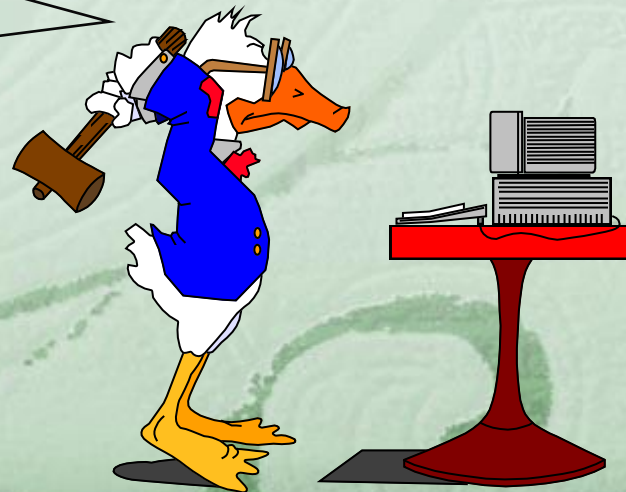


# Computer System Overview

Let's figure out  
what's inside  
this thing...



# Multiple Interrupts

## ■ 2 Choices

### ☞ Disable Interrupts

- Disable upon entering an interrupt handler
- Enable upon exiting

### ☞ Allow Interrupts

- Allow an interrupt handler to be interrupted
- Priorities?

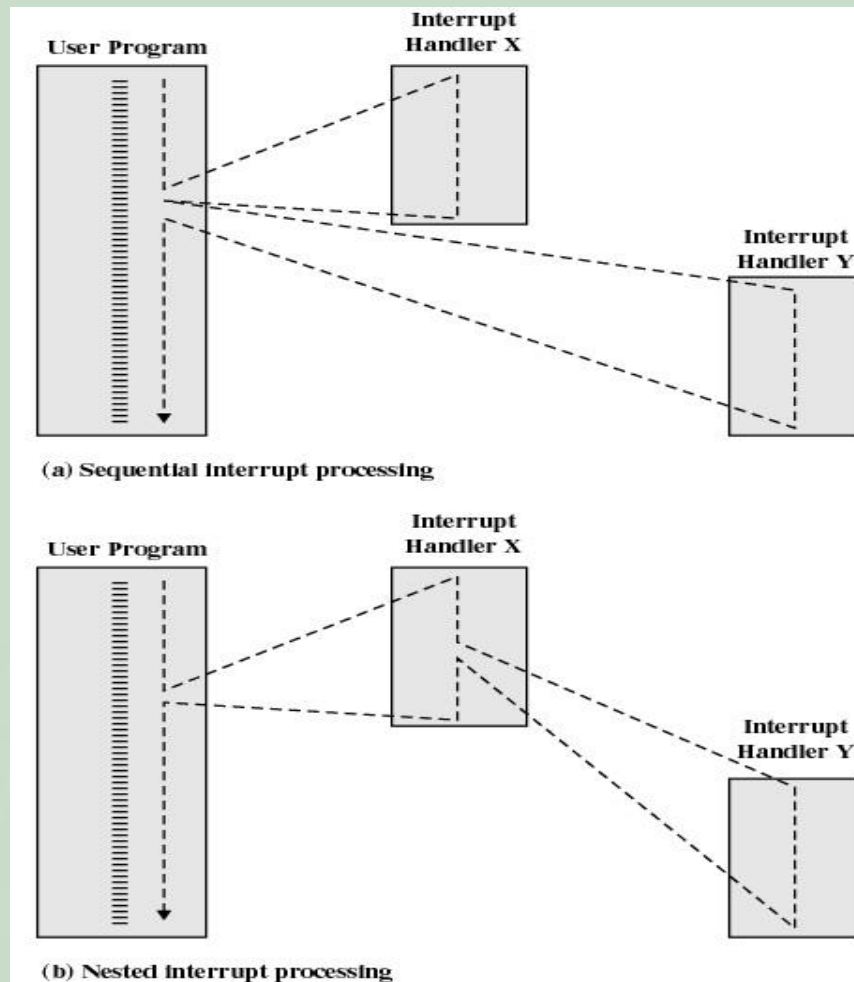


Figure 1.12 Transfer of Control with Multiple Interrupts

# Examples of interrupts

- Mouse moved/clicked.
- Disk drive operation completed.
- Keyboard key pressed.
- Printer out of paper.
- Video card wants memory access.
- Modem sending or receiving.
- USB scanner has data.



# Interrupt handlers

- Interrupt handlers are the routines that are called when an interrupt is detected.
- Interrupt handlers are usually short sections of code that are designed to handle the immediate needs of the device and return control to the operating system or user program.



# Summary

- An **interrupt** is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.
- An interrupt can occur at any time.
- Hardware and software are needed to support interrupt handling.
- The current state of the interrupted program must be saved.



