Computer System Hardware

Reading:

Silberschatz

chapter 2

Additional Reading:

Stallings

chapter 1, 2

Outline

- Computer Hardware
- CPU Components
- Registers
- Instruction Execution
- Interrupt and Trap
- Memory
  - Hierarchy
  - Cache
- Dual Mode Operation
- Protection
  - Hardware
  - Memory
  - CPU, I/O

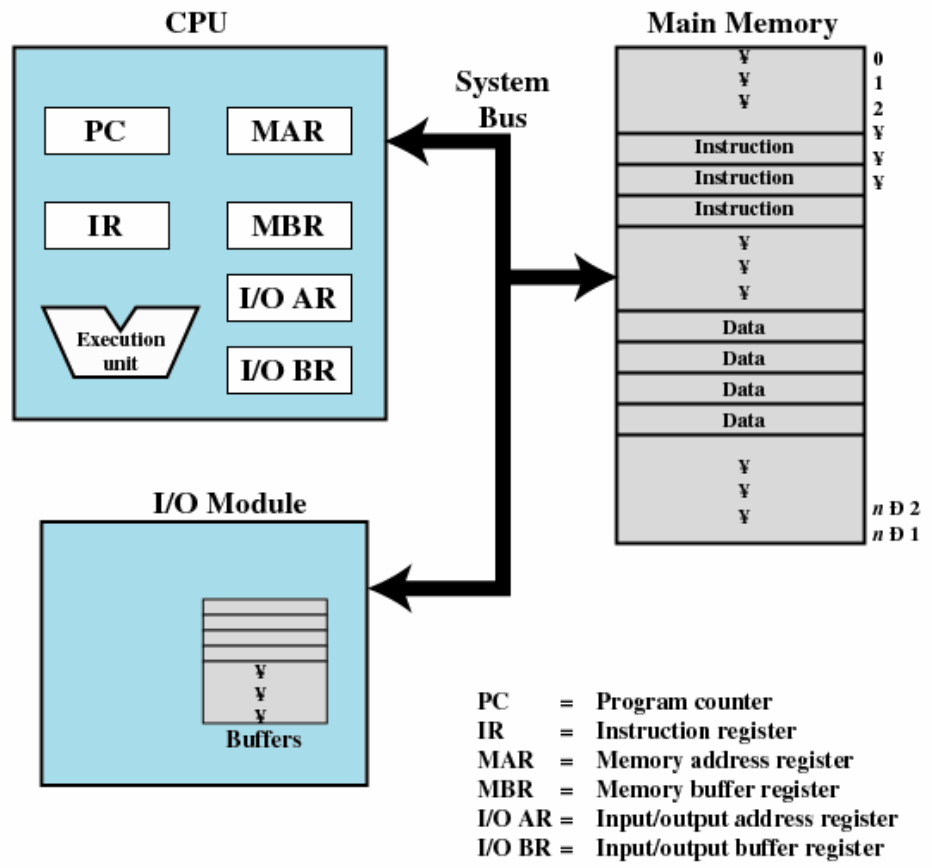
Computer Hardware

- Why study hardware?

  OS exploits hardware resources to provide set of services

- Key Elements
  - CPU or Processor
  - Main Memory
    - typically volatile
    - also referred as real memory or primary memory
  - I/O modules
    - secondary memory devices
    - communications equipment
    - terminals
  - System bus
    - communication among processors, memory, and I/O modules

CPU Components



Computer Components: Top-Level View

Processor Registers

- User-Visible registers
  - Helps programmer to minimize main memory references
  - Typically two types of registers
    1. Data registers
    2. Address registers
       - Index
       - Segment pointer
       - Stack pointer

User-Visible Registers

- Data registers
  - General purpose (programmers or machine)
  - May be dedicated (floating-point and integer operations)
- Address registers
  - Index
    - Involves adding an index to base value to get effective address
  - Segment pointer
    - When memory is divided into segments, memory is referenced by a segment and an offset
  - Stack pointer
    - Points to top of stack

Control and Status Registers

Mostly not visible to user, organization is machine specific

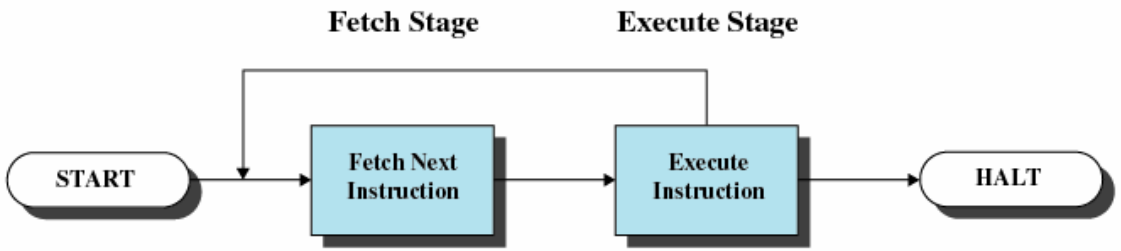
- Program Counter
  - Contains the address of instruction to be fetched
- Instruction Register
  - Contains the instruction most recently fetched
- Program Status Word (PSW)
  - Condition codes
  - Interrupt enable/disable
  - Supervisor/user mode

