Information Technology Institute

Computer Operating System Concepts

Chapter Two

COMPUTER SYSTEM STRUCTURE

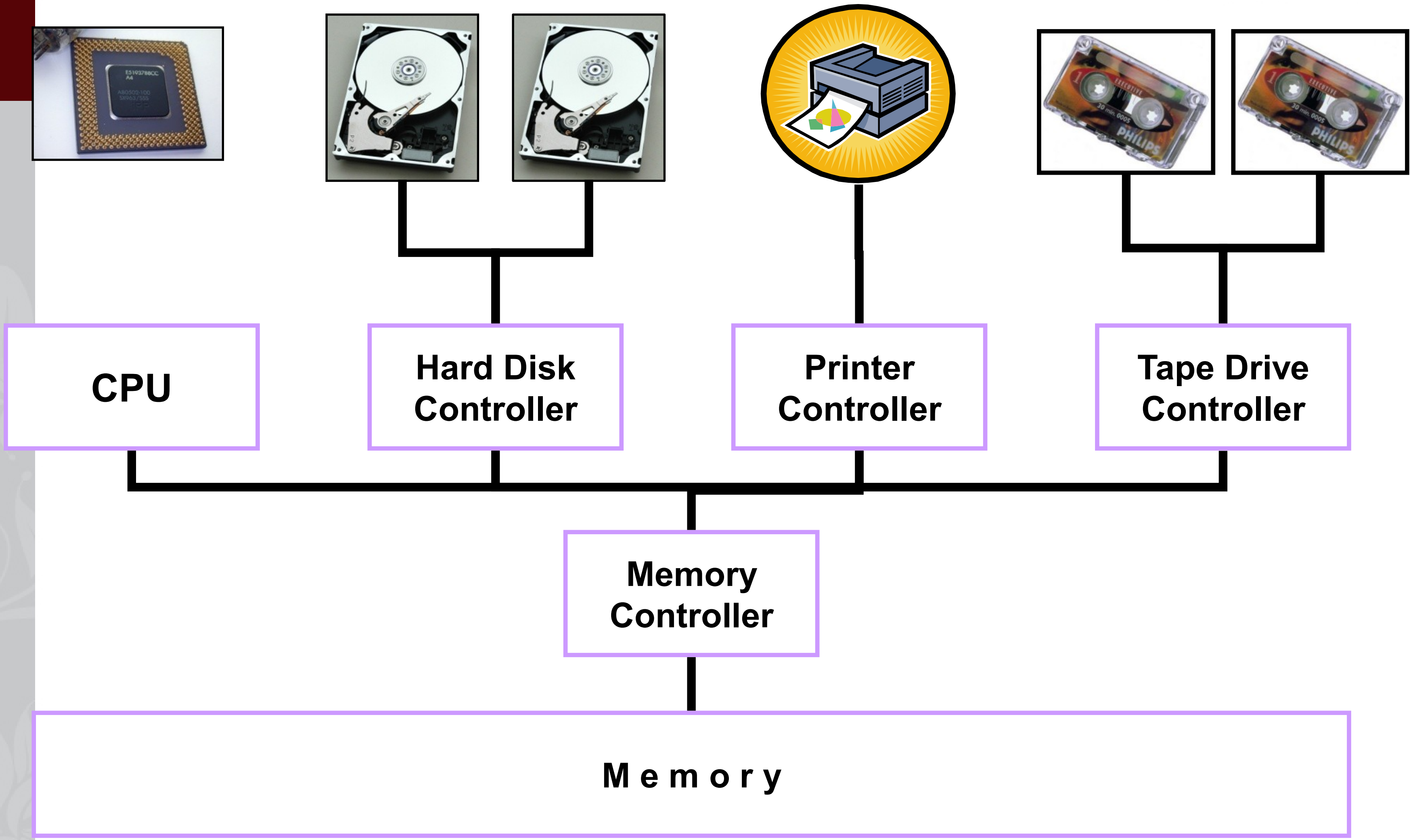
Table of Content

COMPUTER SYSTEM OPERATION

Computer System Operation

- Computer system consists of a CPU and a number of device controllers that are connected through a common bus that provides access to shared memory
- CPU and device controllers execute concurrently.
- Memory controller synchronizes access to memory.

