Today's Plan

Upcoming:

- Quiz #1
- Assignment 1

Last time:

- Introduction to the course

Today's topics:

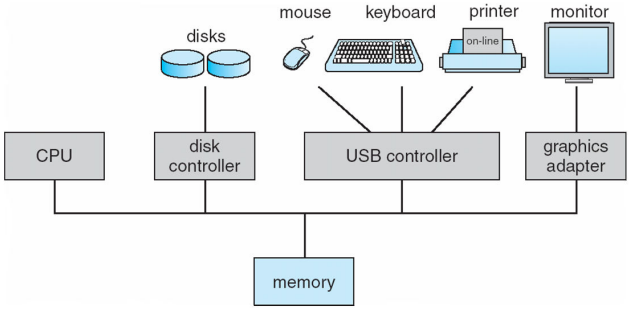
- Computer System Organization
- Interrupts
- I/O Structure
- Storage Structure
- Types of Computer Systems
- Operating System Structure
  - System Calls
  - Hardware Protection

Computer Startup

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

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through a common bus providing access to *shared memory*



Interrupts

- We want the I/O devices and the CPU to be able to execute concurrently
  - The CPU shouldn't have to wait for the MUCH slower I/O device
- The I/O device signals an *interrupt* when it is ready
- When an interrupt occurs, the operating system must:
  -
  -

Interrupts: Controllers vs. Handlers

- A device controller (hardware / firmware) is responsible for moving data between the media and its own local registers
- An interrupt handler (software) is responsible for moving data between the controller registers and memory (so it can be accessed by the user program)
  -

Processing an Interrupt

1. User program has control
2. I/O interrupt occurs
   -
3. User program no longer processed by the CPU
   -
4. Control transferred to interrupt handler
   -

Processing an Interrupt

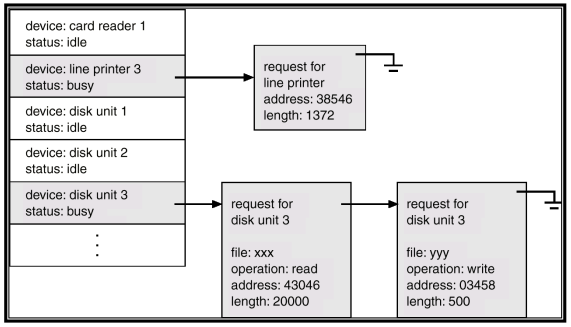
5. Data transferred from controller to memory or from memory to controller depending on whether input or output operation
6. CPU registers restored
7. Control returned to user program
   -

I/O Structure

- The CPU and device controllers operate in parallel
- Two types of I/O operations:
  - *synchronous:* user program waits until I/O operation completes
  - *asynchronous:* user program allowed to continue while I/O operation is in progress
- Asynchronicity is essential for *multiprogramming*
- Asynchronous I/O for **program A** allows the CPU to transfer control to another **program B** until I/O for A is complete.

I/O Structure – Device Status Table

- A device status table keeps track of which devices are busy, what they are doing for which program, and which programs are waiting for access to devices.



I/O Structure – Direct Memory Access (DMA)

- 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.
-

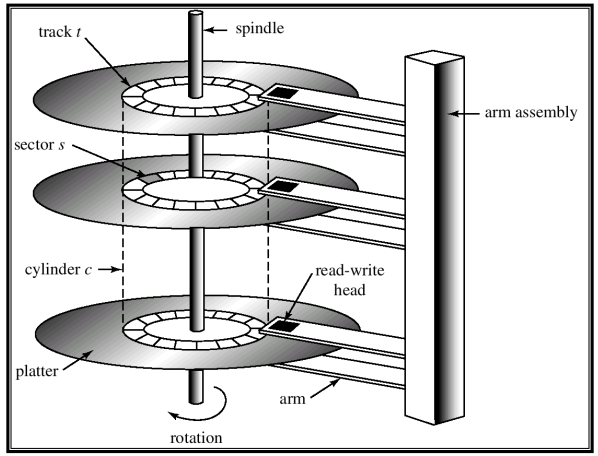
Storage Structure

- Typical (von Neumann) instruction-execution cycle
  - Instruction loaded from *main memory* into CPU
    - Main memory – only large storage media that the CPU can access directly
  - Instruction decoded
    -
  - Instruction executed, repeat
- *Secondary storage* – extension of main memory that provides large nonvolatile storage capacity

Storage Structure – Magnetic Disks

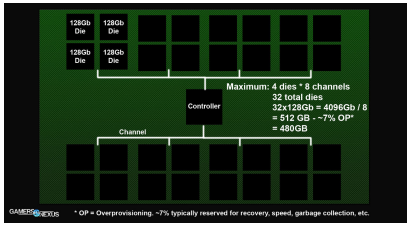
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
  - The *disk controller* determines the logical interaction between the device and the computer.