Control and Status Registers

- Condition Codes or Flags
  - Bits set by the processor hardware as a result of operations
  - Examples
    - Positive result
    - Negative result
    - Zero
    - Overflow

Instruction Execution

- Instruction Cycle – processing required for single instruction
- Two step processing in simplest form
  - Processor reads instructions from memory one at time
    - Fetches
  - Processor executes each instruction



Basic Instruction Cycle

Instruction Fetch and Execute

- The processor fetches the instruction from memory
- Program counter (PC) holds address of the instruction to be fetched next
- Processor increments the PC after each fetch

Instruction Register

- Fetched instruction is loaded in IR
- Instruction categories
  - Processor-memory
    - Transfer data between processor and memory
  - Processor-I/O
    - Data transferred to or from a peripheral device
  - Data processing
    - Arithmetic or logic operation on data
  - Control
    - Alter sequence of execution

Interrupts

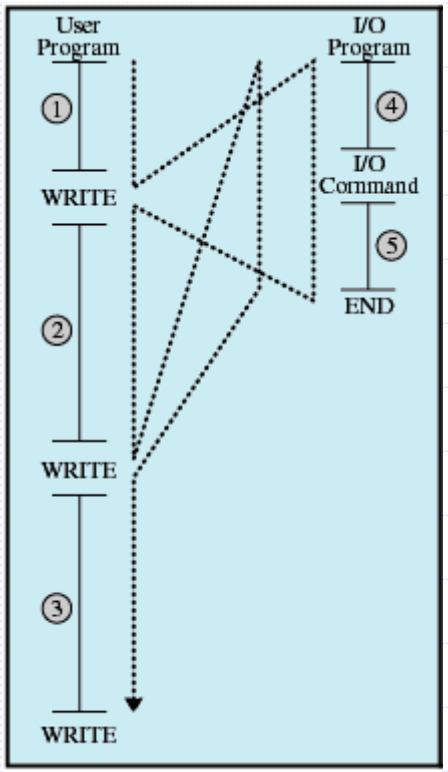
- Common Class of Interrupts
  - Program
    - Result of instruction execution, e.g. arithmetic overflow, division by zero or illegal memory reference
  - Timer
    - Processors timer, functions on regular basis
  - I/O
    - Generated by I/O controller, signal or error
  - Hardware Failure
    - Power failure or memory parity error

Interrupt and Trap

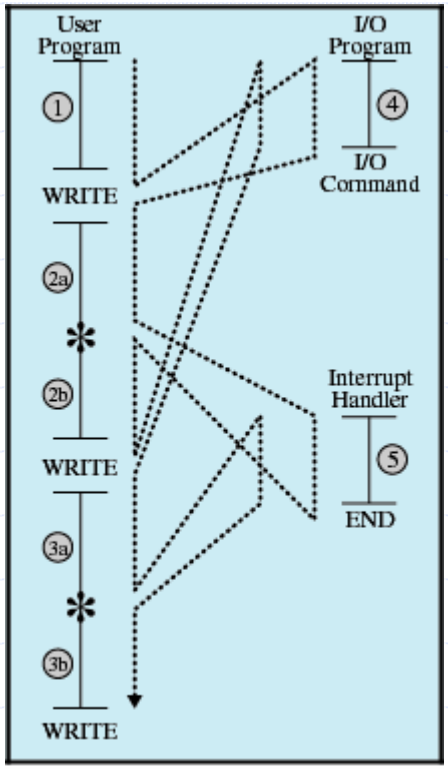
Mechanisms to interrupt the normal processing of processor

- Trap
  - Trap is the notification of an internal event, highest priority
  - Traps are immediate and occur synchronously with the current activity of processor (result of program execution)
- Interrupt
  - Interrupt is the notification of an external event
  - Occur asynchronously with the current activity of processor