Computer System Operation Cont'd



Computer System Operation Cont'd

- 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
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.

Interrupts

- A signal sent to the CPU
- Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines.
- Interrupts:
  - Hardware Interrupts
  - Software Interrupts: system calls

Interrupts Cont'd

- A trap is a software generated interrupt caused by:
  - Error: division by zero or invalid memory access
  - Request: from a user program to O/S
- An operating system is interrupt driven.

The 6502's interrupt latency is one of the 8-bit world's fastest.

Interrupt Handling

1. CPU is interrupted
2. CPU stops current process
3. CPU transfers execution to a fixed location
4. CPU executes interrupt service routine
5. CPU resumes process

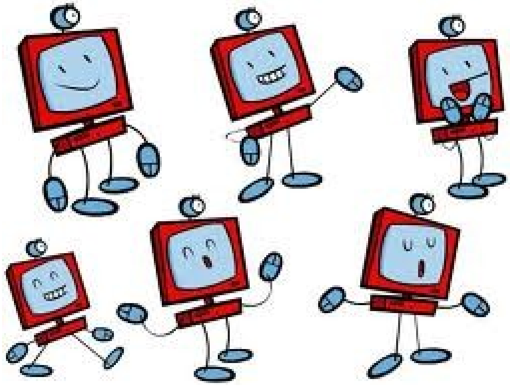
- Notes:
  - Interrupts must be handled quickly
  - Interrupted process information must be stored

Interrupt Handling Mechanisms

- Generic
  - Examine interrupt information and calls Specific routine
- Interrupt vector
  - Table of pointers to interrupt routines for various devices
  - Applied in Unix & DOS

Computer System Startup

1. Power up
2. Initial program: bootstrap
   - Stored in ROM
   - Initialize:
     1. CPU registers
     2. Device controllers
     3. Memory contents
     4. Load the operating system (kernel)
3. Kernel starts the first process, init
4. Init waits for an event (interrupt) to occur



I/O STRUCTURE

I/O Structure

- Controllers:
  - Is in charge of a specific type of device
  - Moves data between device and local buffer
  - A controller may have more than one device
  - Buffer size varies

Two I/O Methods

- Synchronous I/O:
  - Process request I/O operation
  - I/O operation is started
  - I/O Operation is complete
  - Control is returned to the user process
- Asynchronous I/O:
  - Process Request I/O operation
  - I/O operation is started
  - Control is returned immediately to the user process
  - I/O continues while system operations occur

Device Status Table

- The operating system uses a table containing an entry for each I/O device
- Each table entry indicates the device's type, address, and state (not functioning, idle, or busy)
- If the device is busy with a request, the type of request and other parameters will be stored in the table entry for that process
- The operating system will also maintain a wait queue-a list of waiting requests-for each I/O device.

