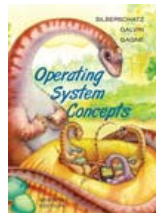
Chapter 1 (Part 2): Introduction

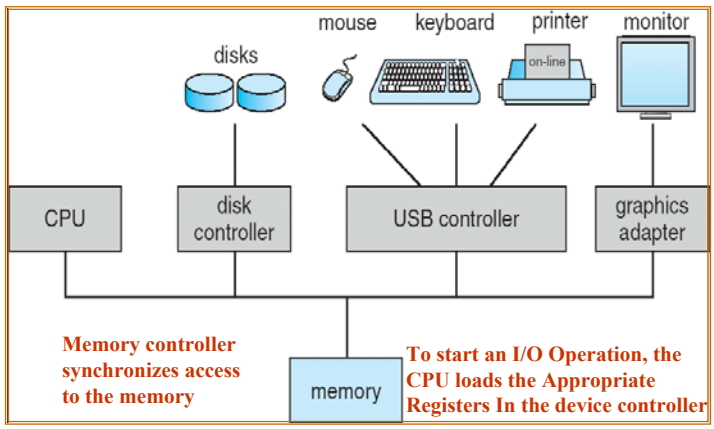
- Computer System Organization
- OS Operations
- Process Management
- Memory Management
- Storage Management
- The Von Neumann Architecture



© Silberschatz et al, *Operating System Concepts 7/e*, Wiley, © 2005

A Modern Computer System

(bootstrap, interrupt)



Computer System Organization

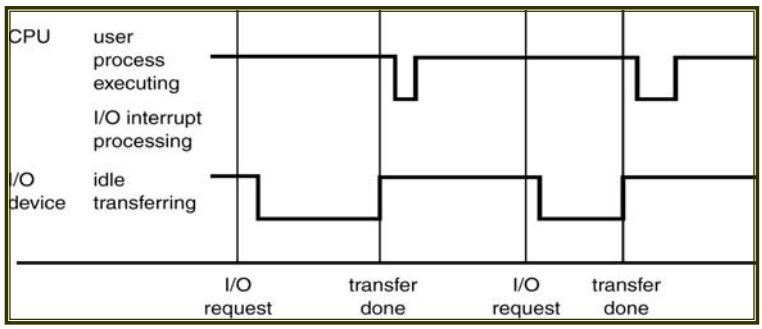
- Computer System Operation
  - I/O devices and the CPU can execute concurrently.
  - Each device controller is in charge of a particular device type.
  - Each device controller has a local buffer.
  - CPU moves data from/to main memory to/from local buffers
  - I/O is from the device to local buffer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

Computer System Organization (cont.)

- Computer System Operation (cont.)
  - Common Functions of CPU Interrupts
    - Bootstrap (or kernel booting/waiting events or interrupts)
    - Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines (hardware/software interrupt) (CPU waits and resumes only when interrupts service routine finishes execution).
    - Interrupt architecture must save the address of the interrupted instruction.
    - Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt.
    - There exits hardware interrupts (such as those coming from device controllers) and software interrupts which are triggered by executing a system call.

Computer System Organization (cont.)

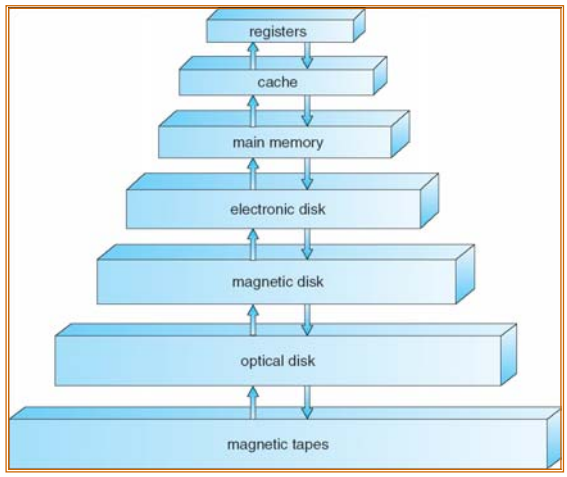
- Computer System Operation (cont.)
  - Interrupt Handling
    - The operating system preserves the state of the CPU by storing registers and the program counter.
    - Determines which type of interrupt has occurred:
      - vectored interrupt system
        (Table of pointers to interrupt routines)
    - Separate segments of code in the OS determine what action should be taken for each type of interrupt



Interrupt Time Line for
a Single Process Doing Output

Computer System Organization (cont.)

- Storage Structure
  - Main memory – only large storage media that the CPU can access directly (millions to billions bytes).
  - Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
    - 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.