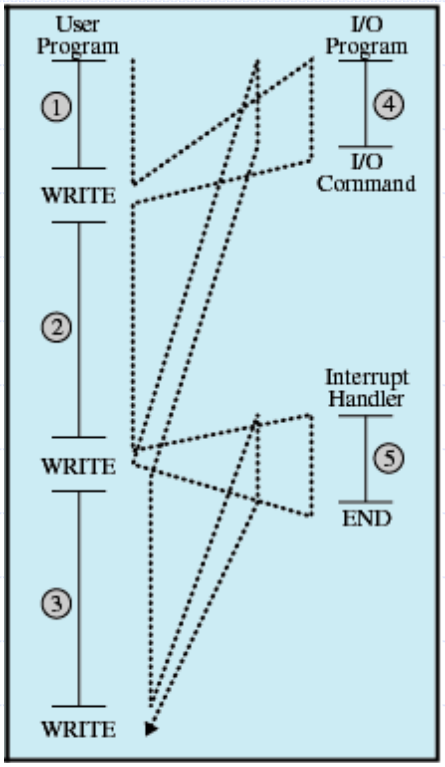
Program Flow of Control



(a) No Interrupts

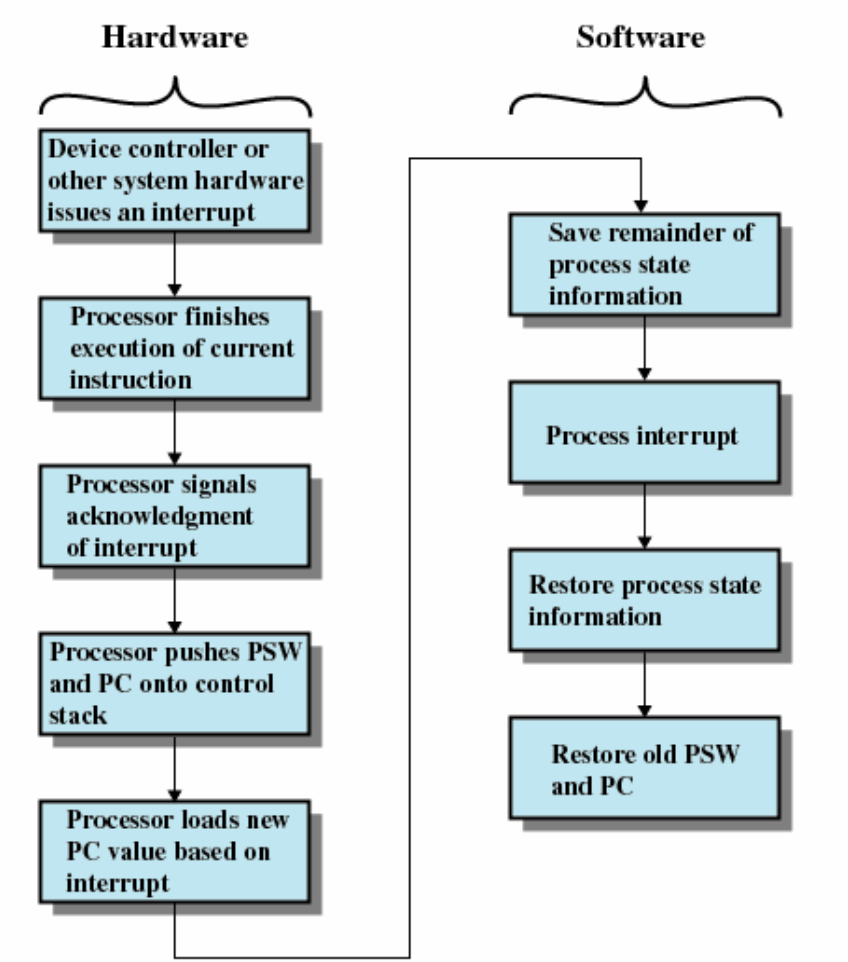


(b) Interrupts; short I/O wait



(c) Interrupts; long I/O wait

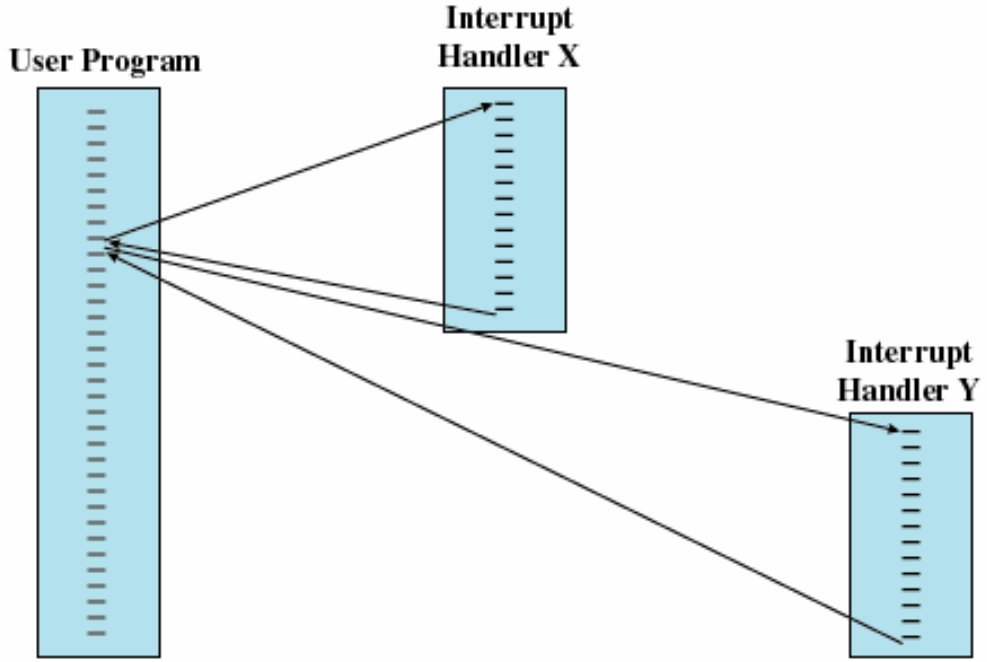
Interrupts



Simple Interrupt Processing

Multiple Interrupts

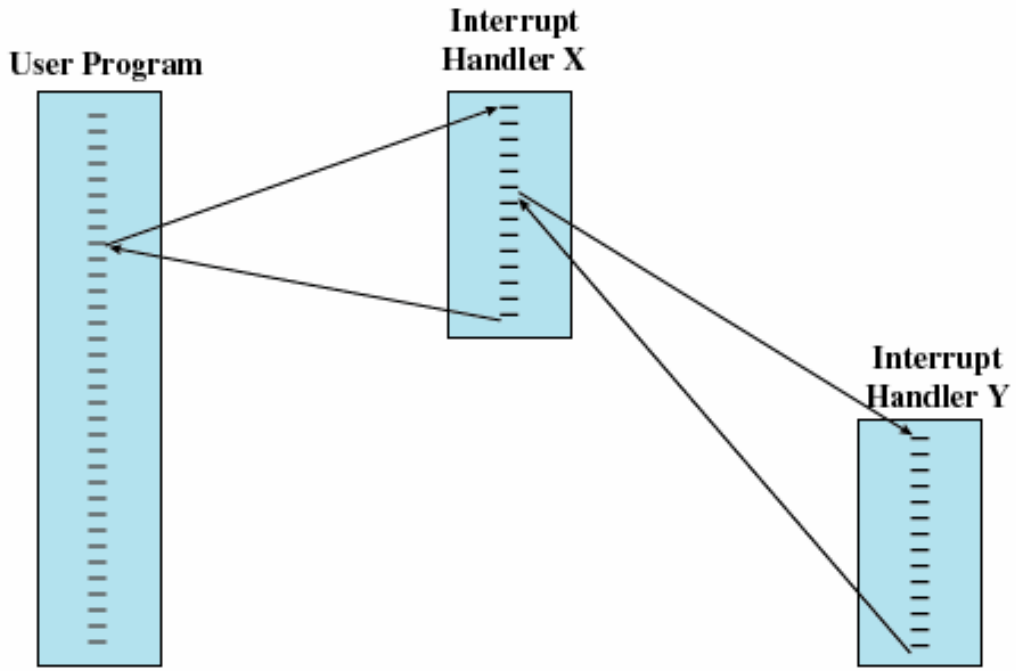