Direct Memory Access Structure

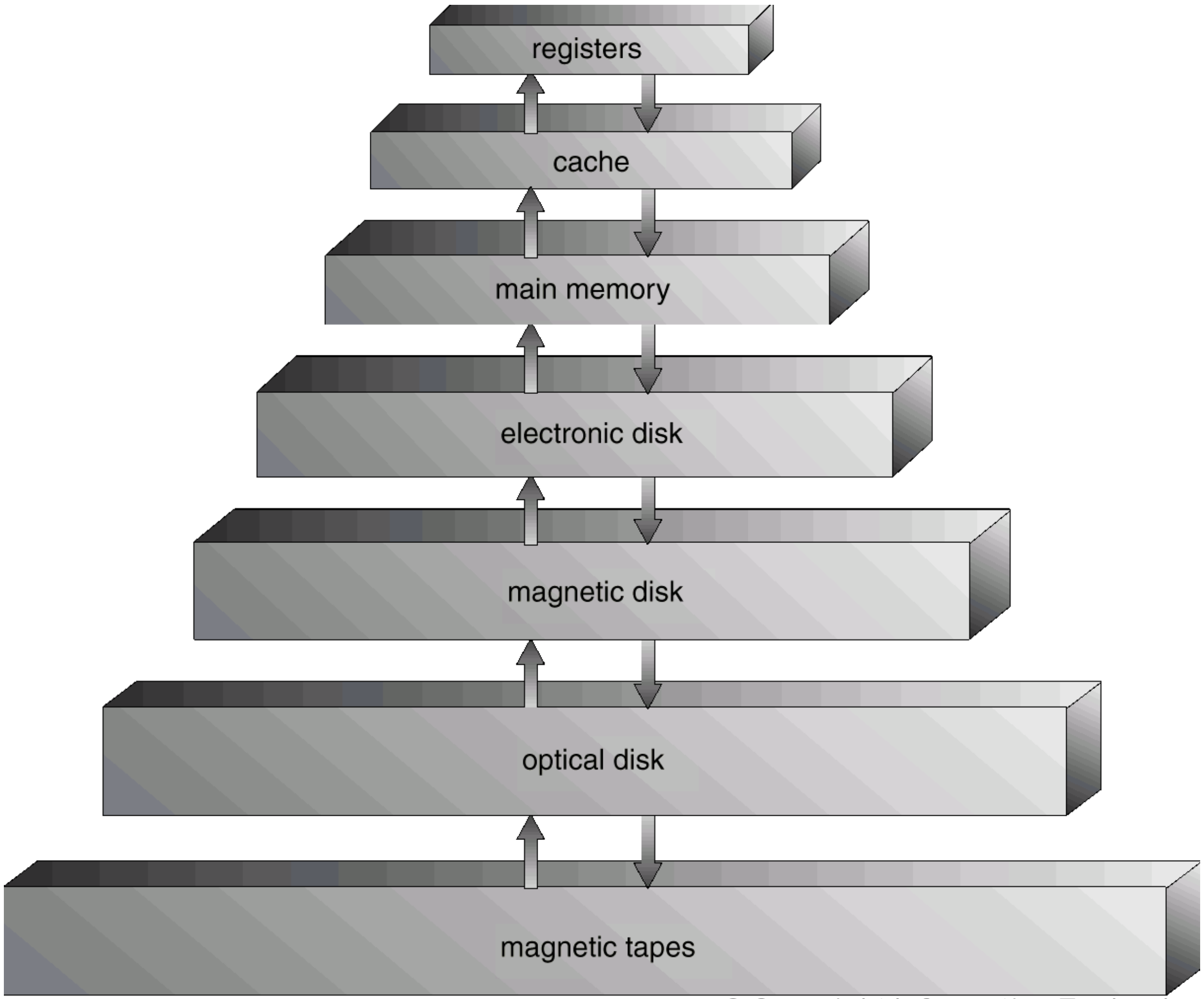
- Used for high-speed I/O devices able to transmit information at close to memory speeds.
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
- Only one interrupt is generated per block, rather than the one interrupt per byte.

STORAGE STRUCTURE

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.