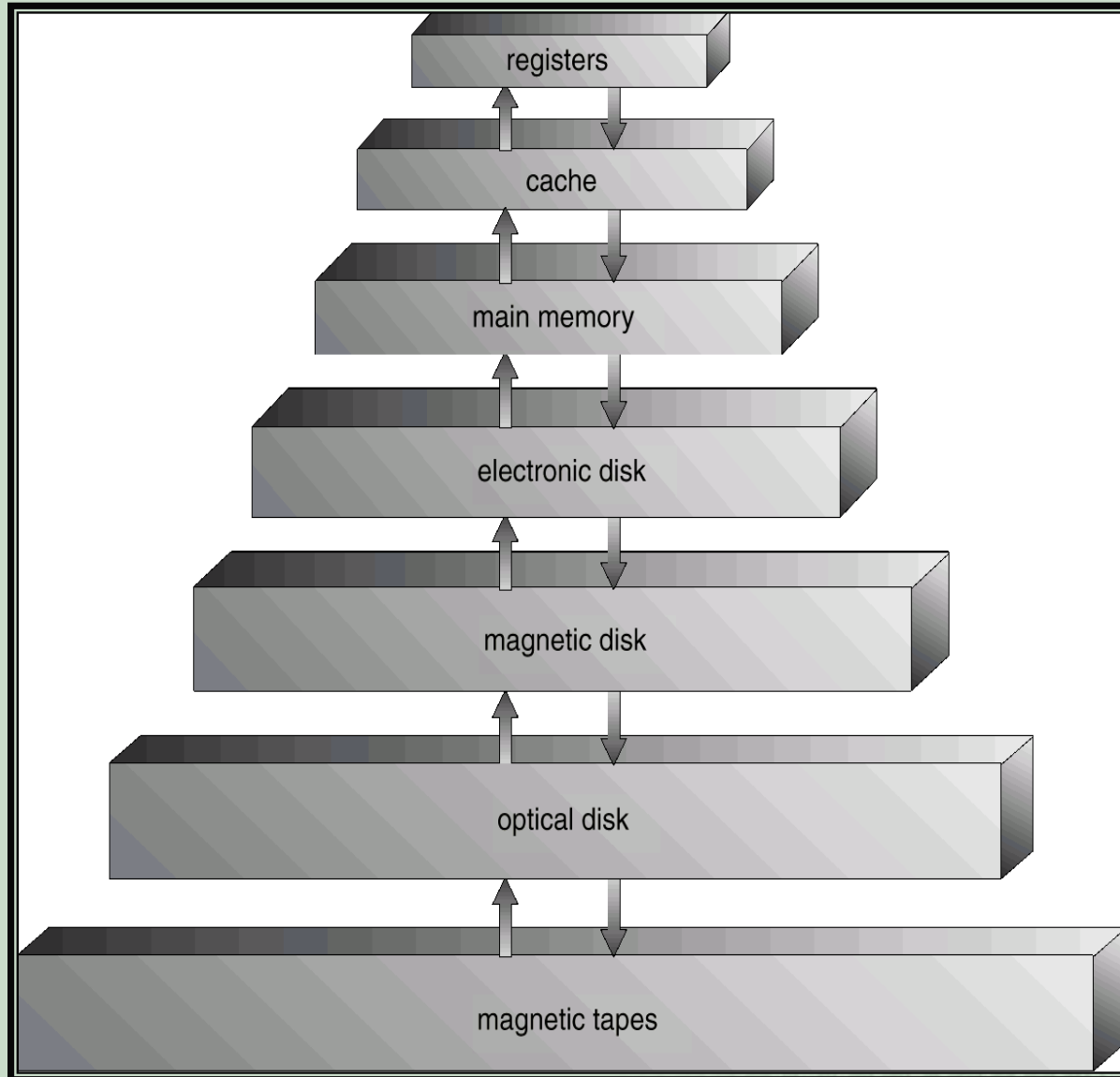
# Computer Systems

- Registers
- Interrupts
- **Caching**
- Input/Output
- Protection
- Summary





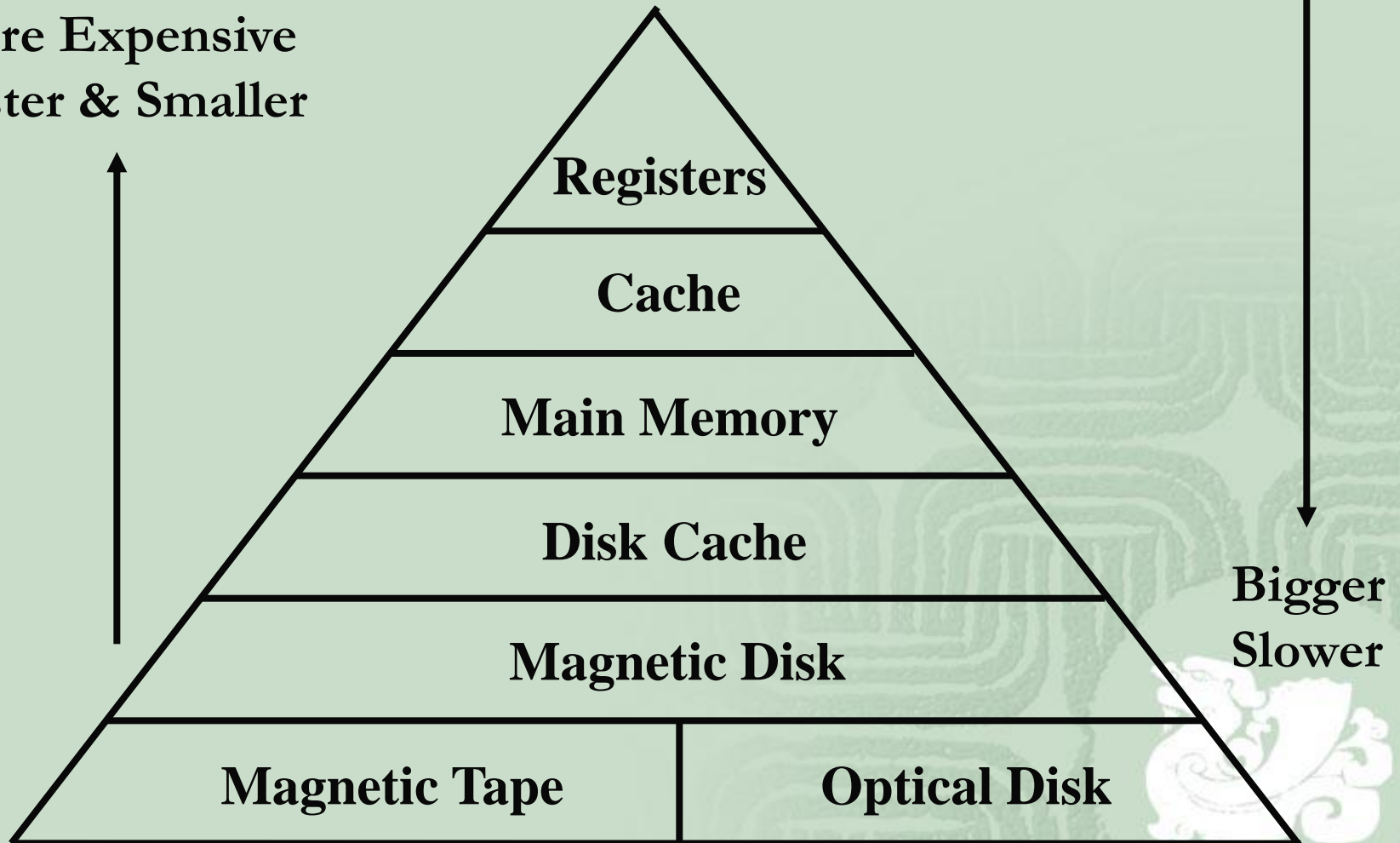
# Storage-Device Hierarchy



# Memory Hierarchy

Caching

More Expensive  
Faster & Smaller





# Comparative access timings

- Processor registers: 5 nanoseconds
- SRAM memory: 15 nanoseconds
- DRAM memory: 60 nanoseconds
- Magnetic disk: 10 milliseconds
- Optical disk: (even slower)