- Disable new interrupts while an interrupt is being processed



Sequential interrupt processing

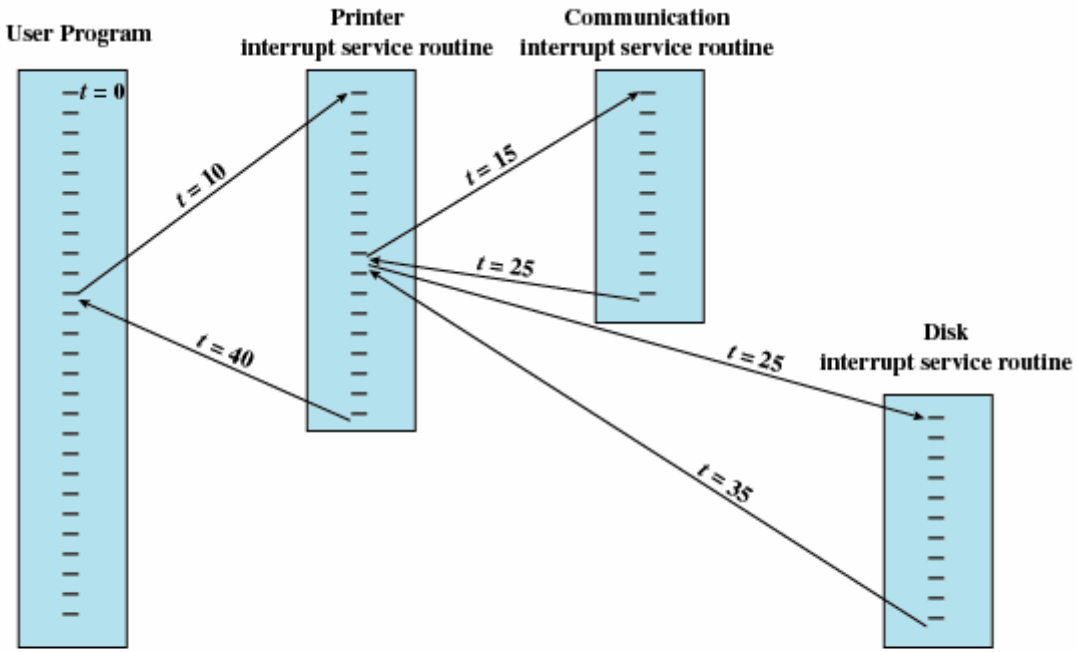
Multiple Interrupts

- Disadvantages of *sequential interrupt processing*
  - Does not account for relative priority or time critical needs
  - I/O device may fill and overflow
- Define priorities and serve in order