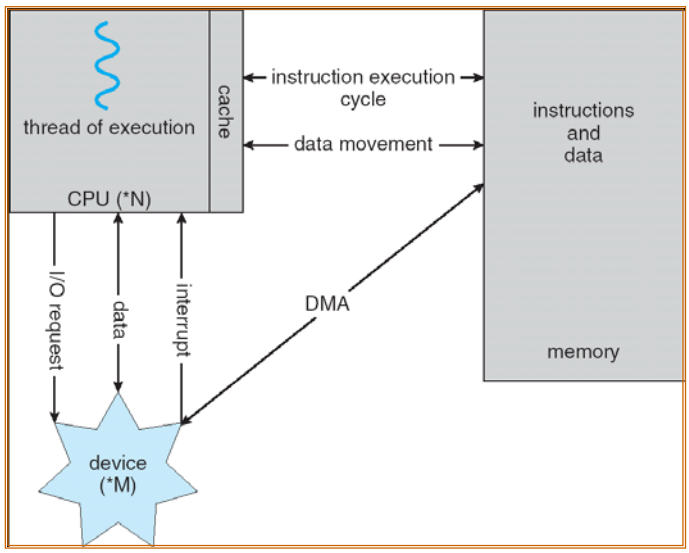
Storage-Device Hierarchy

Computer System Organization (cont.)

- I/O Structure
  - To start an I/O operation, the device driver loads the appropriate registers within the device controller.
  - The device controller examines the contents of these registers to determine what action to take (such as “read a character from the keyboard”).
  - The controller starts the transfer of data from the device to its local buffer.
  - Once the transfer of data is complete, the device controllers informs the device driver via an interrupt that it has finished its operation.
  - The device driver then returns control to the operating system.

Computer System Organization (cont.)

- I/O Structure (cont.)
  - Direct Memory Access Structure (DMA)
    - Used for high-speed I/O devices able to transmit information at close to memory speeds.
    - Characters in a terminal are being transmitted at close to memory speeds, CPU will be “smothered” by several interrupts-→ no user process execution is possible !
    - Device controllers circumvent the CPU by sending an entire block of data from buffer storage to the memory (one interrupt per block rather than one per byte !)



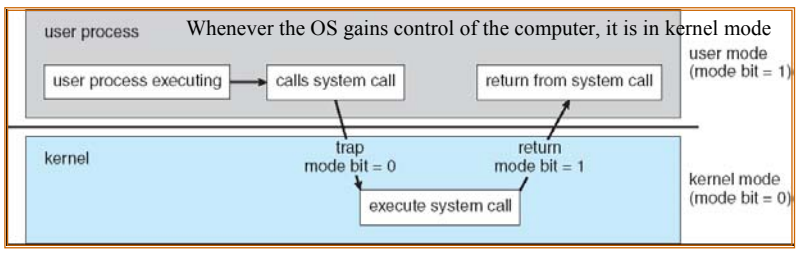
How a modern computer system works

OS Operations

- Dual-Mode Operation
  - Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
  - Provide hardware support to differentiate between at least two modes of operations.
    1. User mode – execution done on behalf of a user.
    2. Monitor mode (also kernel mode or system mode) – execution done on behalf of operating system.

OS Operations (cont.)

- Dual-Mode Operation (Cont.)
  - Mode bit added to computer hardware to indicate the current mode:
    monitor (0) or user (1).
  - When an interrupt or fault occurs hardware switches to monitor mode.



Transition from user to kernel mode

OS Operations (cont.)

- Timer
  - Timer – interrupts computer after specified period to ensure operating system maintains control (Infinite loop !).
    - Timer is decremented every clock tick.
    - When timer reaches the value 0, an interrupt occurs.
  - Timer commonly used to implement time sharing.
  - Timer also used to compute the current time.
  - Load-timer is a privileged instruction.

Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a passive entity, process is an active entity.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data (input given to processes)
- Process termination requires reclaim of any reusable resources

Process Management (cont.)

- Single-threaded process has one program counter specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads

Process Management (cont.)

- Process Management Activities
  - The operating system is responsible for the following activities in connection with process management:
    - Creating and deleting both user and system processes
    - Suspending and resuming processes
    - Providing mechanisms for process synchronization
    - Providing mechanisms for process communication
    - Providing mechanisms for deadlock handling

Memory Management

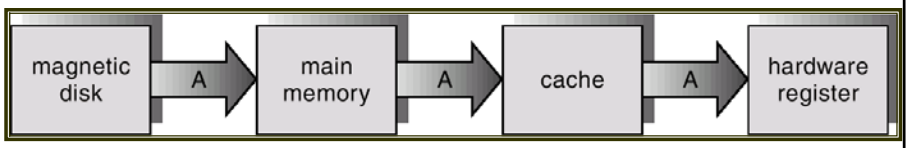
- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently 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

Storage Management

- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility
- Caching – copying temporary information into faster storage system; main memory can be viewed as a last cache for secondary storage.
- Internal programmable registers (index registers) provides a high speed cache for main memory.