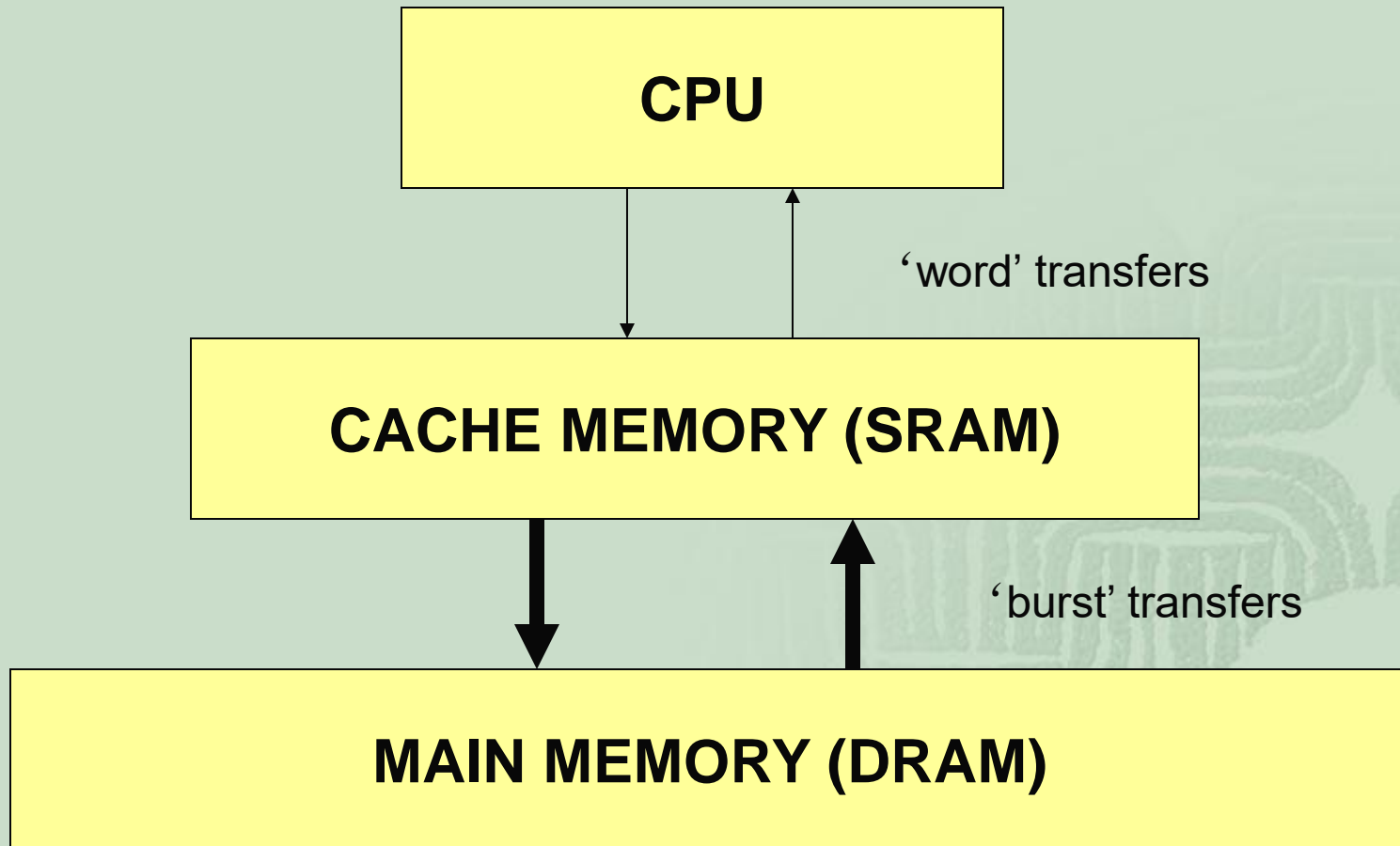


# System design decisions

- How much memory?
- How fast?
- How expensive?
- Tradeoffs and compromises
- Modern systems employ a 'hybrid' design



# Memory hierarchy



# Memory Cache

- Given:
  - ∞ Processor speed is faster than memory speed
  - ∞ Execution/data localizes
- Cache:
  - ∞ Contains a portion of main memory
  - ∞ Invisible to operating system
  - ∞ Increases the speed of memory
- Processor first checks cache
- If not found in cache, the block of memory containing the needed information is moved to the cache



# Cache Design

- Cache size

- ☞ small caches have a significant impact on performance

- Block size

- ☞ the unit of data exchanged between cache and main memory

- ☞ hit means the information was found in the cache



# Cache Design

- Mapping function
  - ∞ Determines which cache location the block will occupy
- Replacement algorithm
  - ∞ Determines which block to replace
  - ∞ Least-Recently-Used (LRU) algorithm
- Write policy
  - ∞ write a (dirty) block of cache back to main memory



# Another caching technique: Disk Cache

- A portion of main memory used as a buffer to temporarily hold data for the disk
- Disk writes are clustered
- Some data written out may be referenced again. The data are retrieved rapidly from the software cache instead of slowly from disk





