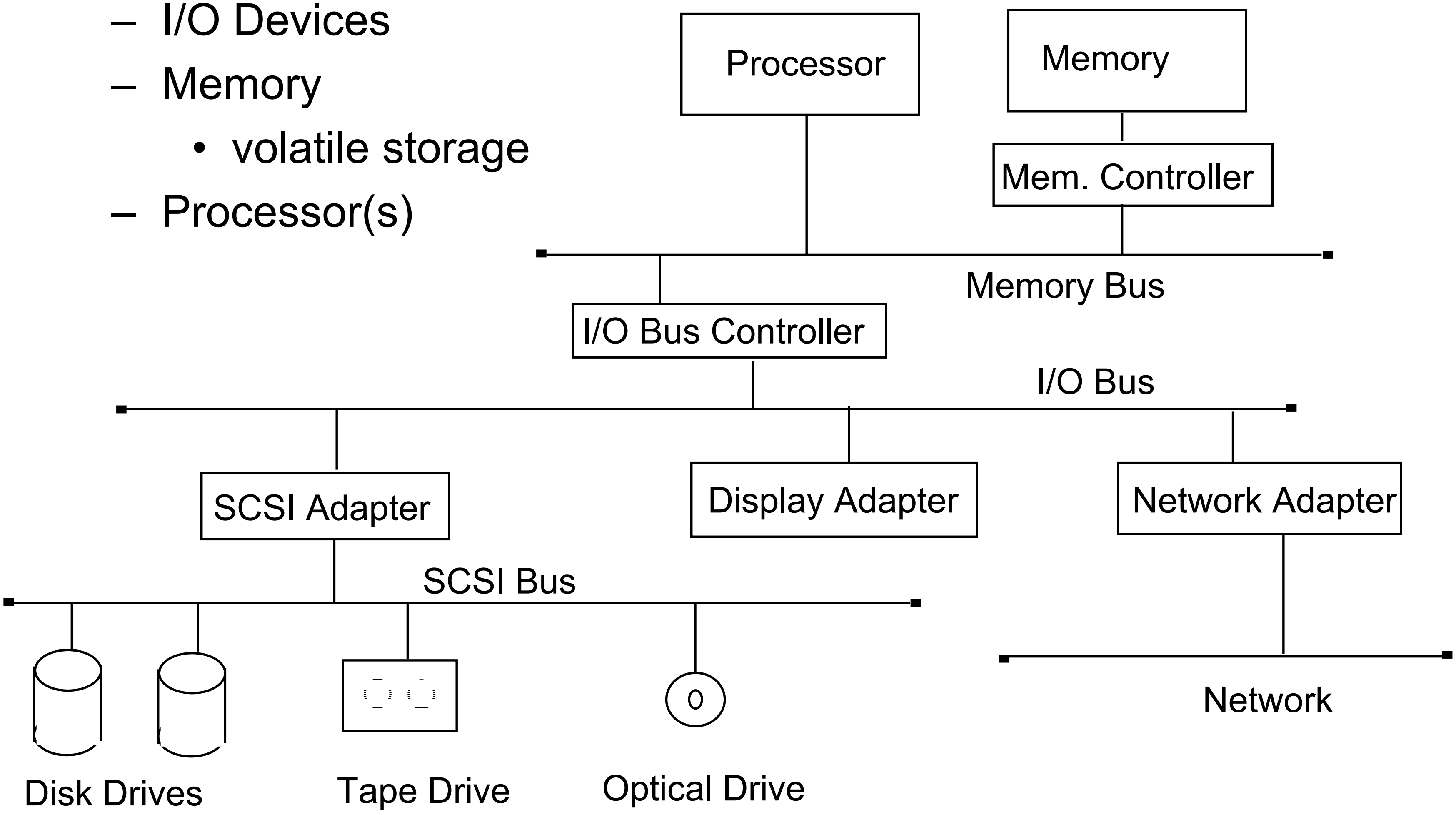
Announcements

- Program #0
  - its due Tuesday
  - See note on web page about update to .bochsrc
- Cell Phones and Pagers
  - Must be to “off” or “vibrate” during class
  - Failure to comply will lower your grade in the class
- Reading
  - Chapter 2
  - Chapter 3 (for Tuesday)

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)
  - Many Gigabytes/sec
- Memory Bus (on processor board)
  - ~1-2 Gigabyte 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, USB)
  - 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 (CDs and DVDs) - 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 hundreds 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

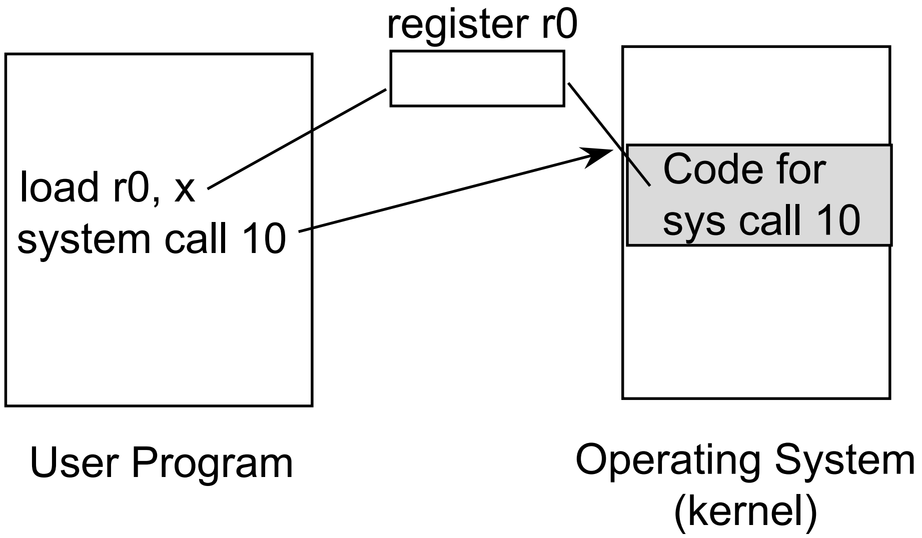
- Synchronous I/O
  - program traps into the OS
  - request is made to the device
  - processor waits for the device
  - request is completed
  - processor returns to application process
- Asynchronous I/O
  - request is made to the device
  - processor records request
  - processor continues program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

Hardware Protection

- Need to protect programs from each other
- Processor has modes
  - user mode and supervisor (monitor, privileged)
  - operations permitted in user mode are a subset of supervisor mode
- Memory Protection
  - control access to memory
  - only part of the memory is available
    - can be done with base/bound registers
- I/O Protection
  - I/O devices can only be accessed in supervisor mode
- Processor Protection
  - Periodic timer returns processor to supervisor mode

System Calls

- Provide the interface between application programs and the kernel
- Are like procedure calls
  - take parameters
  - calling routine waits for response
- Permit application programs to access protected resources



System Call Mechanism

- Use numbers to indicate what call is made
- Parameters are passed in registers or on the stack
- Why do we use indirection of system call numbers rather than directly calling a kernel subroutine?
  - provides protection since the only routines available are those that are export
  - permits changing the size and location of system call implementations without having to re-link application programs

Types of System Calls

- File Related
  - open, create
  - read, write
  - close, delete
  - get or set file attributes
- Information
  - get time
  - set system data (OS parameters)
  - get process information (id, time used)
- Communication
  - establish a connection
  - send, receive messages
  - terminate a connection
- Process control
  - create/terminate a process (including self)