Storage Management (cont.)

- Caching
  - Use of high-speed memory to hold recently-accessed data.
  - Requires a cache management policy (cache size, replacement policy).
  - Caching introduces another level in storage hierarchy. This requires data that is simultaneously stored in more than one level to be consistent.
  - Data that resides on the disk are copied into the main memory, then into the cache and then into the internal registers for modification such as a variable incrementation.
  - This modification is done on the CPU registers and should be copied back on the disk for a consistency purpose.

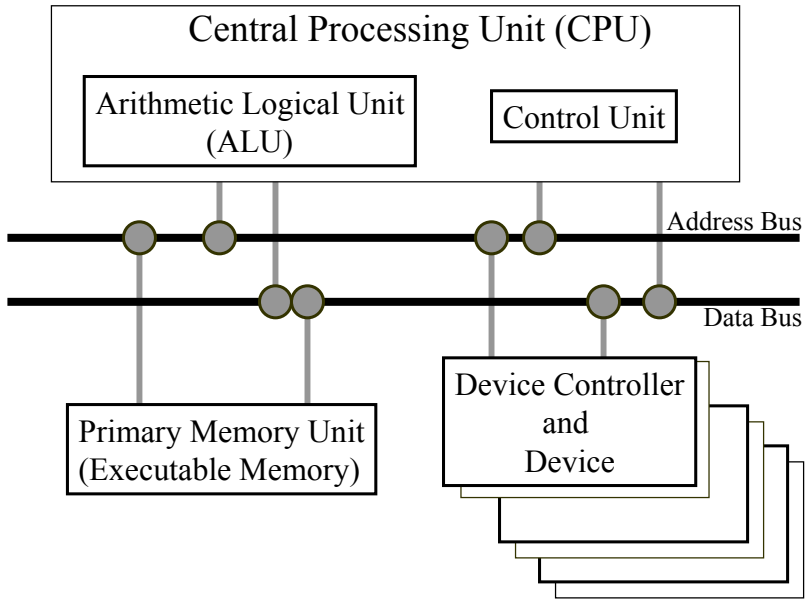


Migration of A From Disk to Register

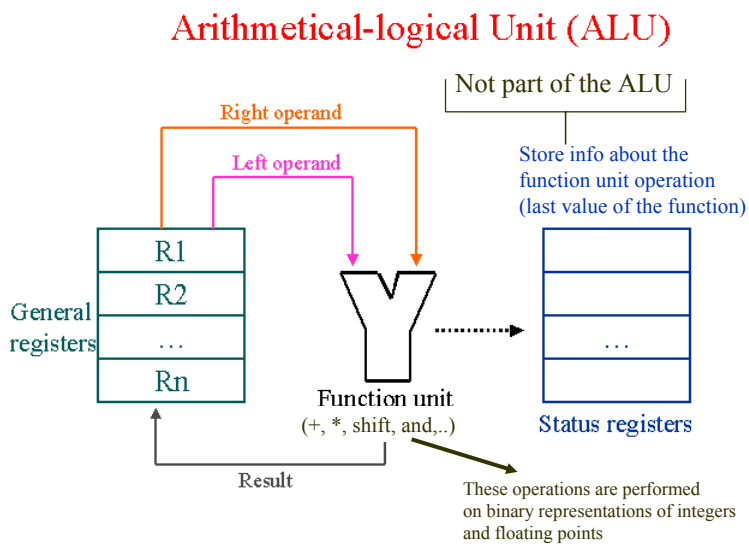
Storage Management (cont.)

- I/O Systems
  - One purpose of OS is to hide peculiarities of hardware devices from the user.
  - I/O subsystem responsible for
    - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs).
    - General device-driver interface.
    - Drivers for specific hardware devices.

Appendix: The Von Neumann Architecture



The Von Neumann Architecture (cont.)



The Von Neumann Architecture (cont.)

For example, if a C source program contains the code segment

a = b + c;
d = a – 100;

then the following assembly language instructions can be executed by the CPU to accomplish these 2 statements:

```
// Code for a = b + c;
Load     R3, b // Copy the value of b from memory to R3
Load     R4, c // Copy the value of c from memory to R4
Add      R3, R4 // Sum placed in R3
Store    R3, a // Store the sum into memory cell a
// Code for d = a – 100;
Load     R4, = 100 // Load the value 100 into R4
Subtract  R3, R4 //   Difference placed in R3
Store      R3, d //   Store the difference in memory cell d
```

The Von Neumann Architecture (cont.)

(responsible of retrieving and executing a sequence of instructions from the memory)