Storage-Device Hierarchy



Difference of Storage Devices

- Speed
- Cost
- Capacity
- Volatility
- Reliability
- Portability

RAM

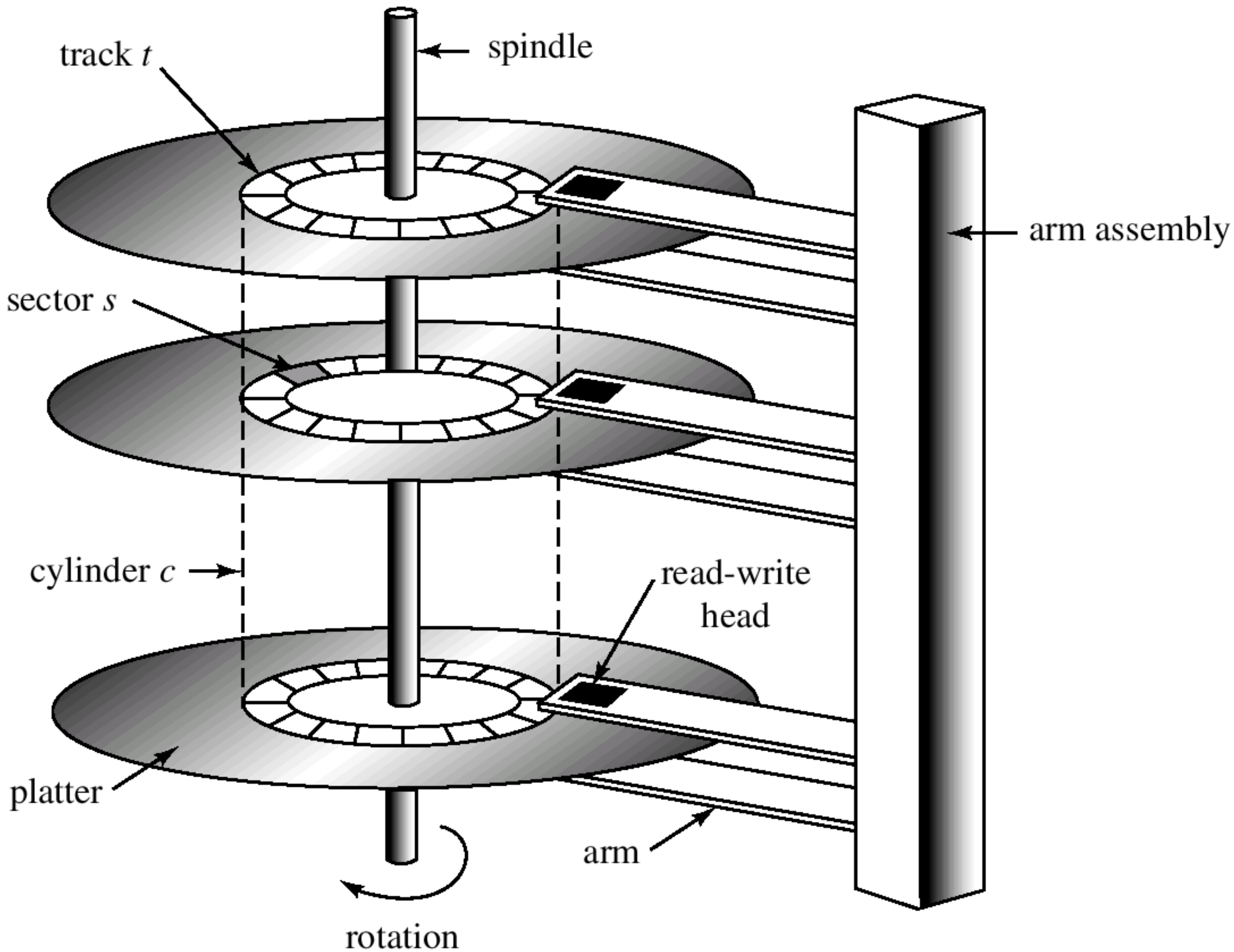
- Array of memory words
- Each byte has an address
- Memory Address:
  - Physical
  - Logical

- CPU Instructions:
  - Load: moves a word from main memory to CPU register
  - Store: move a word from CPU register to main memory

Magnetic Disk

- 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.

