

CSMC 412

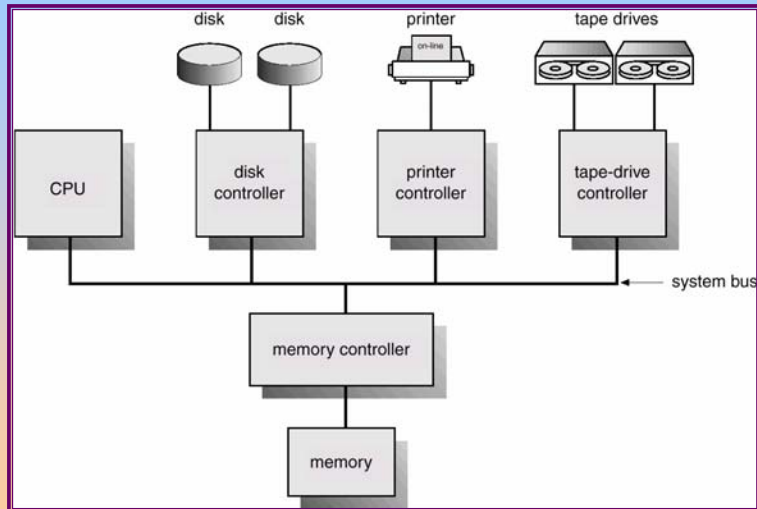
Operating Systems Prof. Ashok K Agrawala

© 2004 Ashok Agrawala
Set 2

Computer-System Structures

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

Computer-System Architecture



Operating System Concepts

2.3

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*.

Operating System Concepts

2.4

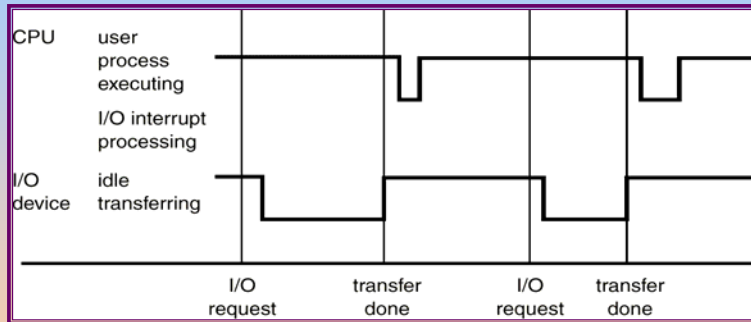
Common Functions of Interrupts

- 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

Interrupt Time Line For a Single Process Doing Output



Operating System Concepts

2.7

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

Operating System Concepts

2.8

Different Requirements lead to Multiple Buses

- Processor Bus (on chip)
 - ☞ Many Gigabytes/sec
- Memory Bus (on processor board)
 - ☞ ~1-2 Gigabyte 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, USB)
 - ☞ tens of megabytes per second

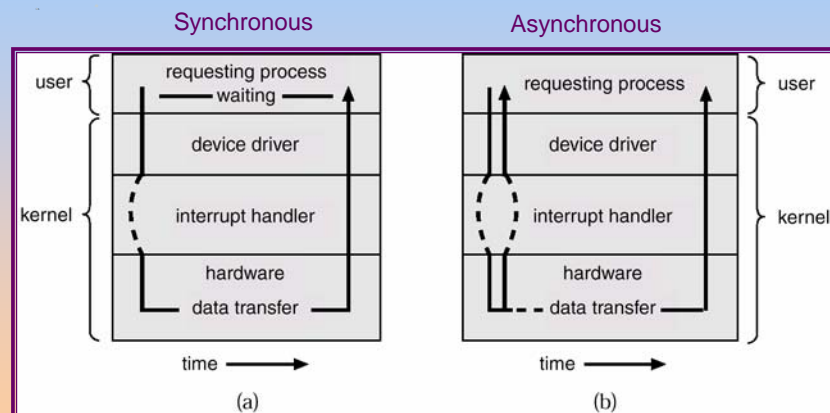
Issues In Busses

- 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

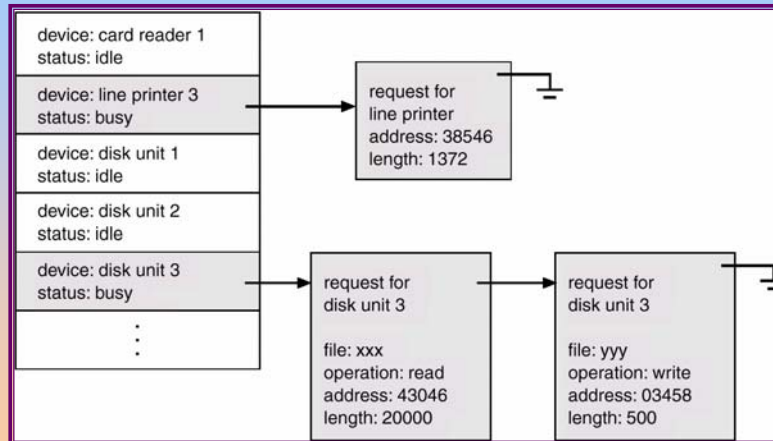
I/O Structure

- 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



Device-Status Table



Operating System Concepts

2.13

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.

