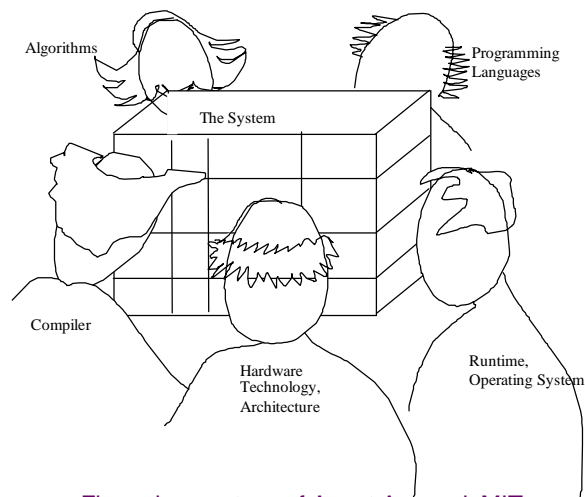


## Chapter 2: Computer-System Structures

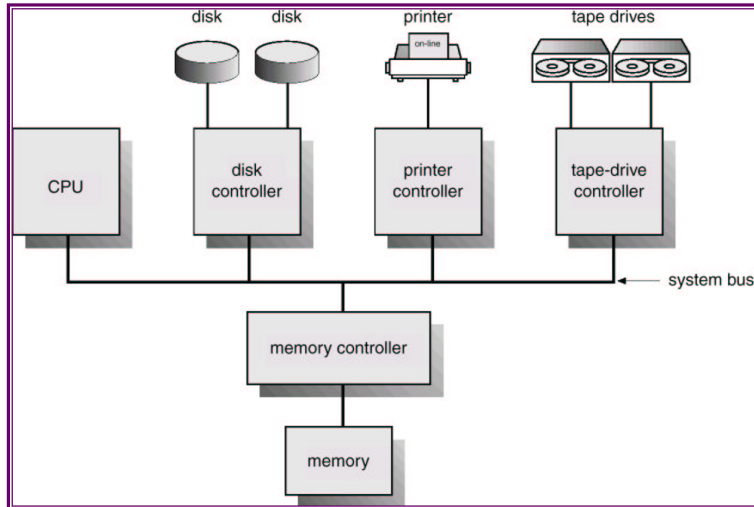
- Last lecture: why study operating systems?
- Purpose of this lecture: general knowledge of the structure of a computer system and understanding technology trends
- Key issues in a computer system
  - ◆ General System Architecture (CPU, \$s, MM, disk, bus, IO devices and controllers), Uni vs. Multi Processors
  - ◆ I/O Structure (IO interrupts, IO methods, HW support, e.g., DMA)
  - ◆ Storage Structure (CPU regs, \$, MM, disk)
  - ◆ Storage Hierarchy (why? expensive→cheap; small→large)
  - ◆ Hardware Protection (user/system, IO protection, Mem protection)

## Hmm ... this looks like a Computer System?

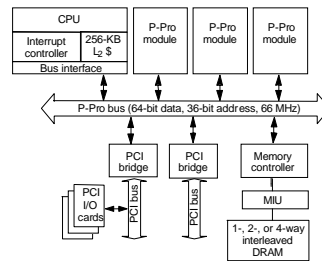
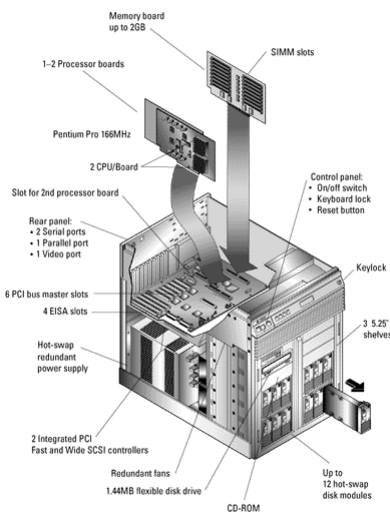