Storage Structure – Magnetic Disk Architecture



Storage Structure – Solid-State Drives

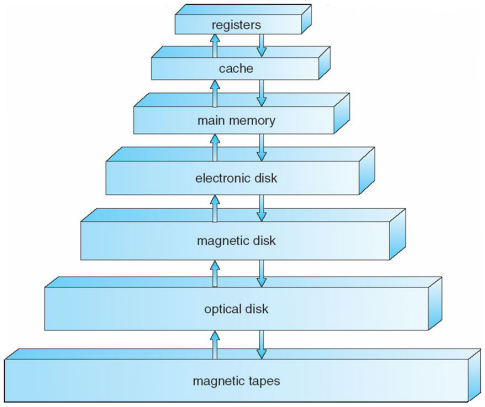
- Solid-State Drives use **NAND Flash** technology
- Bits stored as charges in cells made of transistors
- SLC, MLC, and TLC drives differ in how many bits are stored per cell
- Organization: 8 channels, 4 dies per channel



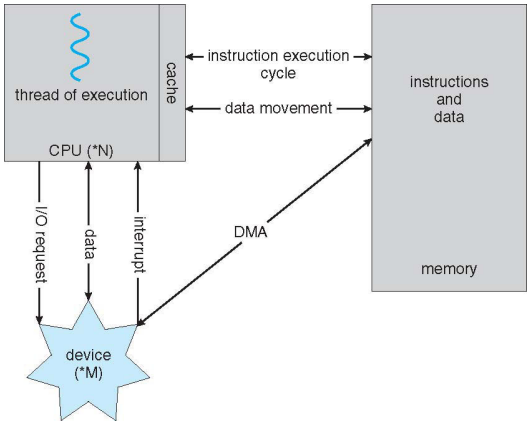
Storage Structure - Caching

- Caching is the use of a smaller, but faster, memory system to speed up a bigger, but slower, memory system
  - The cache holds the data most recently accessed by the CPU
  -
- *Cache management* is an important design problem. A well chosen cache size and replacement policy can result in 80+% of all accesses being in the cache.
- Problem: Consistency
  -

Storage Hierarchy

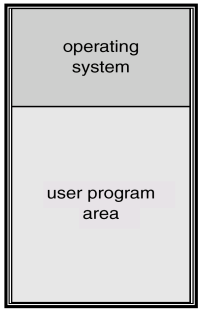


How a Modern Computer Works



Mainframe Systems

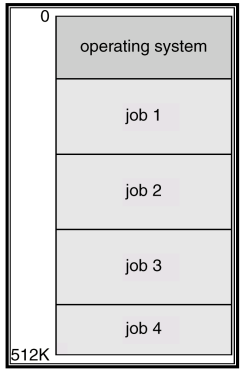
- Mainframes are large, powerful computers designed to handle large numbers of users and jobs
- The first operating systems appeared in mainframes
  - Similar jobs batched together to reduce set-up time
  -
- The OS (called a monitor) transfers control to the job, when the job is done control is returned to the OS



Mainframe Systems

The next step up is the Multiprogrammed Batch System

- Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them
-



Mainframe Systems

Some of the OS features needed for Multiprogramming:

- I/O routine supplied by the system
- Memory management
  -
- CPU scheduling
  -
- Allocation of devices (storage devices, etc.)

Desktop Systems

- *Personal computers* – computer system dedicated to a single user
- I/O devices – keyboards, mice, monitors, printers
- User convenience and responsiveness
- Can adopt technology developed for larger operating systems
  - Usually individuals have sole use of computer and do not need advanced CPU utilization or protection features
- May run several different types of operating systems (Windows, Mac OS/X, UNIX, Linux)

Multiprocessor Systems

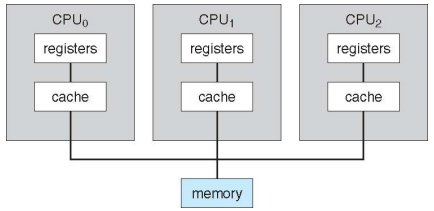
- Multiprocessor systems are systems with more than one CPU in close communication.
  - E.g. multi-core CPUs
- This is a *tightly coupled system* –
- Advantages of parallel system:
  - Increased throughput
  - Economical
  - Increased reliability

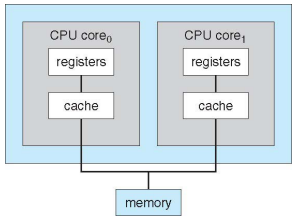
Types of Multiprocessor Systems

- *Symmetric multiprocessing (SMP)*
  - Each processor runs an identical copy of the operating system
  - Many processes can run at once without performance deterioration
  - Most modern Oses support SMP
- *Asymmetric multiprocessing*
  - Each processor is assigned a specific task
  -