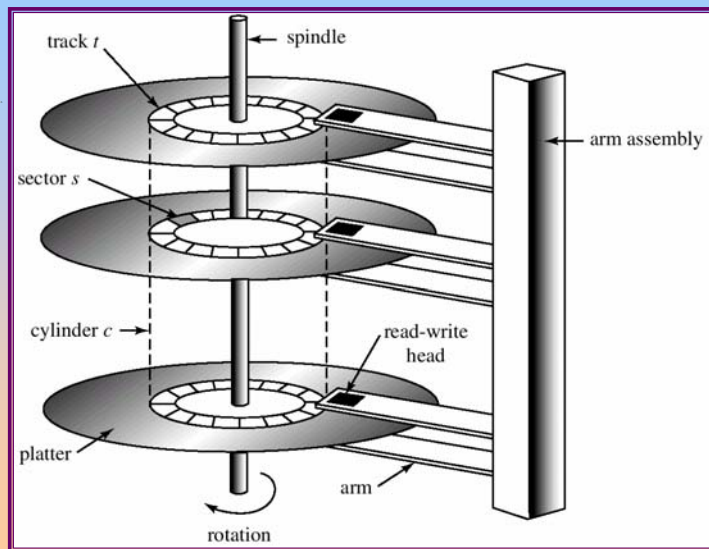
Operating System Concepts

2.14

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.

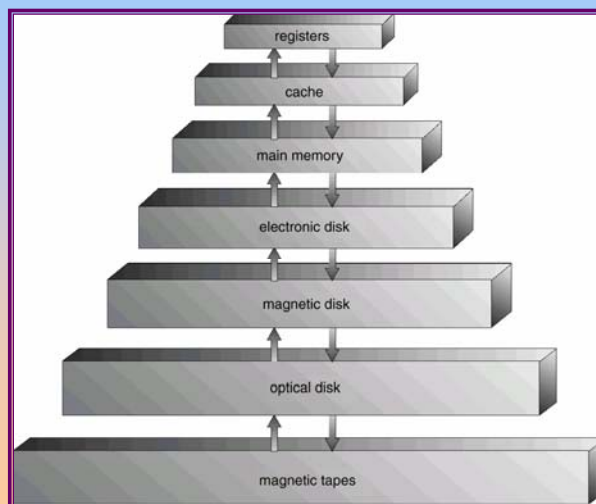
Moving-Head Disk Mechanism



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.

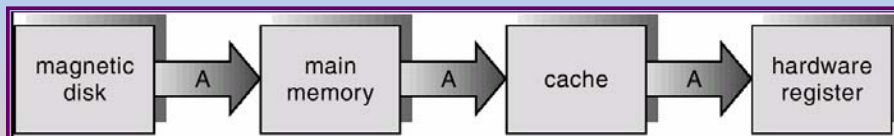
Storage-Device Hierarchy



Caching

- Use of high-speed memory to hold recently-accessed data.
- Requires a *cache management* policy.
- Caching introduces another level in storage hierarchy. This requires data that is simultaneously stored in more than one level to be *consistent*.

Migration of A From Disk to Register



Hardware Protection

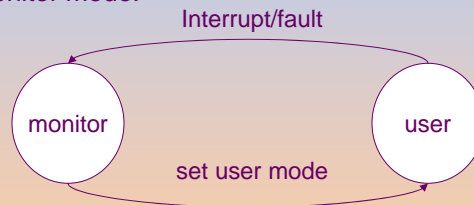
- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

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 *kernel 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).

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

I/O Interrupts

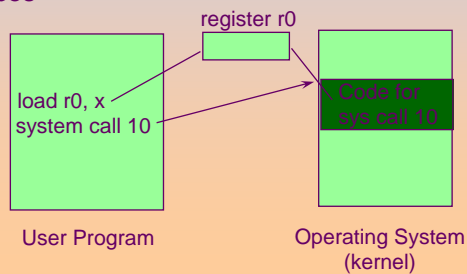
- Interrupt defined
 - ↪ indication of an event
 - ↪ can be caused by hardware devices
 - ▢ indicates data present or hardware free
 - ↪ can be caused by software
 - ▢ system call (or trap)
 - ↪ CPU stops what it is doing and executes a handler function
 - ▢ saves state about what was happening
 - ▢ 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