◆ Figure by courtesy of Anant Agarwal, MIT

## Uniprocessor Computer Architecture

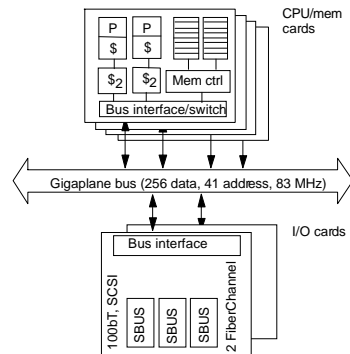


## MP Example: Intel Pentium Pro Quad



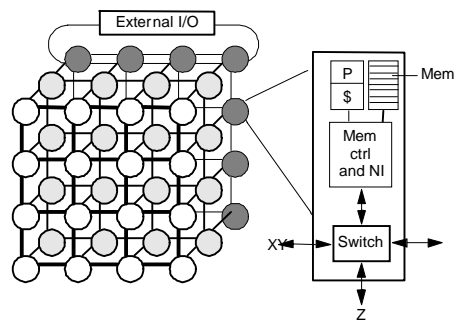
- ◆ Multiprocessor
- ◆ All coherence and multiprocessing glue in processor module
- ◆ Highly integrated, targeted at high volume

## Example: SUN Enterprise



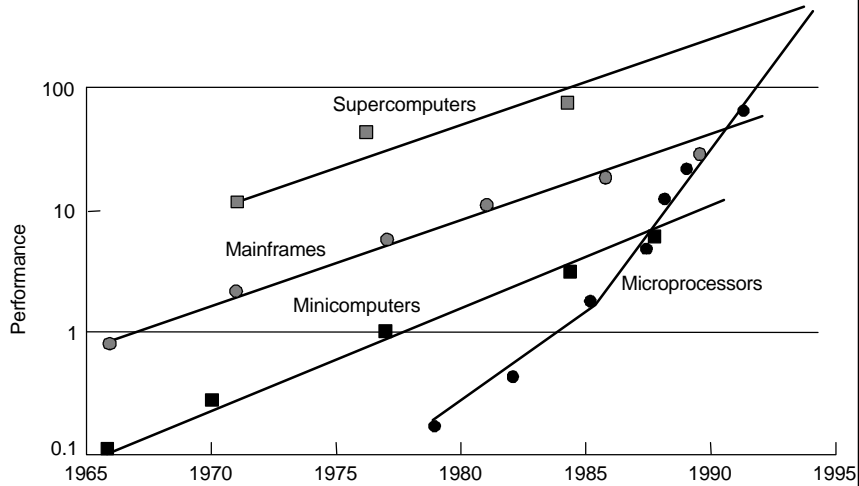
- ◆ 16 cards of either type: processors + memory, or I/O
- ◆ All memory accessed over bus, so symmetric multiproc. (SMP)
- ◆ Higher bandwidth, higher latency bus

## Example: Cray T3E



- Multiprocessor system
- Scale up to 1024 processors, 480MB/s links

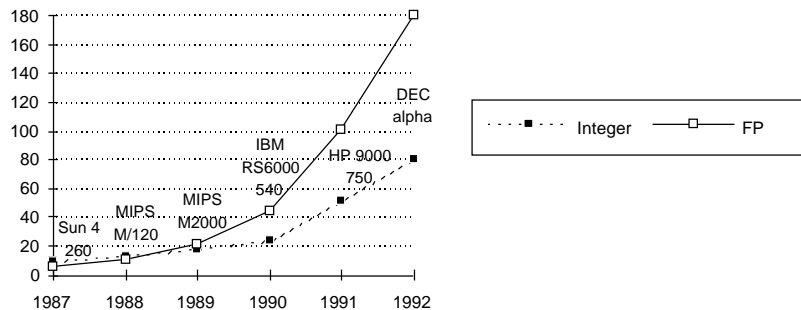
## Let's look at trends, 1<sup>st</sup> Technology Trends



The natural building block for multiprocessors is now also about the fastest!