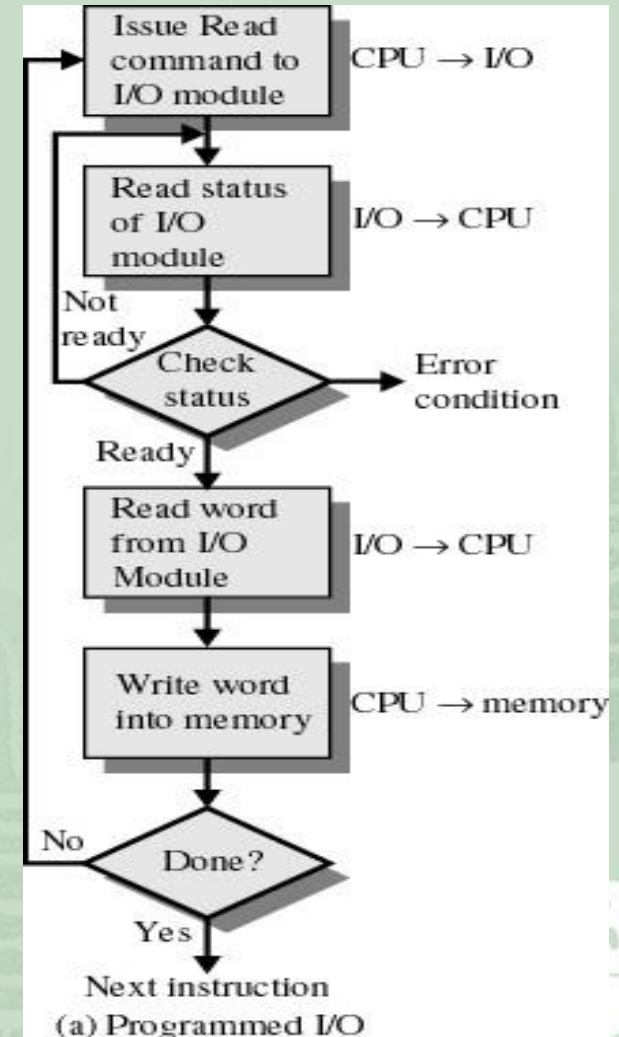
# Computer Systems

- Registers
- Interrupts
- Caching
- **Input/Output**
- Protection
- Summary



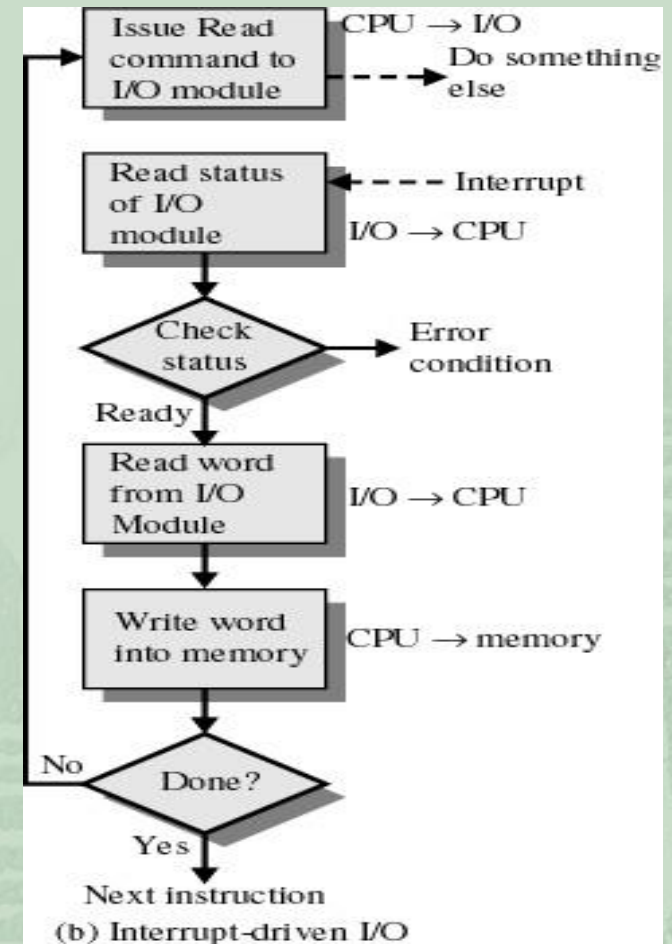
# Programmed I/O

- I/O module performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur
- Processor is kept busy checking status (busy waiting)



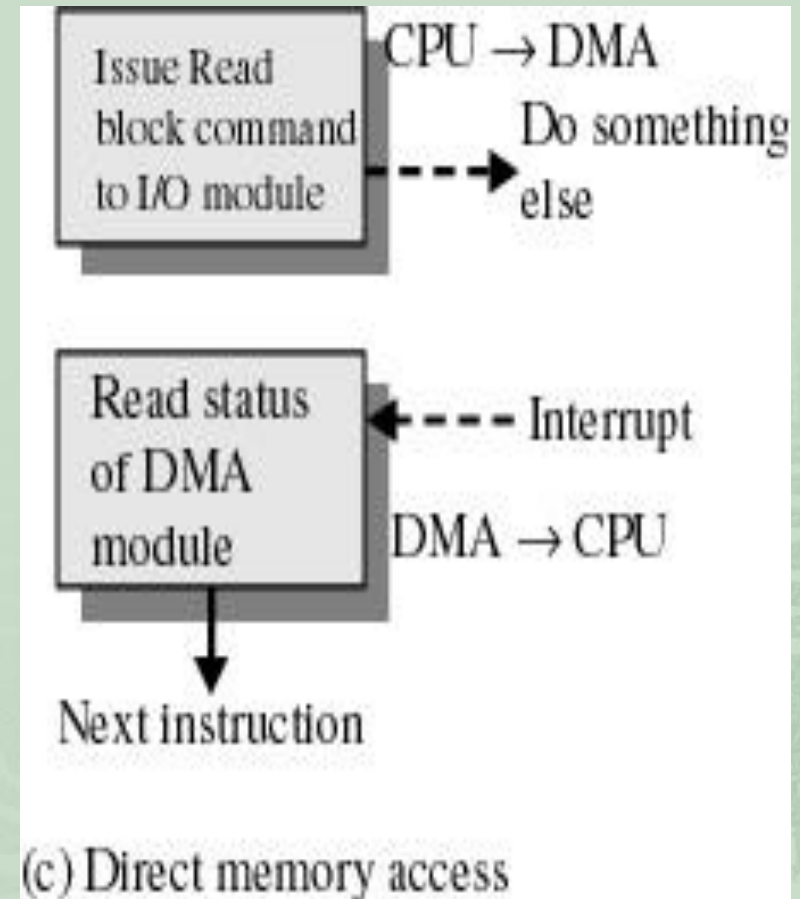
# Interrupt-Driven I/O

- Processor is interrupted when I/O module ready to exchange data
- Processor is free to do other work
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor



# Direct Memory Access

- Transfers a block of data directly to or from memory
- An interrupt is sent when the task is complete
- The processor is only involved at the beginning and end of the transfer