Nested interrupt processing

Multiple Interrupts

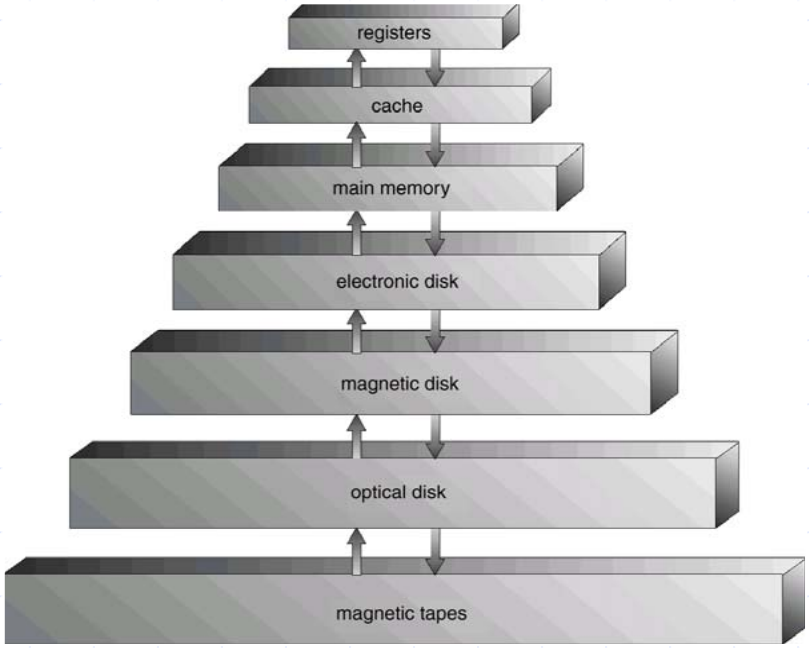


Example: Time sequence of multiple interrupts with priorities

Memory Hierarchy

- Design Constraints
  - How much?
    - Open ended, large capacity
  - How fast?
    - Mach the processor, do not wish to wait
  - How expensive?
    - Reasonable relationship to other components
- Tradeoff among three components
  - Faster access time, greater cost per bit
  - Greater capacity, smaller cost per bit
  - Greater capacity, slower access speed

Memory Hierarchy



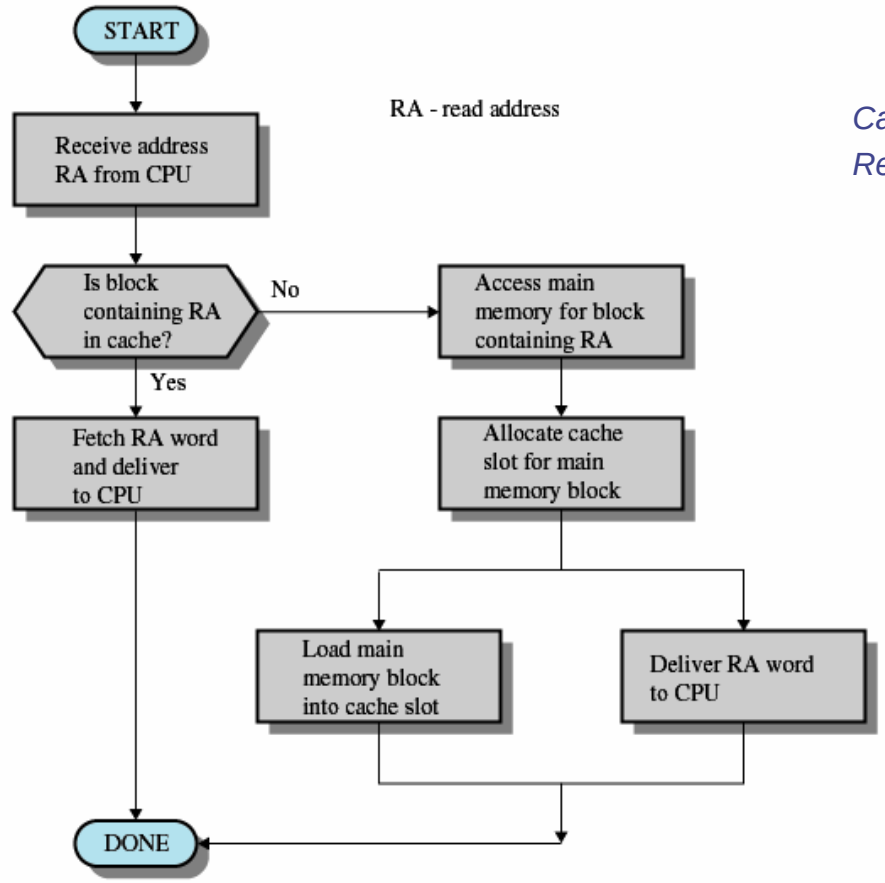
- As one goes down
  - Decreasing cost per bit
  - Increasing capacity
  - Increasing access time