Magnetic Disk Cont'd



Magnetic Tapes

- Early secondary storage
- Slow access time
- Usage
  - Backup
  - Storage of infrequently used information

HARDWARE PROTECTION

Hardware Protection

- Early OS:
  - Single user
  - Programmer had full control of hardware
  - Programmer was responsible of I/O
- Error in a program
  - Single task
    - Only one program affected
  - Multi task
    - Could cause problems to other programs
- Desktop OS allow a program to access any part of memory or affect other programs instructions or data

Error Handling

- Errors
  - Illegal instruction
  - Infinite loop
  - Access of other memory addresses

- Handling Errors
  1. Errors are detected by hardware
  2. Hardware trap the error to the O/S
  3. Errors are handled by the O/S
  4. O/S terminates process
  5. O/S dumps the process to disk (if needed)

Ensuring OS Proper Operation

1. I/O Protection: illegal instructions

2. Memory protection: illegal memory access
   - Base & limit registers

3. CPU Protection: infinite loops
   - Timers

Hardware Dual-Mode

- Monitor Mode
  - Execution done on behalf of operating system.
- User Mode
  - Execution done on behalf of a user.

- Mode bit added to computer hardware to indicate the current mode
  - 0 → Monitor mode
  - 1 → User mode

Dual Mode Operation

- At boot time: Monitor mode
- OS start user processes: User mode
- Interrupt / Trap: Monitor mode

- OS always in monitor mode
- User program always in user mode

Privileged Instructions

- Machine instructions that may cause harm
- It can be executed only in monitor mode
- It can not be executed in user mode: trapped
- A user process request the OS to execute a privileged instruction: System call

1. I/O Protection

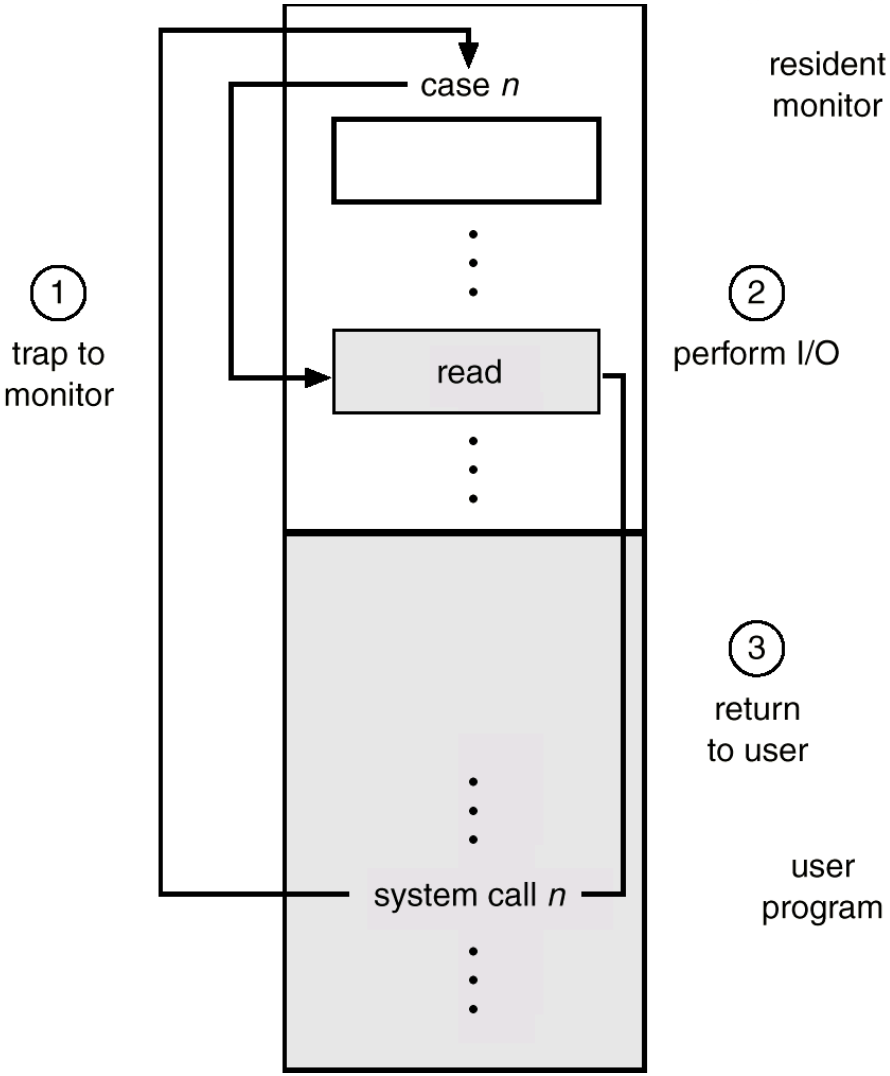
- All I/O instructions are privileged
- I/O device registers are inaccessible by user
- Users can not issue I/O instructions directly
- Users must do it through the OS