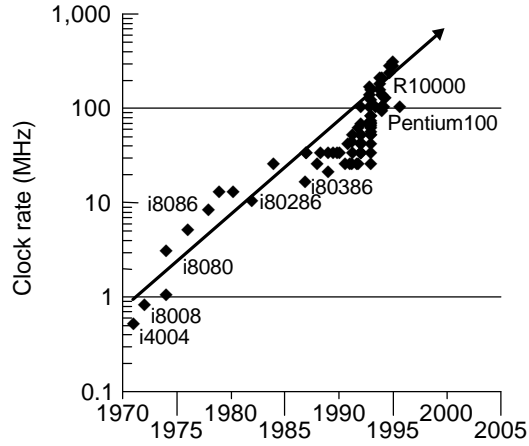
## General Technology Trends

- *Microprocessor performance* increases 50% - 100% per year
- *Transistor count* doubles every 3 years
- *DRAM size* quadruples every 3 years
- Huge investment per generation is carried by huge commodity market



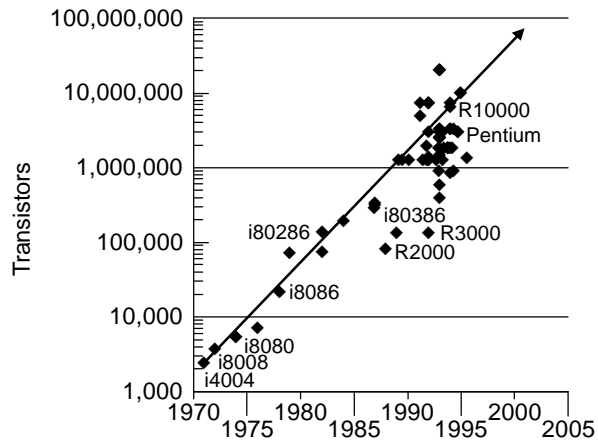
- Not that single-processor performance is plateauing, but that parallelism is a natural way to improve it.

## Clock Frequency Growth Rate



- 30% per year

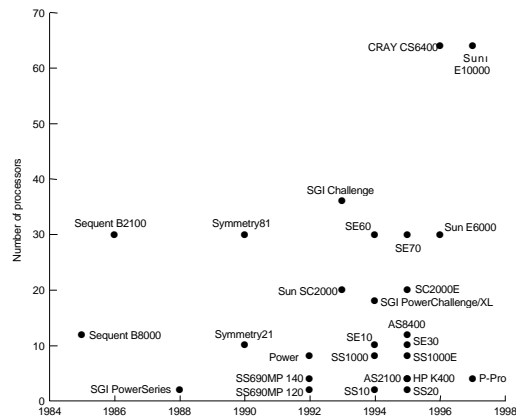
## Transistor Count Growth Rate



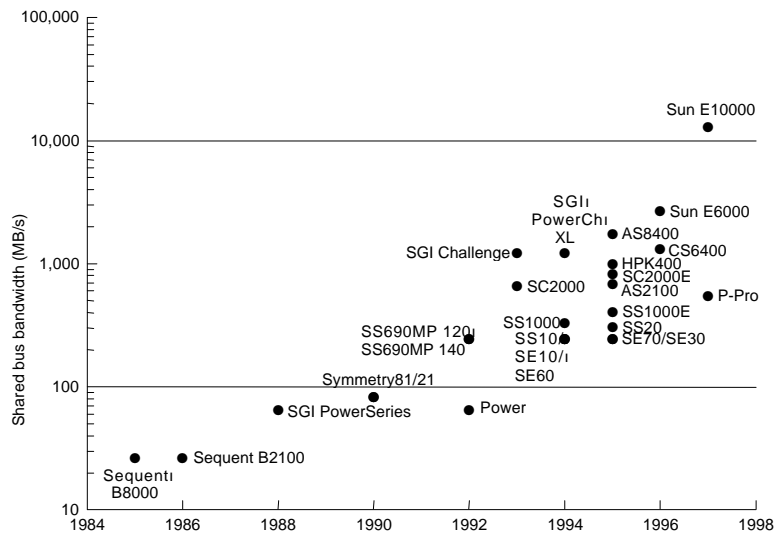
- 100 million transistors on chip by early 2000's A.D.
- Transistor count grows much faster than clock rate
  - 40% per year, order of magnitude more contribution in 2 decades