Key to success – Decrease the frequency of access

Caching

- Copying information into faster storage system; main memory can be viewed as a fast *cache* for secondary storage
- Use of high-speed memory to hold recently-accessed data
- Requires a *cache management* policy

Caching

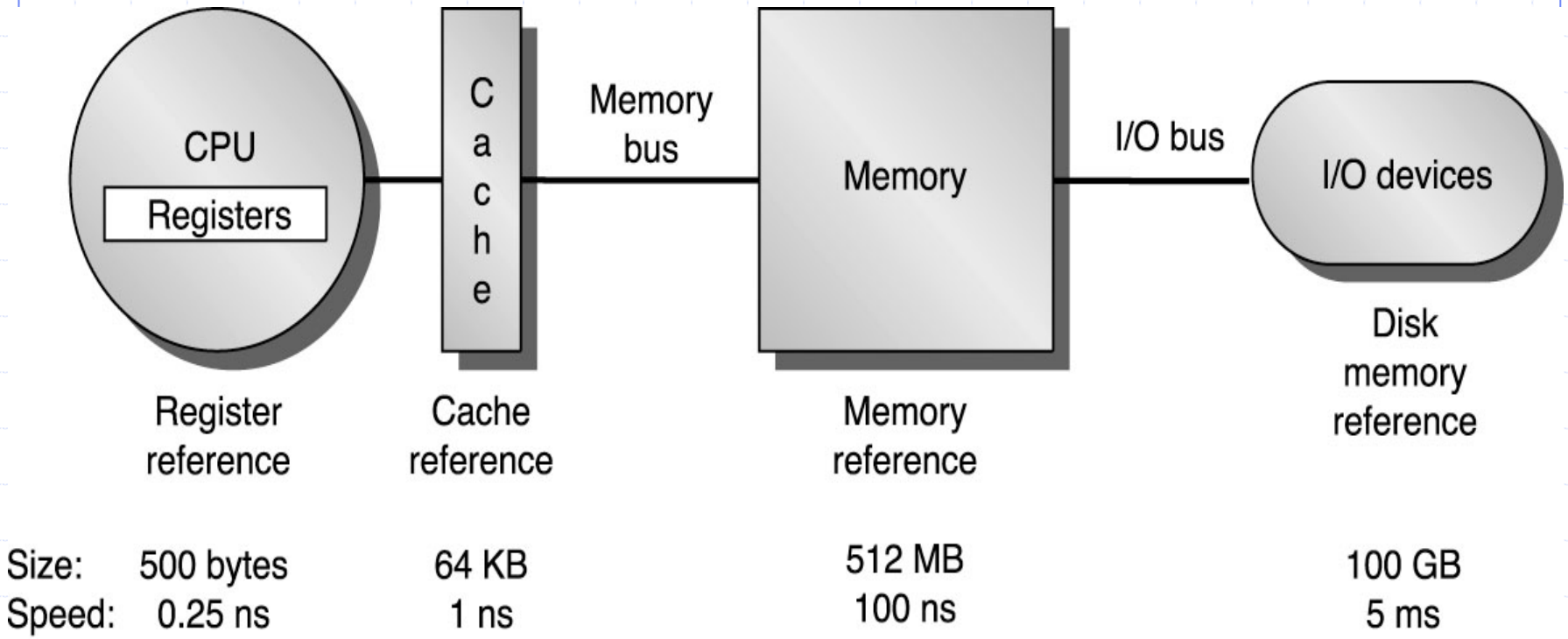


Cache size

Replacement policy

Cache Read Operation

Typical Memory Hierarchy



Hardware Protection

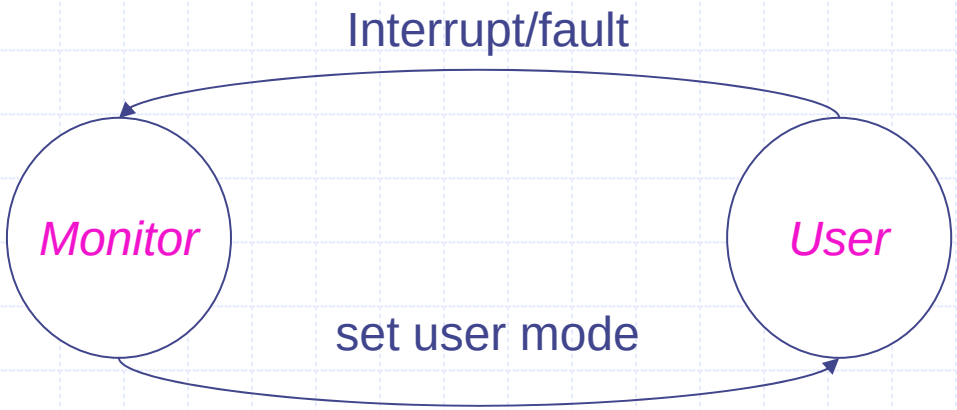
- Sharing of Resources → Utilization and Problems
- OS must ensure that an incorrect program cannot cause other programs to execute incorrectly
- Hardware Protection
  - Dual-Mode Operation
  - I/O Protection
  - Memory Protection
  - CPU Protection