*Note: Users can never have control of the computer in monitor mode

System Calls Execution

User System call

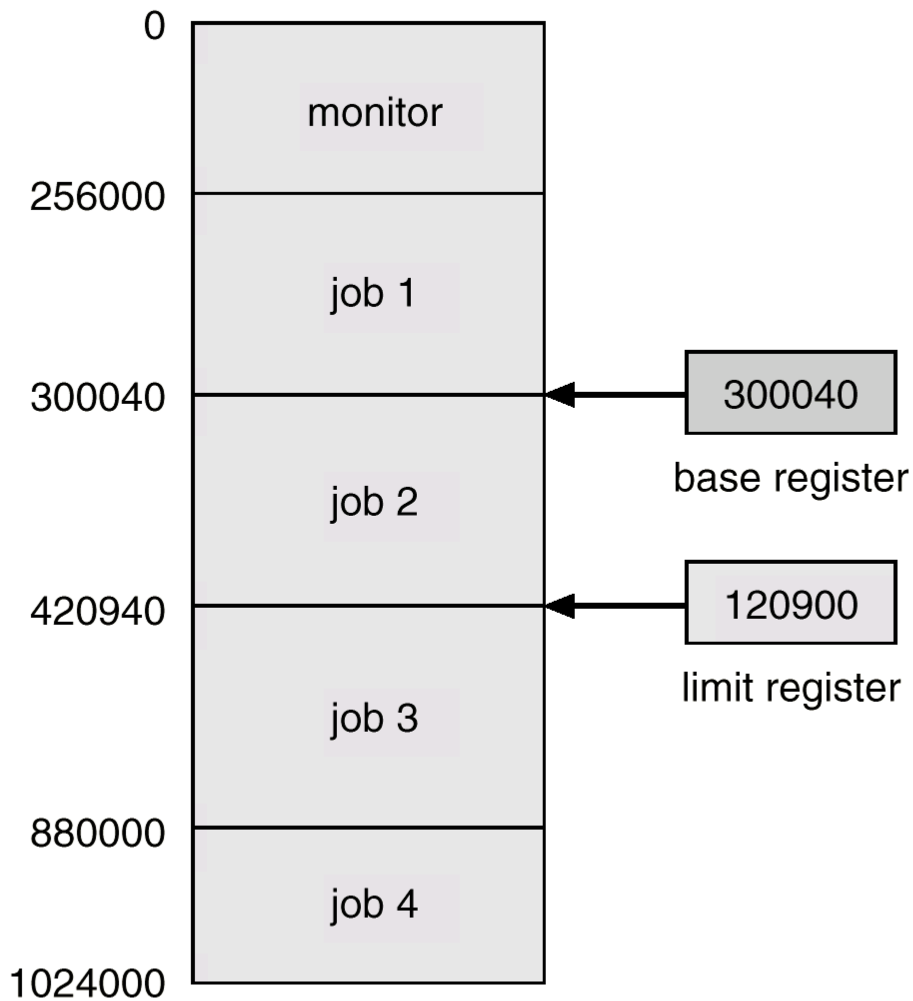
1. Trap to monitor
2. Perform I/O
3. Return to user