## Architectural Trends: Bus-based MPs

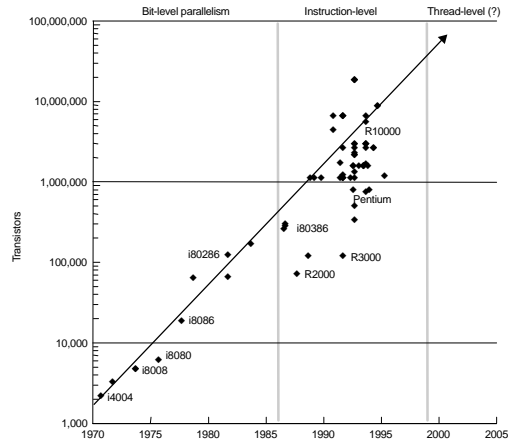
- Micro on a chip makes it natural to connect many to shared memory
  - dominates server and enterprise market, moving down to desktop
- Faster processors began to saturate bus, then bus technology advanced
  - today, range of sizes for bus-based systems, desktop to large servers



## Bus Bandwidth



## Phases in VLSI Generation



- How good is instruction-level parallelism?
- Thread-level needed in microprocessors?

## Economics

- Commodity microprocessors not only fast but CHEAP
  - Development cost is tens of millions of dollars (5-100 typical)
  - BUT, many more are sold compared to supercomputers
  - ◆ Crucial to take advantage of the investment, and use the commodity building block
  - ◆ Exotic parallel architectures no more than special-purpose
- Multiprocessors being pushed by software vendors (e.g. database) as well as hardware vendors
- Standardization by Intel makes small, bus-based SMPs commodity
- Desktop: few smaller processors versus one larger one? Multiprocessor on a chip is here.

## 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*.

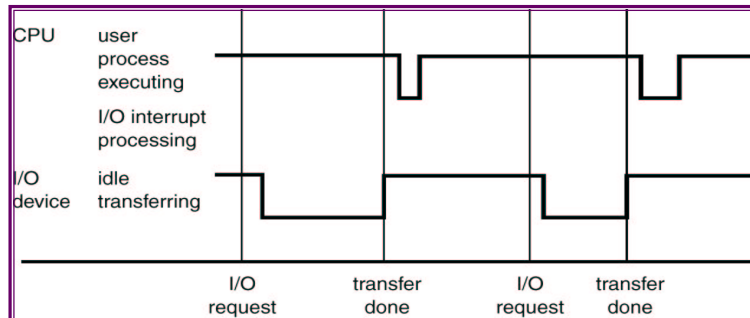
## Common Functions of Interrupts

- 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

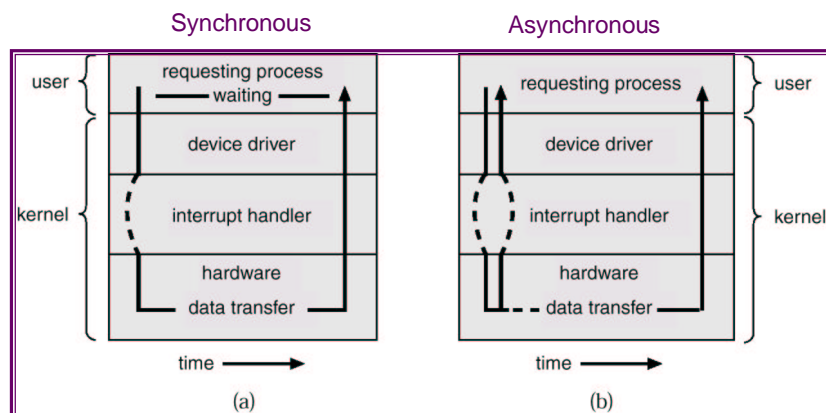
## Interrupt Time Line For a Single Process Doing Output