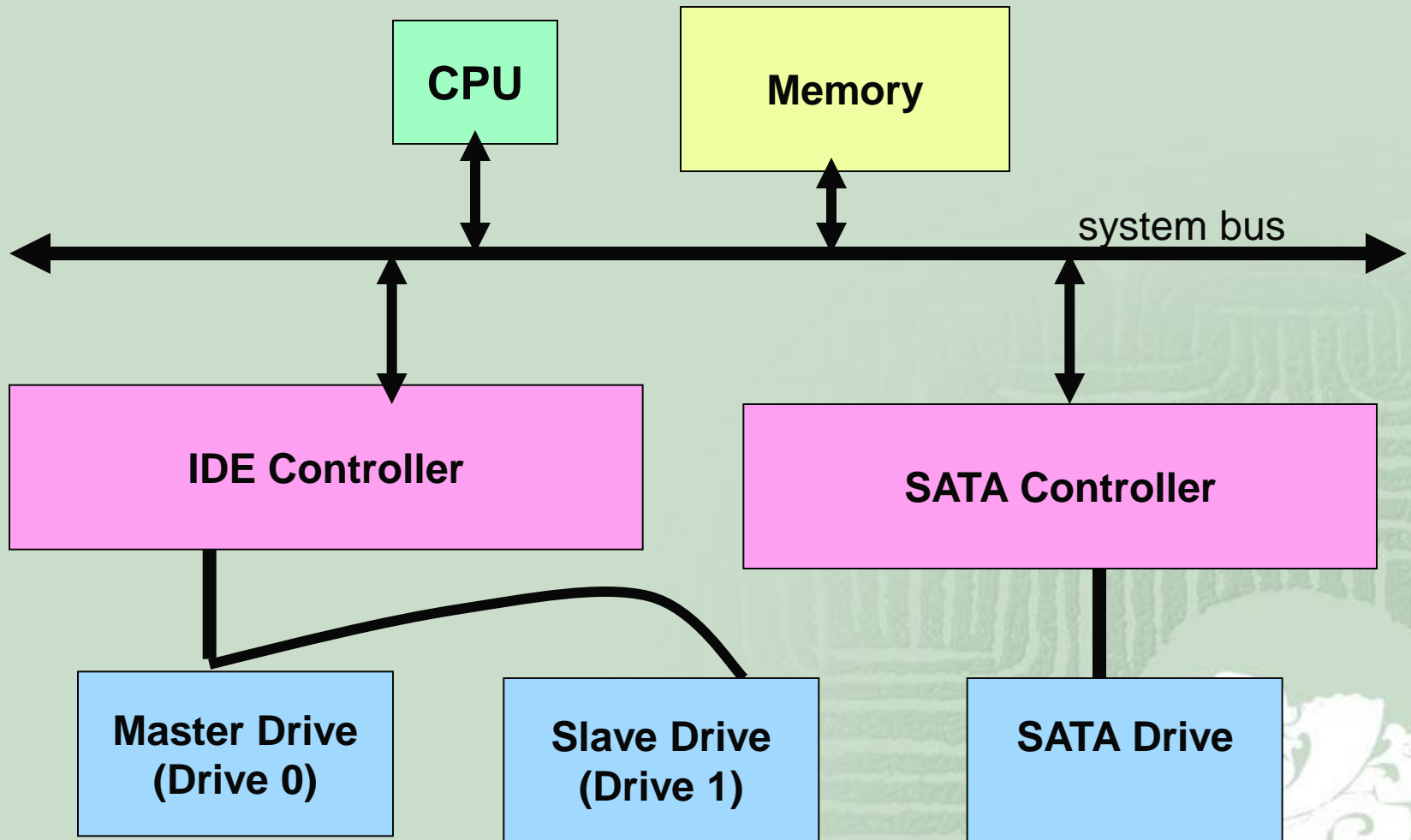


# Example: Hard Disks

- Most PCs use either **SATA** or **IDE** disks for storage of files and system data.
- **SATA** stands for **Serial Advanced Technology Attachment** (or **Serial ATA**)
- **IDE** means **Intelligent Drive Electronics**, also called **Parallel ATA** or **PATA**.
- By the beginning of 2007, **SATA** had largely replaced **IDE** in all new systems.
  - Inexpensive, large storage capacity, hot plugging, faster



# The HD Controller



# ‘IDENTIFY DRIVE’

- There exist about 40 different commands (e.e., read, write, seek, format, sleep, etc)
- Some are ‘mandatory’, others ‘optional’
- An example: the ‘Identify Drive’ command
  1. It provides information on disk’s geometry and some other operational characteristics
  2. It identifies the disk’s manufacturer and it provides a unique disk serial-number





# Disk Command Protocol

- Disk Commands typically have 3 phases:
  - ⌘ COMMAND PHASE: CPU issues a command
  - ⌘ DATA PHASE: data moves to/from buffer
  - ⌘ RESULT PHASE: CPU reads status/errors



