CSMC 412

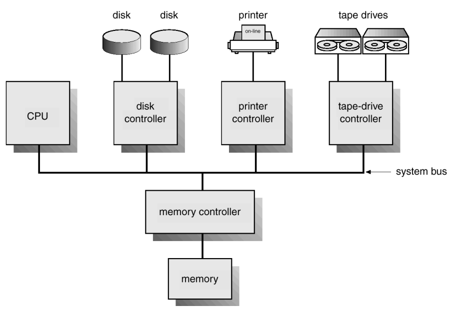
Operating Systems
Prof. Ashok K Agrawala

Set 3

Computer-System Structures

- Computer System Operation
- I/O Structure
- Storage Structure
- Storage Hierarchy
- Hardware Protection
- General System Architecture

Computer-System Architecture



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*.

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.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- A *trap* is a software-generated interrupt caused either by an error or a user request.
- An operating system is *interrupt* driven.

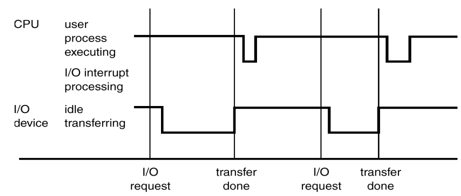
Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
  - *polling*
  - vectored interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

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

Interrupt Time Line For a Single Process Doing Output



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)
  - > 1Gigabyte/sec
- Memory Bus (on processor board)
  - ~500 megabytes 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)
  - tens of megabytes per second

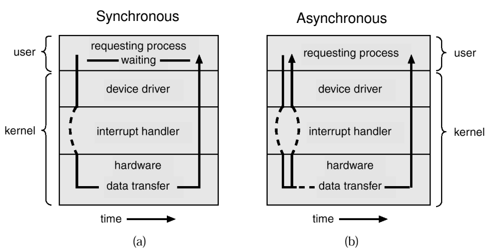
Issues In Busses

- 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

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 operating system 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.
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.

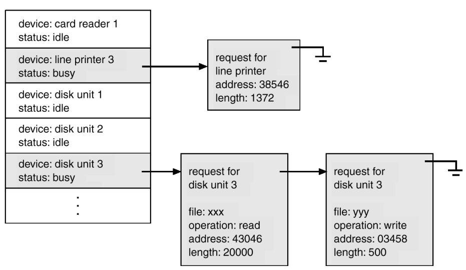
Two I/O methods



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**

Device-Status Table



Direct Memory Access (DMA) Structure

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

Storage Structure

- **Main memory – only large storage media that the CPU can access directly.**
- **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.**

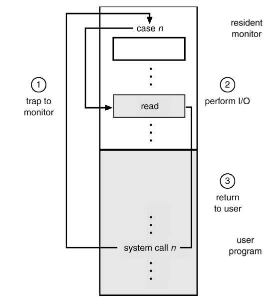
I/O Operations

- **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**

General-System Architecture

- Given the I/O instructions are privileged, how does the user program perform I/O?
- System call – the method used by a process to request action by the operating system.
  – Usually takes the form of a trap to a specific location in the interrupt vector.
  – Control passes through the interrupt vector to a service routine in the OS, and the mode bit is set to monitor mode.
  – The monitor verifies that the parameters are correct and legal, executes the request, and returns control to the instruction following the system call.

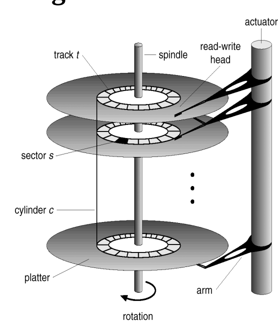
Use of A System Call to Perform I/O



Disks

- Several types:
  – Hard Disks - rigid surface with magnetic coating
  – Floppy disks - flexible surface with magnetic coating
  – Optical (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 tens of gigabytes
    » throughput: sustained < 10 megabytes/sec
    » latency: mili-seconds

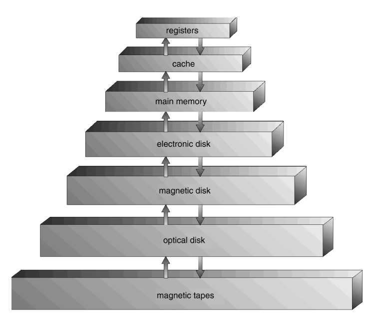
Moving-Head Disk Mechanism



Storage Hierarchy

- Storage systems organized in hierarchy.
  – Speed
  – cost
  – volatility
- *Caching* – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage.

Storage-Device Hierarchy



Hardware Protection

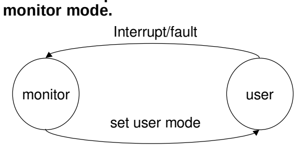
- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

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 *supervisor mode* or *system mode*) – execution done on behalf of operating system.

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.



- *Privileged instructions* can be issued only in monitor mode.

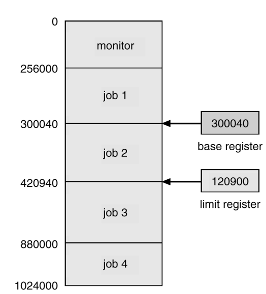
I/O Protection

- All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode (I.e., a user program that, as part of its execution, stores a new address in the interrupt vector).

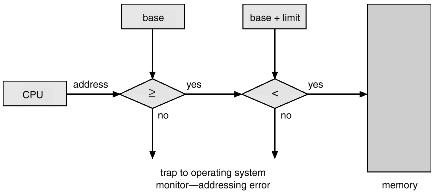
Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add tow registers that determine the range of legal addresses a program may access:
  - base register – **holds the smallest legal physical memory address.**
  - Limit register – **contains the size of the range**
- Memory outside the defined range is protected.

A Base And A limit Register Define A Logical Address Space



Protection Hardware



- **When executing in monitor mode, the operating system has unrestricted access to both monitor and user's memory.**
- **The load instructions for the *base* and *limit* registers are privileged instructions.**

CPU Protection

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