Dual-Mode Operation

- Provide hardware support to differentiate between at least two modes of operations
  - User Mode → execution done on behalf of a user
  - Monitor Mode (also *kernel mode* or *system mode*) → execution done on behalf of operating system

Dual-Mode Operation

- *Mode bit* added to computer hardware to indicate current mode: *Monitor* (0) or *User* (1)
- When an interrupt or fault occurs hardware switches to monitor mode



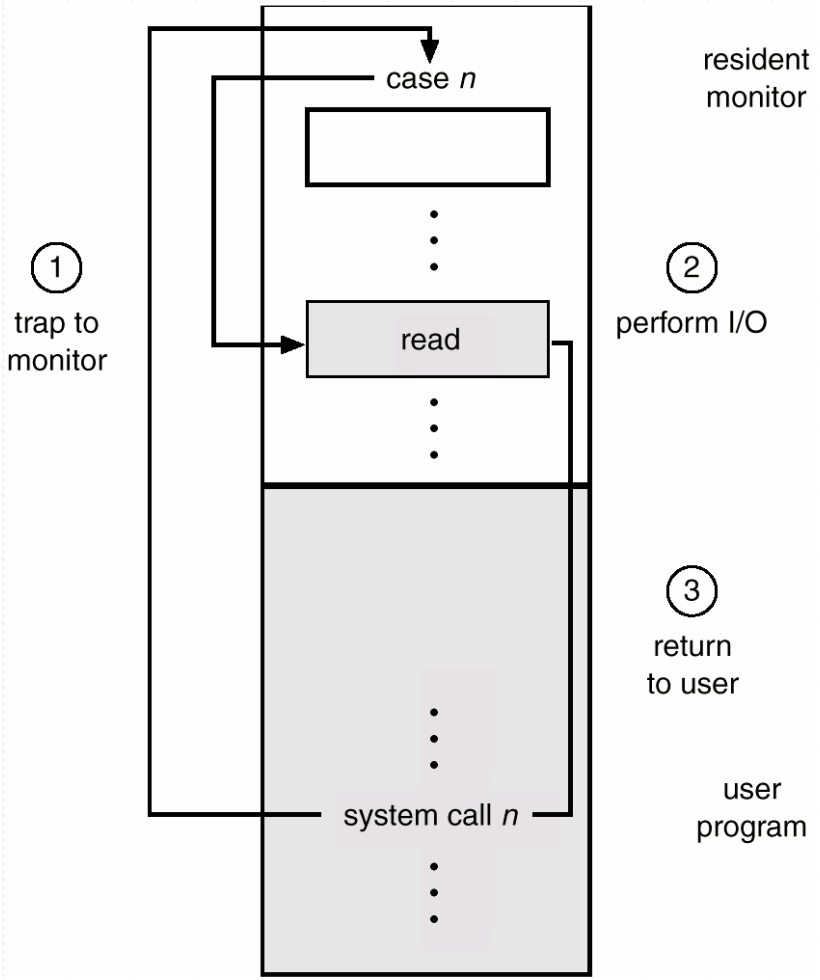
- *Privileged instructions* can be issued only in monitor mode
- *MSDOS* (8088 architecture) – *No Mode bit*
- *Win 2000, IBM OS/2 (advanced versions of Intel CPU, Pentium)*

provide dual-mode operation

I/O Protection

- All I/O instructions → *privileged instructions*
- Must ensure that a user program could never gain control of the computer in monitor mode
- All I/O instructions → through OS, checks if valid

