BUCKNELL UNIVERSITY
Computer Science

CSCI 315 Operating Systems Design

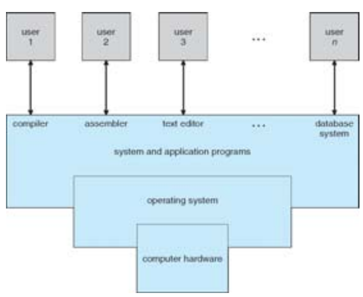
Operating Systems Overview

Notice: This set of slides is based on the notes by Professor Perrone of Bucknell and the textbook authors Silberschatz, Galvin, and Gagne

What is an Operating System?

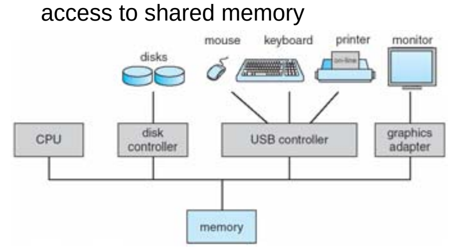
- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  – Execute user programs
  – Make the computer system convenient to use
  – Use the computer resources (e.g., memory, storage, CPU) in an efficient manner

Four Components of a Computer System



Computer System Organization

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



What Operating Systems Do

- Depends on the point of view
  – Users want convenience, **ease of use**
  – System managers and owners want **efficiency** use of resources such as CPU time, memory, and storage; ultimately saving money and serving business

Operating System Definition

- OS is a **resource allocator**
  – Manages all resources
  – Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  – Controls execution of programs to prevent errors and improper use of the computer

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
  - Linux command "ps –aef | less" to search for process 0 and process 1

Computer-System Operation

- I/O devices and the CPU can execute in parallel
- 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 (e.g., by setting the interrupt bit in a register)

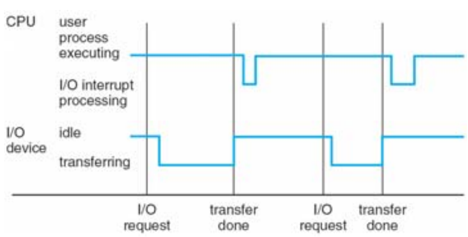
Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request (e.g., I/O, see MIPS *syscall*)
- An operating system is **interrupt driven**

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter (where?)
- Determines which type of interrupt has occurred (e.g., check interrupt register)
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt Timeline



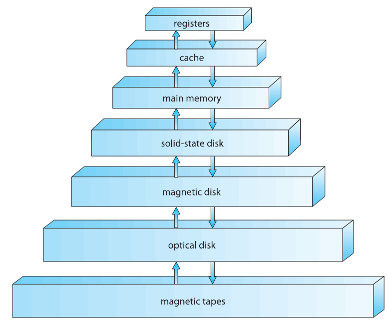
I/O Structure

- After I/O starts, control returns to user program only upon I/O completion (e.g., keyboard input, or reading file from disk)
  - The user program is taken off the CPU
  - When the I/O is complete, the device (keyboard, or disk controller) sends an interrupt to the OS
  - The OS saves the current process, handles the interrupt, the system continues …

Direct Memory Access (DMA)

- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention, once a request is received
- Only one interrupt is generated per block of data, rather than the one interrupt per byte

Storage-Device Hierarchy



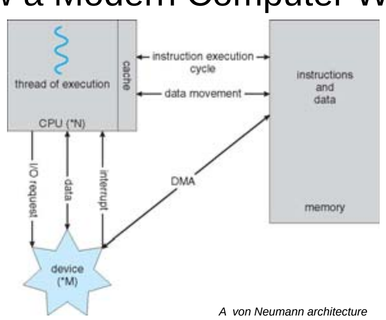
Storage Structure

- Storage systems organized in hierarchy
  - Speed: fast to slow (register, cache, memory …)
  - Cost: expensive to cheap (register, cache, memory …)
  - Volatility:
    - Maintain data with power, registers, cache, main memory
    - Maintain data without power, solid state devices, magnetic and optic disks

Caching

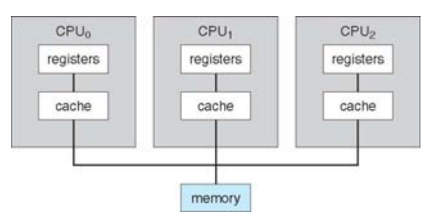
- Important concept, performed at many levels in a computer (in hardware, operating system, software)
- Faster storage (cache) checked first to determine if information is there
  - Yes: information used directly from the cache
  - No: data copied to cache from next level storage for use and keep it in the cache
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

How a Modern Computer Works



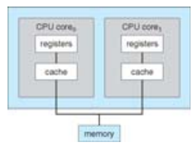
A von Neumann architecture

Symmetric Multiprocessing Architecture



A Dual-Core Design

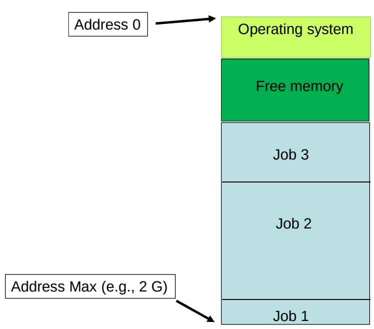
- **UMA** and **NUMA** architecture variations
- Multi-chip and **multicore**
- Systems containing all chips vs. **blade servers**
  - Chassis containing multiple separate systems



Operating System Structure

- **Multiprogramming** needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via ***job scheduling***
  - When it has to wait (for I/O for example), OS switches to another job
- **Timesharing** (**multitasking**) is logical extension in which CPU switches jobs so frequently that users can interact with its job while it is running, creating ***interactive*** computing

Memory Layout for Multiprogrammed System

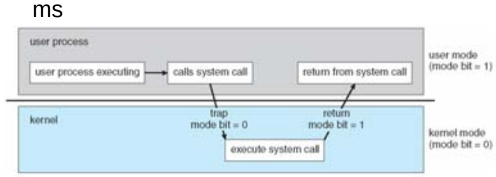


Operation Mode

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e., **virtual machine manager** (**VMM**) mode for guest **VMs**

User Mode and Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Set the timer for specific period
  - Generate interrupt when the timer expires
  - System clock interrupts every "tick", e.g., 10 ms



Operating System Components

- Process management
- Memory management
- Storage/file management
- I/O system
- Protection and security

Types of Operating Systems

- General purpose OS: *e.g., Windows, Linux*
- Real time operating systems: *e.g., airline reservation systems or control systems which has fixed response time requirement*
- Embedded Operating System: *e.g., Android for cell phones and tablets*