Today

- Review: Computer Organization
- Booting
- Project 1

Review

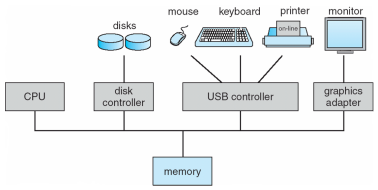
- How did assign0b go?
- What are the two main modes that the OS can run in?
- Why do these two modes exist?
- What mechanism does the OS use to switch between these modes?

Computer Startup

- ***bootstrap program*** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as ***firmware***
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an ***interrupt***

I/O Structure

- After I/O starts, control returns to user program only upon I/O completion
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access)
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing
- After I/O starts, control returns to user program without waiting for I/O completion
  - System call – request to the OS to allow user to wait for I/O completion
  - Device-status table contains entry for each I/O device indicating its type, address, and state
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

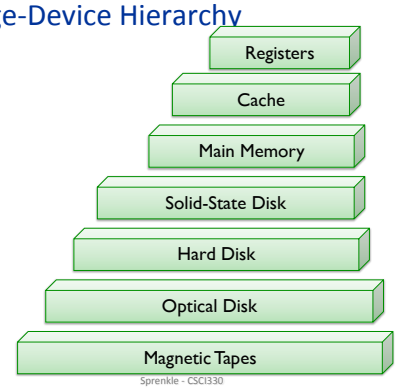
Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - Random access
  - Typically *volatile* – requires power to maintain stored info
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
- Hard disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into tracks, which are subdivided into sectors
  - The disk controller determines the logical interaction between the device and the computer
- Solid-state disks – faster than hard disks, nonvolatile
  - Various technologies
  - Becoming more popular

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- Caching
  - copying information into faster storage system
  - main memory can be viewed as a cache for secondary storage
- Device Driver for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

Storage-Device Hierarchy



Caching

- Performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management: important design problem
  - Cache size and replacement policy

Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Migration of data "A" from Disk to Register

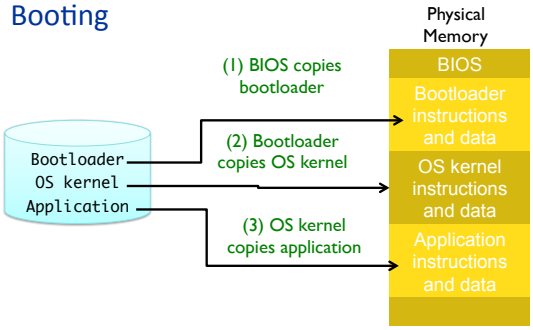
- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy
- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache

BOOTING

System Boot

- When power initialized on system, execution starts at a *fixed* memory location
  - Firmware ROM used to hold initial boot code
- OS must be made available to hardware so hardware can start it
  - Small piece of code – bootstrap loader—stored in ROM or EEPROM locates the kernel, loads it into memory, and starts it
  - Sometimes two-step process where boot block at fixed location loaded by ROM code, which loads bootstrap loader from disk
- Common bootstrap loader, GRUB, allows selection of kernel from multiple disks, versions, kernel options
- Kernel loads and system is then running

Booting



Basic Input/Output System (BIOS)

- A number of small programs and subroutines:
  - Power on self test (POST)
  - System configuration utility
    - Settings stored in small amount of battery backed CMOS memory.
  - A set of routines for performing basic operations on common input/output devices. Such as…
    - Read/write a specified C:H:S from disk.
    - Read character from keyboard.
    - Display character on the screen.
  - OS bootstrap program
- Stored on a Flash ROM that is part of the computer's address space.

Bootstrap Process

- Program Counter (PC) is initialized to the address of the POST program contained in the BIOS
- The last instruction of the POST jumps to the address of the *bootstrap program*, also contained in the BIOS.
- The bootstrap program uses the BIOS routines to load a program contained in the *Master Boot Record* (MBR) of the boot disk into memory at a known address.
  - MBR = first sector on the disk (512 bytes).
  - Boot disk is identified by data stored in the configuration CMOS.
- The last instruction in the bootstrap program jumps to the address at which the MBR program was loaded.
- The MBR program loads the OS kernel.
  - Often indirectly by loading another program (a *secondary boot loader*) that then loads the kernel

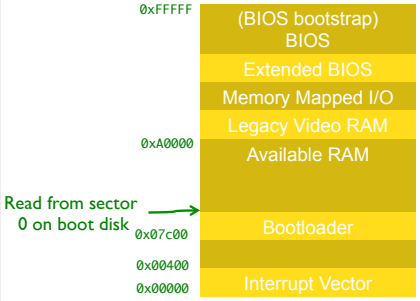
Our Project

- "**Build an operating system from scratch: a project for an introductory operating systems course**" by Michael Black
- 6 Projects: Build on each other
  - Project #1 - Introduction and Booting
  - Project #2 –System calls
  - Project #3 – Loading & Executing Programs + Command Line Shell
  - Project #4 – Writing Files + improved Shell
  - Project #5 – Processes and Multiprogramming
  - Project #6 – OS Enhancements

Intel Architecture

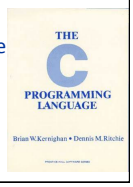
- Bootstrap Process
  - Machine starts in 16-bit real mode
    - 16-bit registers, 20-bit memory addresses
  - Instruction Pointer (IP) initialized to address of BIOS bootstrap
    - 0xFFFF0
  - BIOS bootstrap program runs
    - Loads sector 0 from boot disk at 0x07C00
    - Jumps to 0x07C00

16-bit Real Mode Memory Map



bcc

- We'll be using bcc to compile our programs
  - bcc – Bruce's C Compiler
- Produces 8086 executables that can run in 16-bit real mode
- Understands original K&R C Syntax + a few extensions if the –ansi flag is used
  - K&R = Brian Kernighan and Dennis Ritchie
  - 1978 – *The C Programming Language*



Next Steps

- Project 1: Due next Monday