I/O Protection

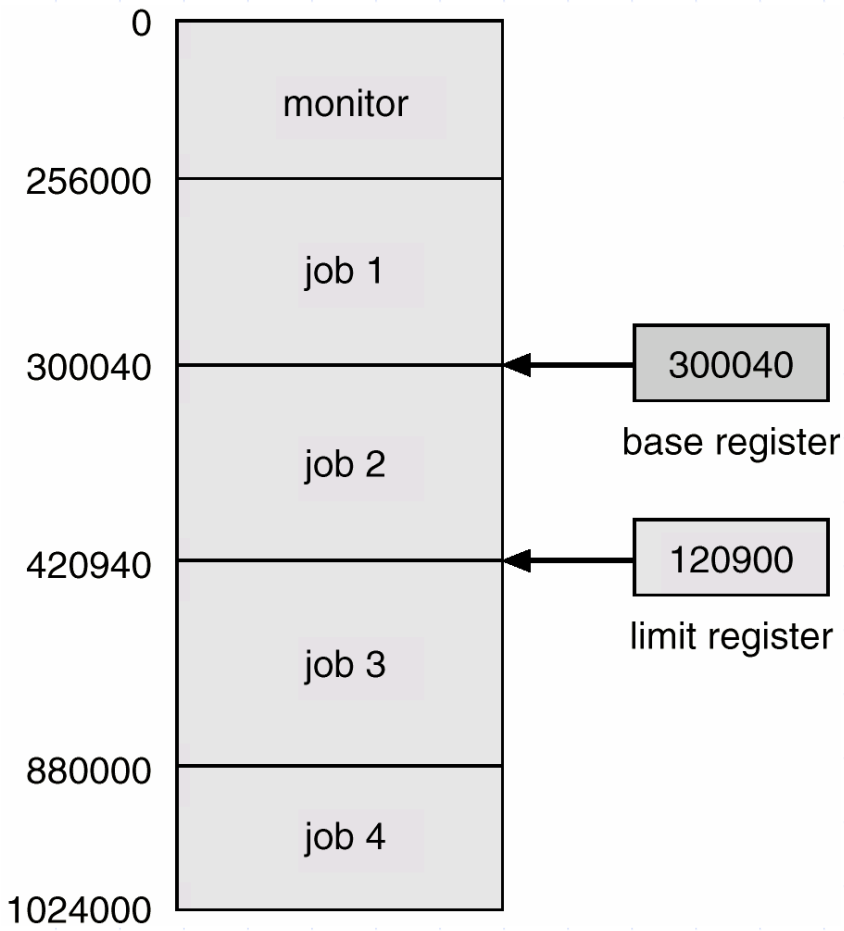


Use of system call to perform I/O

Memory Protection

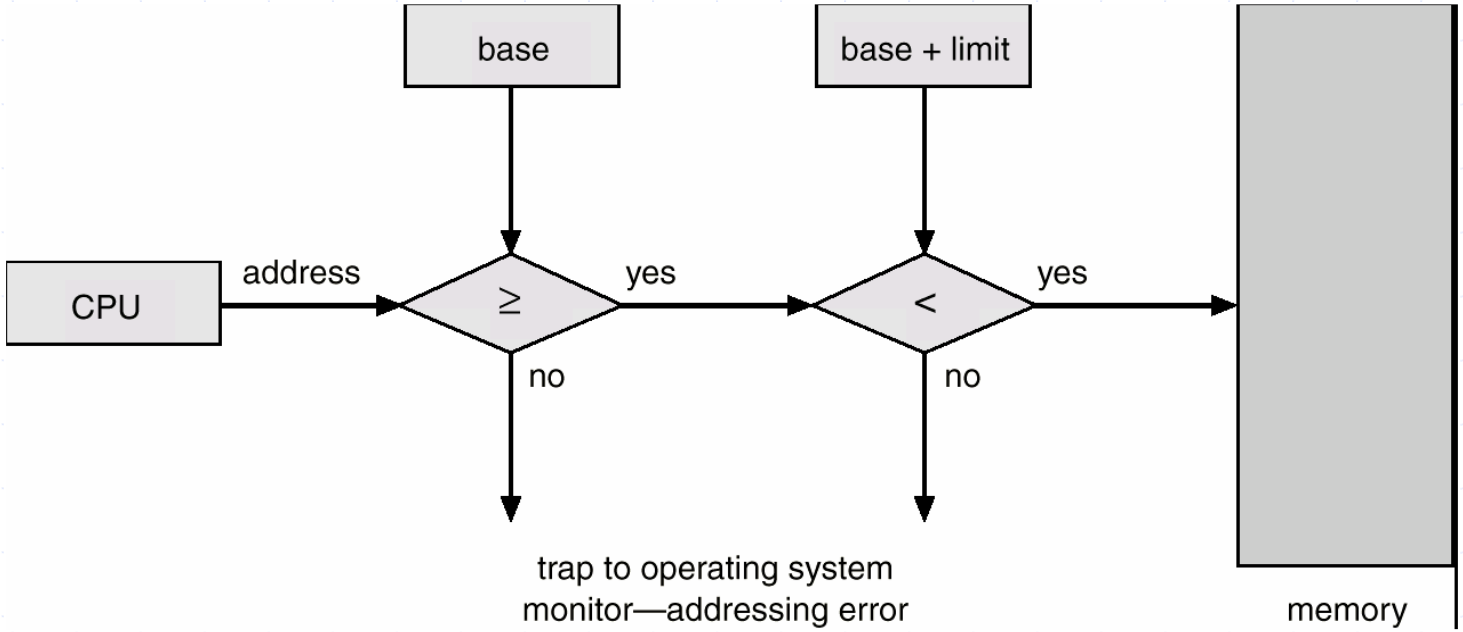
- Memory protection → Prevent interrupt vector and interrupt service routines from user program
- Range of legal addresses - two registers
  - **Base register** → holds the smallest legal physical memory address
  - **Limit register** → contains the size of the range
- Memory outside the defined range is protected

Memory Protection



Use of *base* and *limit* register

Hardware Address Protection



Hardware Protection

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

CPU Protection

- Prevent user program from *struck in infinite loop* or *never returning control to OS*
- *Timer* – interrupts computer after specified period to ensure OS 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
- Load-timer is a privileged instruction