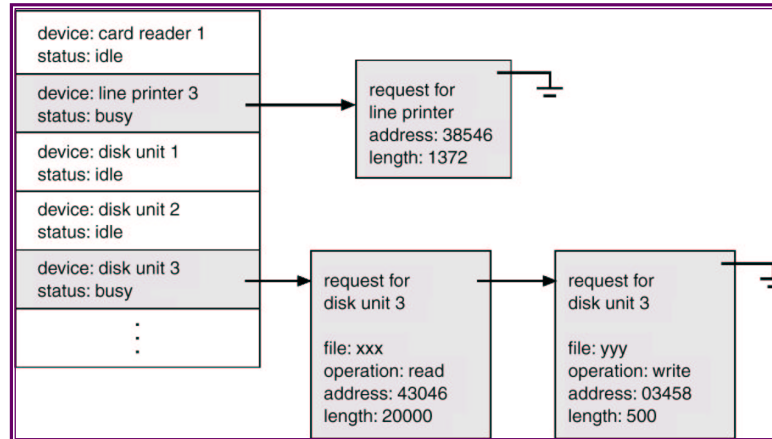
## I/O Structure

- 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



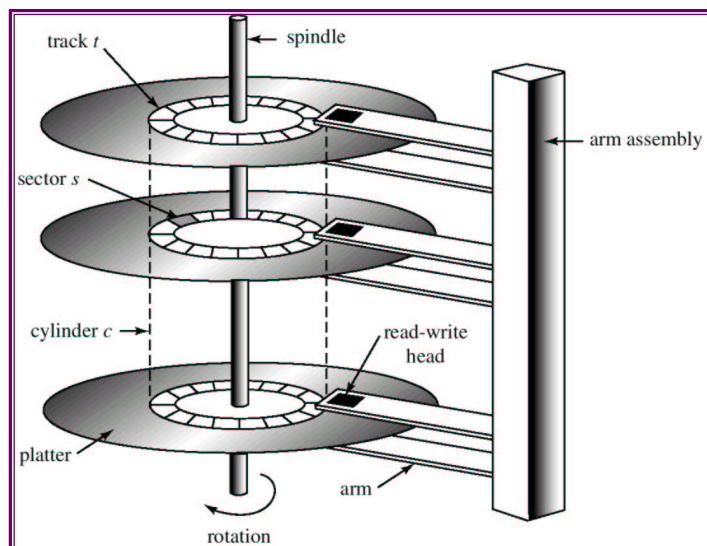
## Direct Memory Access Structure

- 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

- 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.

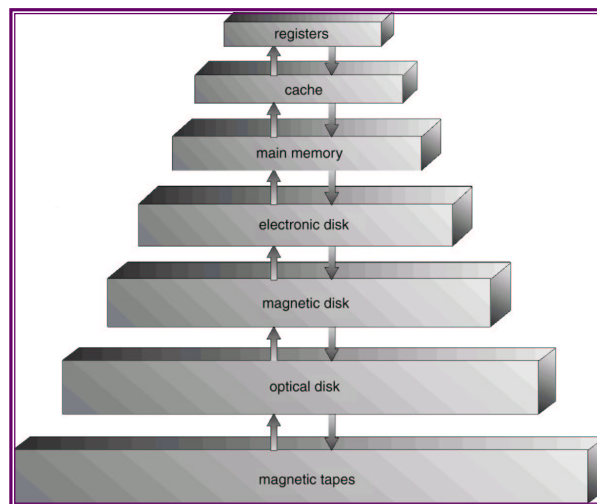
## Moving-Head Disk Mechanism



## 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.

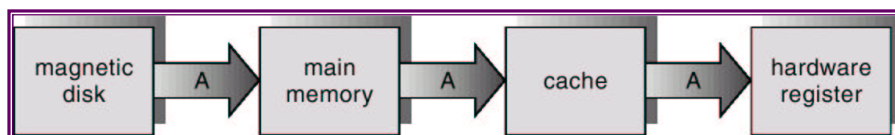
## Storage-Device Hierarchy



## 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*.
- Caching is typically transparent to the OS

## Migration of A From Disk to Register



## Hardware Protection

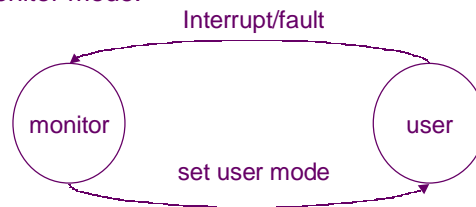
- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