# Computer Systems

- Registers
- Interrupts
- Caching
- Input/Output
- **Protection**



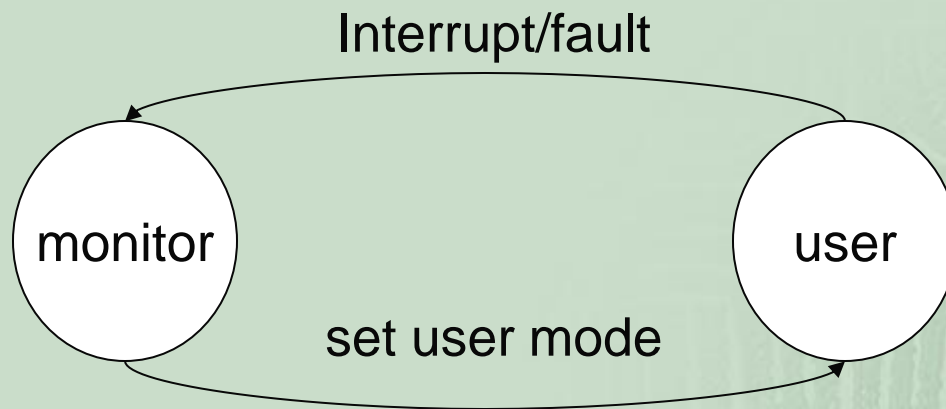
# Hardware Protection

- Why protect hardware?
- From what?
  - ∞ Shared hardware resources – memory, disk, ...
  - ∞ Errant programs
- How?
  - ∞ CPU provides 2 modes of operation
    - User Mode (non-privileged)
    - Supervisor/monitor/OS mode (privileged)
  - ∞ Privileged instructions can only be executed in monitor mode
    - All I/O instructions are privileged

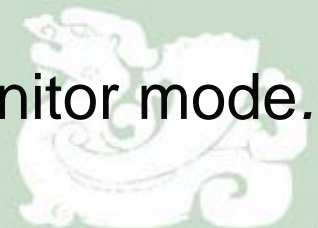


## Dual-Mode Operation (Cont.)

- *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.

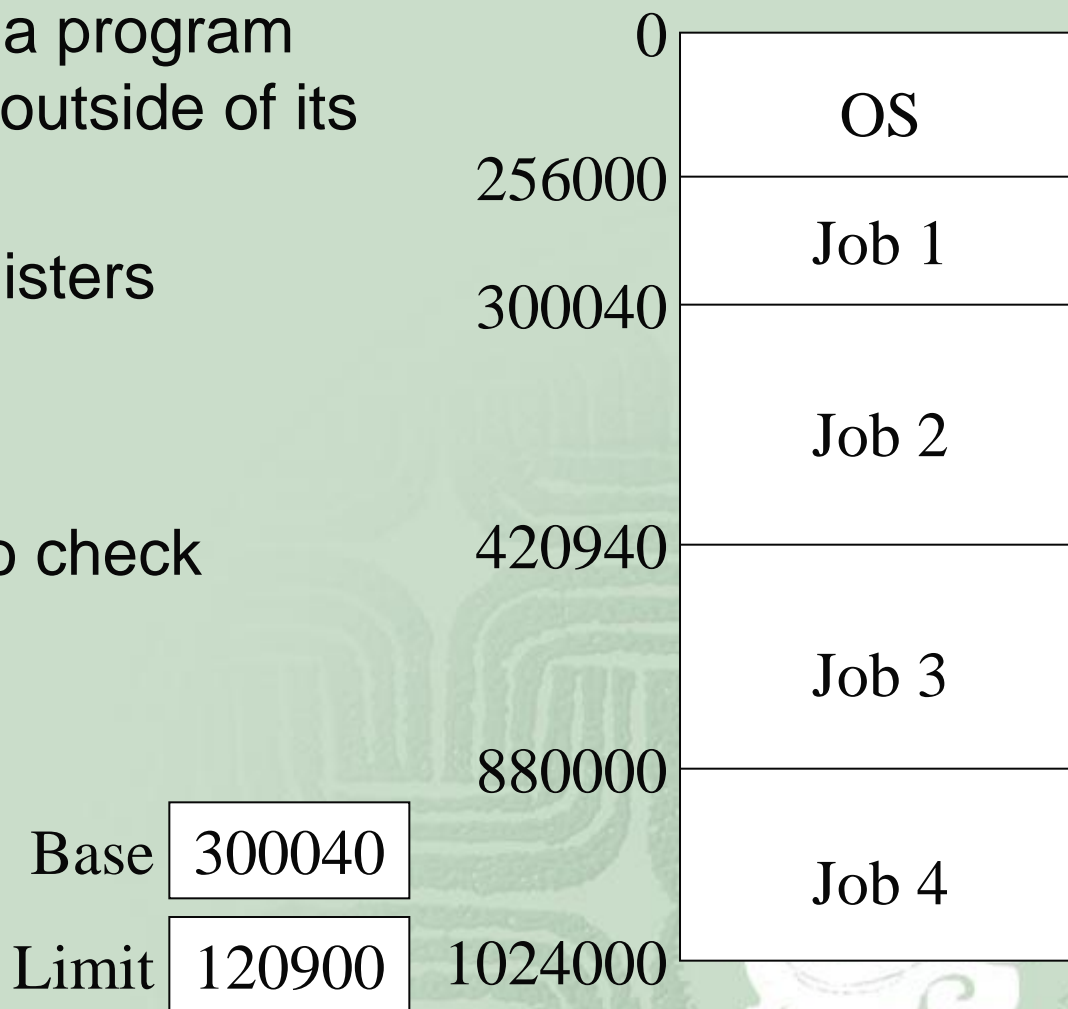


*Privileged instructions* can be issued only in monitor mode.

