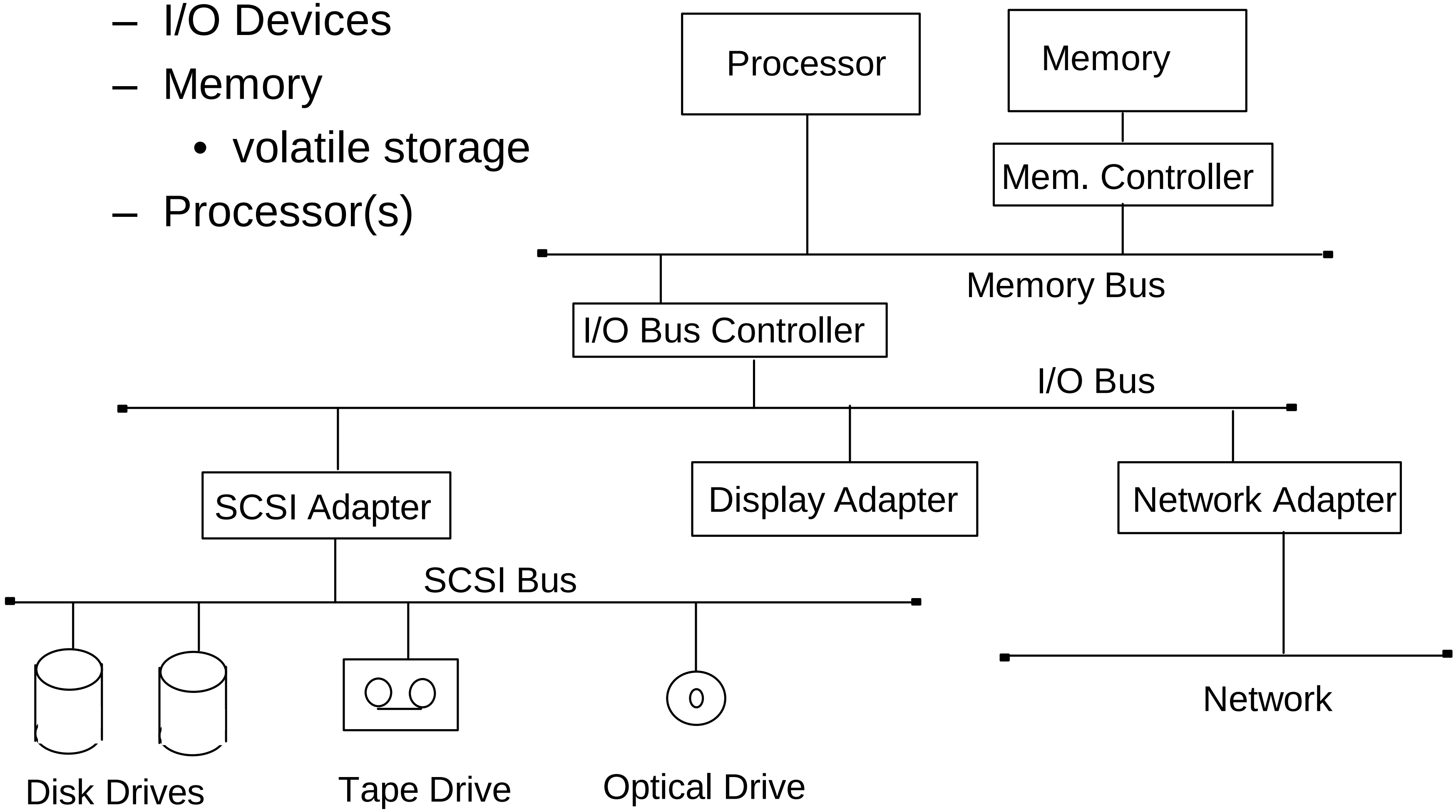
- They are large and complex programs
  - good software engineering examples
- There is no perfect OS
  - too many types of users
    - real-time, desktop, server, etc...
  - many different models and abstractions are possible
    - OS researchers have been termed abstraction merchants
- Many levels of abstraction
  - hardware details: where the bits really go and when
  - high level concepts: deadlock, synchronization

Why Study Operating Systems (cont.)

- Necessity
  - reliability: when the OS is down, computer is down
  - recovery: when the OS goes down it should not take all of your files with it.
- It's fun
  - the details are interesting (at least I think so :)
  - thinking about concurrency makes you better at writing software for other areas

Computer Systems

- Computers have many different devices
  - I/O Devices
  - Memory
    - volatile storage
  - Processor(s)



I/O Systems

- Many different types of devices
  - disks
  - networks
  - displays
  - mouse
  - keyboard
  - tapes
- Each have a different expectation for performance
  - bandwidth
    - rate at which data can be moved
  - latency
    - time from request to first data back

Different Requirements lead to Multiple Buses

- Processor Bus (on chip)
  - > 1Gigabyte/sec
- Memory Bus (on processor board)
  - ~500 megabytes per second
- I/O Bus (PCI, MCA)
  - ~100 megabytes per second
  - buses are more complex than we saw in class
    - show PCI spec.
- Device Bus (SCSI)
  - tens of megabytes per second

Issues In Busses

- Performance
  - – increase the data bus width
  - – have separate address and data busses
  - – block transfers
    - move multiple words in a single request
- Who controls the bus?
  - – one or more bus masters
    - a bus master is a device that can initiate a bus request
  - – need to arbitrate who is the bus master
    - assign priority to different devices
    - use a protocol to select the highest priority item
      - – daisy chained
      - – central control

Disks

- Several types:
  - Hard Disks - rigid surface with magnetic coating
  - Floppy disks - flexible surface with magnetic coating
  - Optical (read only, write once, multi-write)
- Hard Disk Drives:
  - collection of platters
  - platters contain concentric rings called tracks
  - tracks are divided into fixed sized units called sectors
  - a cylinder is a collection of all tracks equal distant from the center of disk
  - Current Performance:
    - capacity: megabytes to tens of gigabytes
    - throughput: sustained < 10 megabytes/sec
    - latency: mili-seconds

I/O Interfaces

- Need to adapt Devices to CPU speeds
- Moving the data
  – Programmed I/O
    - Special instructions for I/O
  – Mapped I/O
    - looks like memory only slower
  – DMA (direct memory access)
    - device controller can write to memory
    - processor is not required to be involved
    - can grab bus bandwidth which can slow the processor down

I/O Interrupts

- Interrupt defined
  - indication of an event
  - can be caused by hardware devices
    - indicates data present or hardware free
  - can be caused by software
    - system call (or trap)
  - CPU stops what it is doing and executes a handler function
    - saves state about what was happening
    - returns where it left off when the interrupt is done
- Need to know what device interrupted
  - could ask each device (slow!)
  - instead use an interrupt vector
    - array of pointers to functions to handle a specific interrupt

I/O Operations

- Synchronous I/O
  - program traps into the OS
  - request is made to the device
  - processor waits for the device
  - request is completed
  - processor returns to application process
- Asynchronous I/O
  - request is made to the device
  - processor records request
  - processor continues program
    - could be a different one
  - request is completed and device interrupts
  - processor records that request is done
  - program execution continues