## 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.

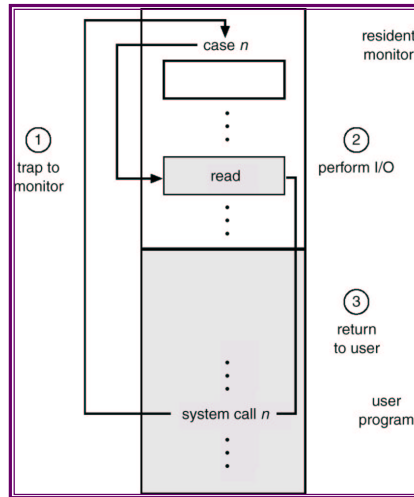


*Privileged instructions* can be issued only in monitor mode.

## 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).

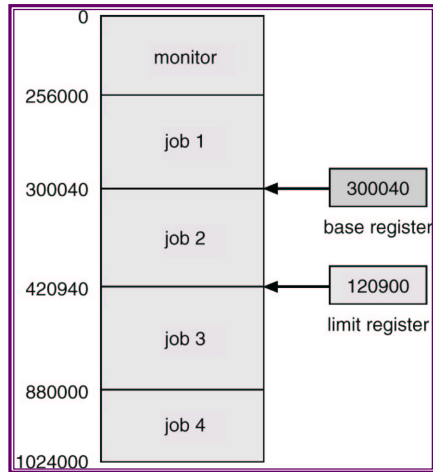
## Use of A System Call to Perform I/O



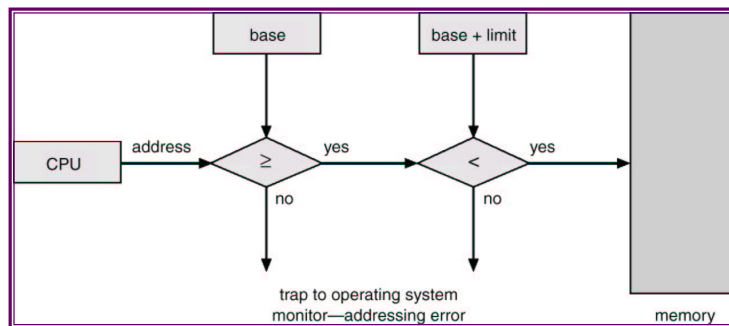
## Memory Protection

- 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 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.

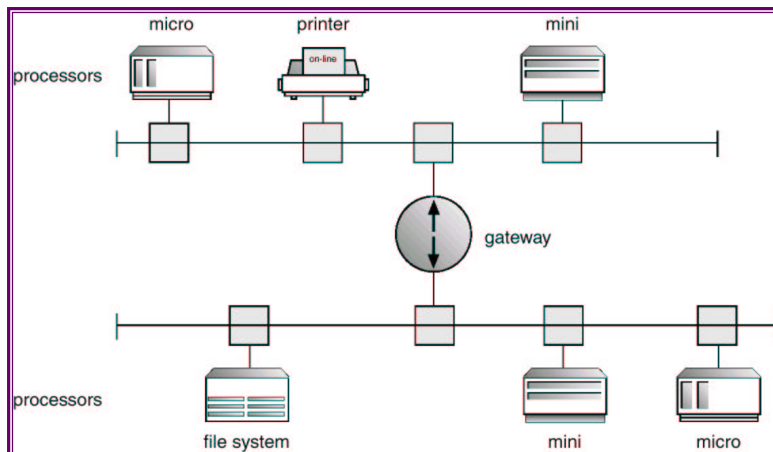
## CPU Protection

- *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