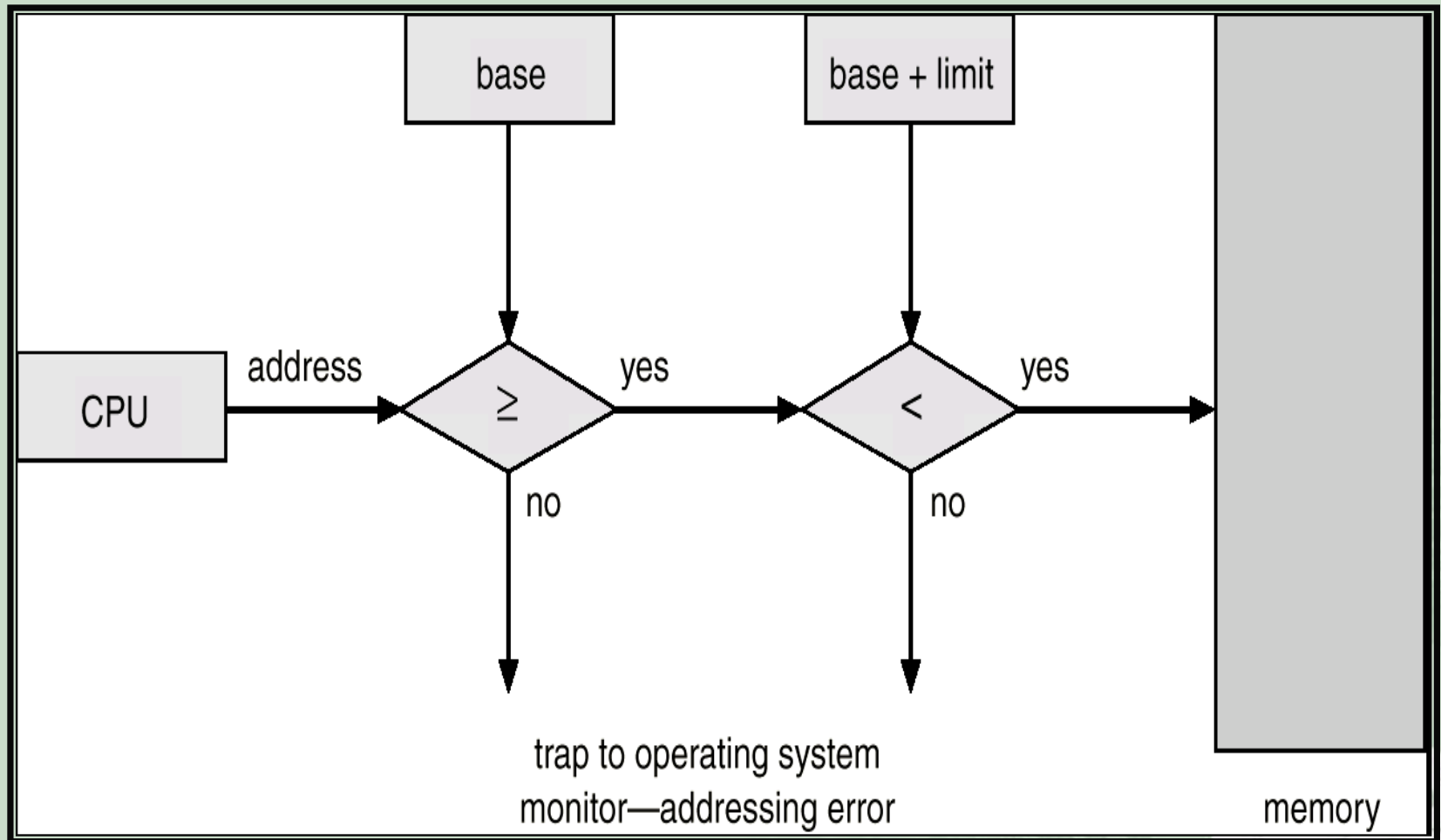


# Memory Protection

- We must not allow a program access to memory outside of its space.
- How? Add two registers
  - ↻ Base
  - ↻ Limit
- Use the registers to check every reference



# Hardware Address Protection



# CPU Protection

- What is there to protect?
  - ❧ Configuration
  - ❧ CPU as a resource
- Timer
  - ❧ Task Switching uses the timer
    - When process is loaded, timer is set
    - Timer is decremented each cycle
    - When time is zero, an interrupt is generated
    - Process is switched
  - ❧ Load-timer is a privileged instruction





# 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.



# Protect at all cost?

- If all I/O instructions are protected, how do you perform input or output?
- System calls
  - ⌘ Often the call is a trap (soft interrupt) to a ISR (Interrupt Service Routine)
  - ⌘ Control is passed to the ISR which sets the mode bit to **monitor mode**
  - ⌘ ISR verifies that parameters are correct
  - ⌘ ISR executes request, resets mode
  - ⌘ Control is returned to user program