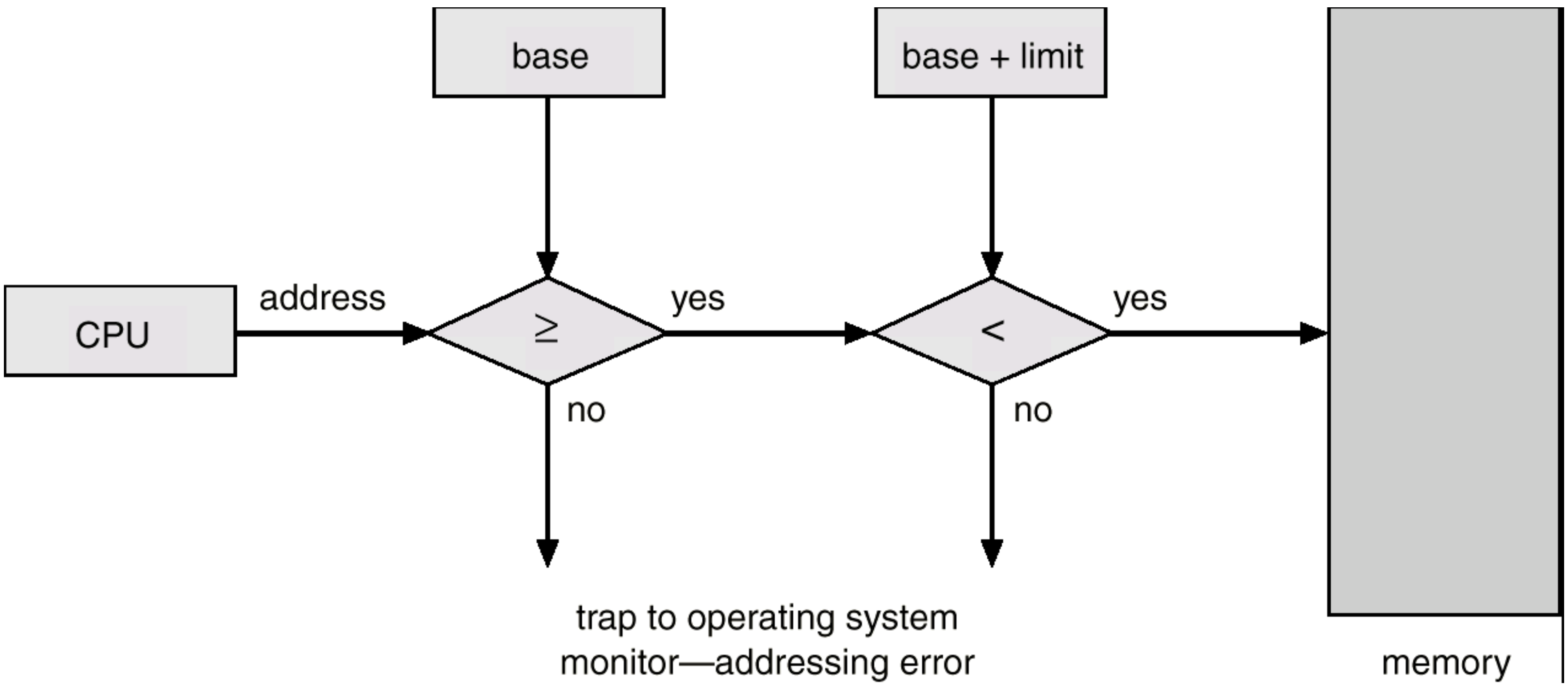
2. Memory Protection

- 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.

Use of Base and Limit Registers

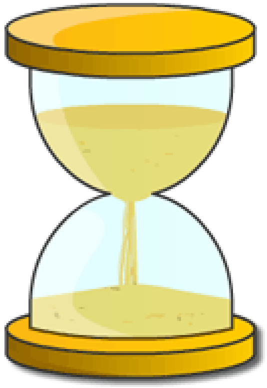


Hardware Address Protection



3. CPU Protection

- Timers
  - interrupts CPU after specified period
- Timer is decremented every clock tick



- Use of timers
  - Time sharing systems
    - Interrupt every N milliseconds
    - Switch control to other processes
  - Computing current time