Symmetric & Multi-core Processors

Symmetric Multiprocessor:





Dual-core processor

Distributed Systems

- Another idea is to distribute the computation among several physical processors
- This is a *loosely coupled system* –
  - Each processor has its own local memory
  - Processors communicate with one another through communications lines, such as networks (LAN's and WAN's)

Real-Time Systems

- A *real-time system* is one where there are well-defined fixed-time constraints
  - I.e. things need to happen in a reasonable amount of time and in the correct order
- Some examples:

Handheld Systems

Some examples:

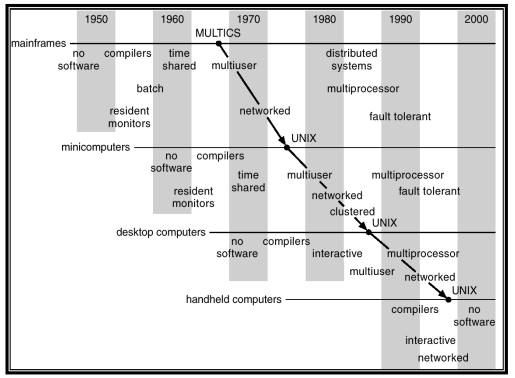
- Personal Digital Assistants (PDAs)
- Smartphones

Some of the issues when designing handheld systems:

-
-
-
-

OS Feature Migration

Operating system features tend to migrate over time from older systems to newer systems.



Operating System Structure

- **Multiprogramming** is needed for efficiency
  - A single user cannot keep CPU and I/O devices busy at all times!
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job is selected and run via **CPU/job scheduling**
  - When a job has to wait (for I/O for example), OS switches to another job

Operating System Structure

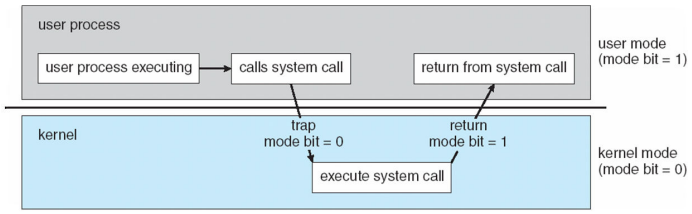
- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
  - Response time should be < 1 second
  - If each user has at least one program executing in memory ⇨
  - If several jobs are ready to run at the same time ⇨
  - If processes don't fit in memory, swapping moves them in and out to run
  - ***Virtual memory*** allows execution of processes not completely in memory

Hardware Protection – Dual Mode

- Sharing system resources requires OS to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- **Dual-Mode operation**: Provide hardware support to differentiate between at least two modes of operations:
  1. *User mode:* Execution done on behalf of the user
  2. *Monitor mode* (also *kernel mode* or *system mode*):
  - Instruction set is restricted in user mode
  - A program, running in user mode, attempting to execute a privileged instruction will cause a trap

System Calls

- **System calls** are used to request services from the OS
  - E.g. give me the current date/time, open a file for reading, etc.
- Executing a system call changes mode to kernel, return from call resets it to user

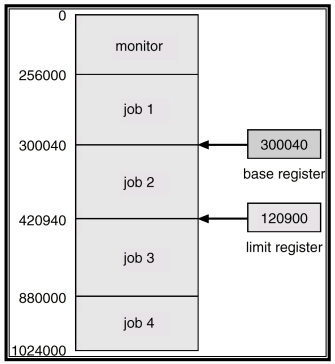


Hardware Protection – I/O & Memory Protection

- We must ensure that I/O devices are protected as well, to have *I/O protection* we ensure:
  -
- We must also ensure that processes (jobs) are not able to access each other's memory space
  - User jobs must also not be able to access the interrupt handlers or interrupt vectors

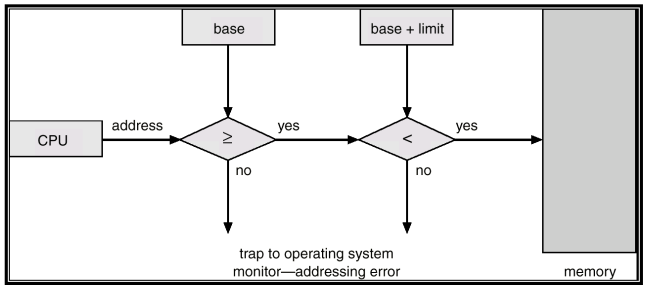
Hardware Protection – Memory Protection

- In order to have *memory protection*, add two registers that determine the range of legal addresses a program may access:
  - **Base Register:**
  - **Limit Register:**



Hardware Protection – Memory Protection

- Checking memory addresses in hardware:



Hardware Protection – CPU Protection

- We must also ensure that no one job is using up all the CPU cycles
- *Timer* – interrupts computer after specified period to ensure operating system maintains control
  -
  -
- The timer is commonly used to implement *time sharing*