Hardware Protection

- Need to protect programs from each other
- Processor has modes
 - user mode and supervisor (monitor, privileged)
 - operations permitted in user mode are a subset of supervisor mode
- Memory Protection
 - control access to memory
 - only part of the memory is available
 - ▢ can be done with base/bound registers
- I/O Protection
 - I/O devices can only be accessed in supervisor mode
- Processor Protection
 - Periodic timer returns processor to supervisor mode

System Calls

- Provide the interface between application programs and the kernel
- Are like procedure calls
 - take parameters
 - calling routine waits for response
- Permit application programs to access protected resources



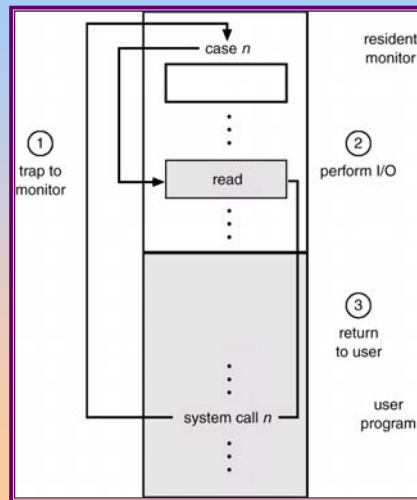
System Call Mechanism

- Use numbers to indicate what call is made
- Parameters are passed in registers or on the stack
- Why do we use indirection of system call numbers rather than directly calling a kernel subroutine?
 - provides protection since the only routines available are those that are export
 - permits changing the size and location of system call implementations without having to re-link application programs

Types of System Calls

- File Related
 - open, create
 - read, write
 - close, delete
 - get or set file attributes
- Information
 - get time
 - set system data (OS parameters)
 - get process information (id, time used)
- Communication
 - establish a connection
 - send, receive messages
 - terminate a connection
- Process control
 - create/terminate a process (including self)

Use of A System Call to Perform I/O



Operating System Concepts

2.31

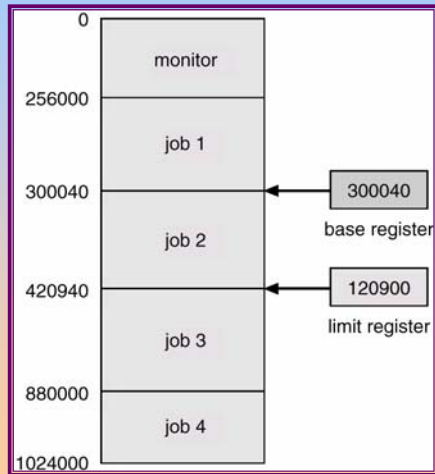
Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two 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.

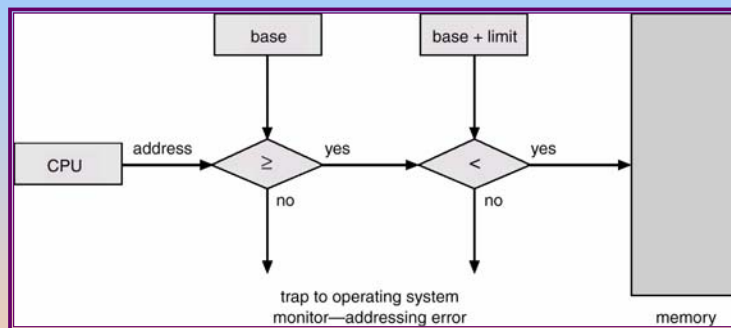
Operating System Concepts

2.32

Use of A Base and Limit Register



Hardware Address Protection



Hardware Protection

- 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.

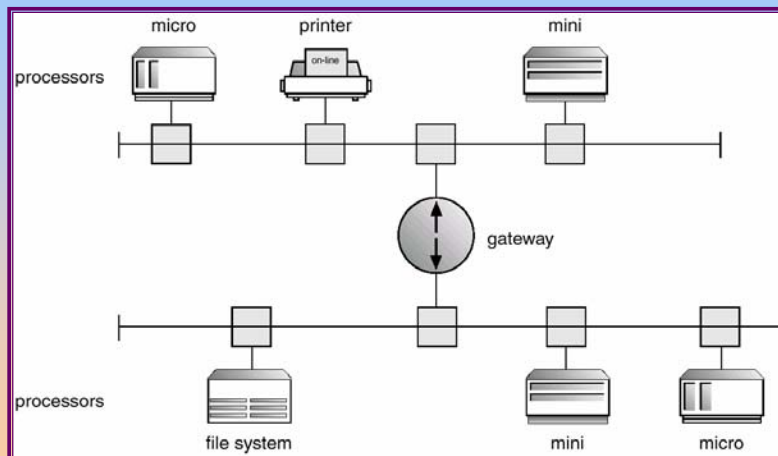
CPU Protection

- *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.

Network Structure

- Local Area Networks (LAN)
- Wide Area Networks (WAN)

Local Area Network Structure



Wide Area Network Structure

