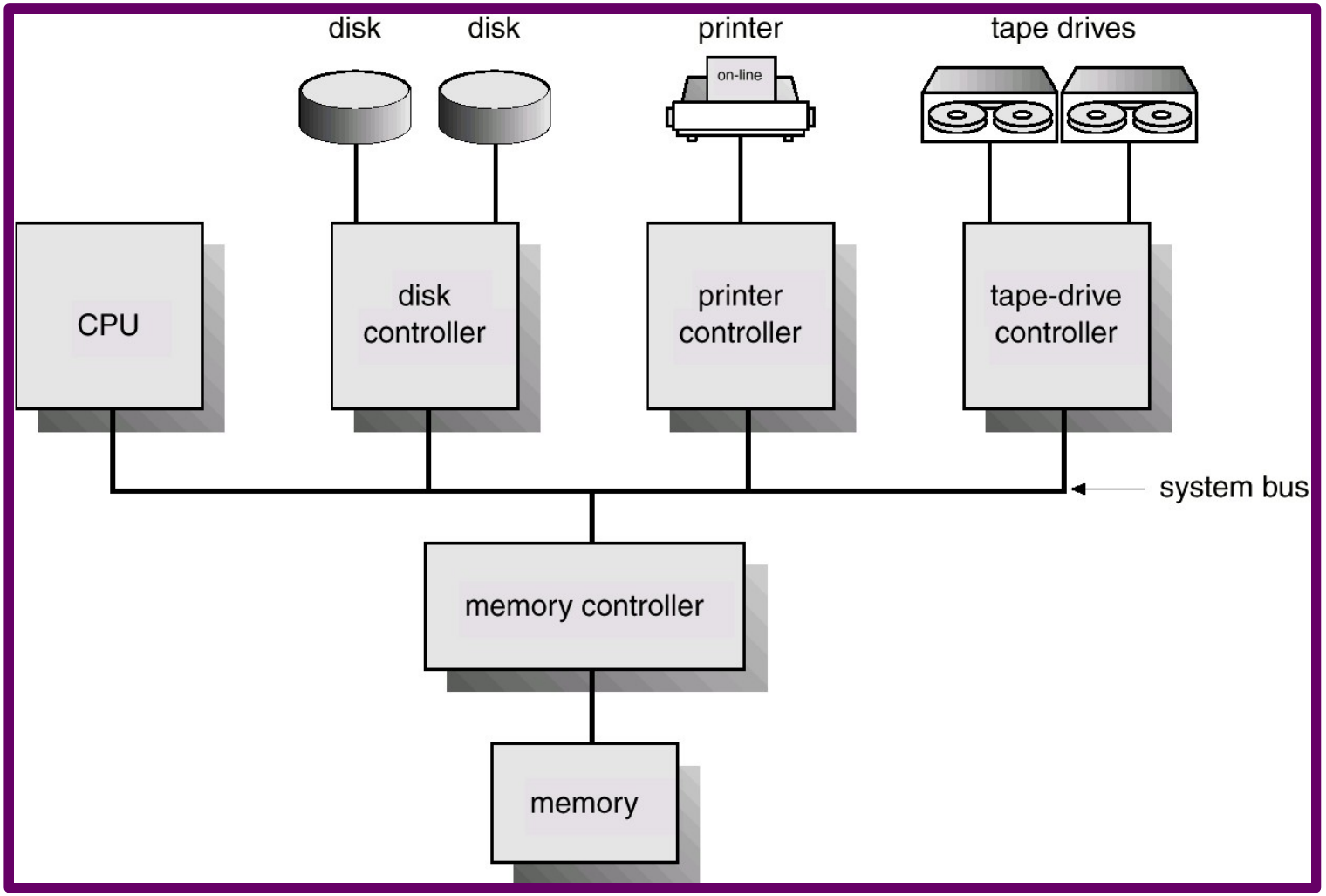
Chapter 2: Computer- System Structures

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

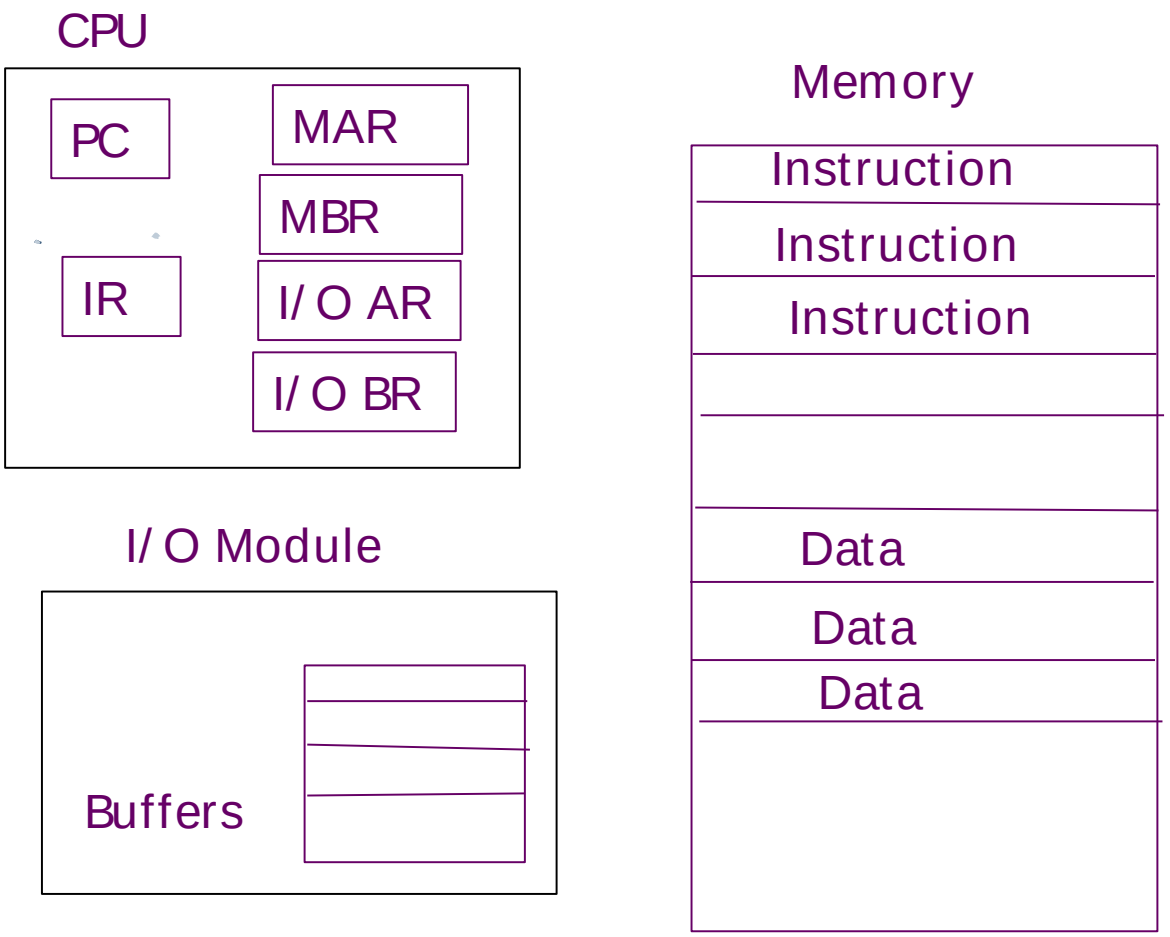
Computer- System Architecture



Computer- Components

- Computer consists of processor, memory, and I/O components, with one or more modules of each type. These modules are connected through interconnection network.
- 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*.

Architecture of a simple computer



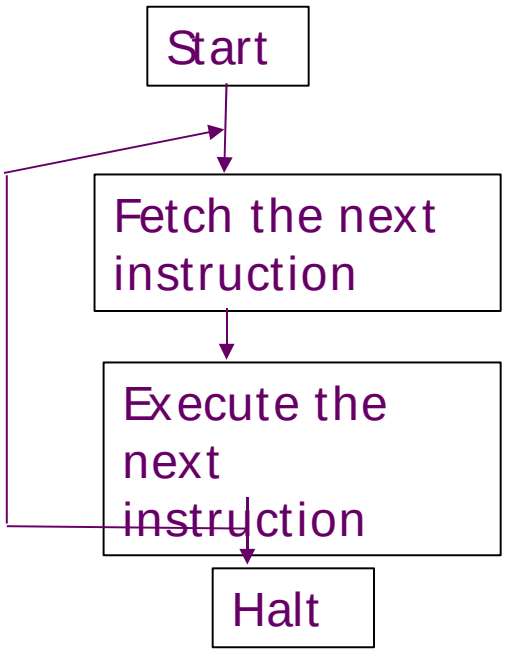
- Processor: Controls the operation of the computer and performs data processing functions. It is called CPU.
- Main memory: Stores data and programs; it is volatile
- I/O modules: Moves data between the computer and external environment.
- System bus: mechanism of communication among processors, main memory, and I/O modules.

Sample Computer

- Operation: Processor controls everything.
  - **MAR**: Memory address register: which specifies the address in memory for the next read or write.
  - **MBR**: Memory buffer register: which contains the data to be written into memory or which receives data read from memory.
  - **I/O AR**: Address of I/O device
  - **I/O BR**: Exchange of data between I/O and computer.
- Processor Registers: Within the processor there is a set of registers that provide a level of memory that is faster and smaller than main memory.
  - User visible registers: Available to programmer
    - ✔ **Data registers**: Can be used by the programmer.
    - ✔ **Address Registers**: Contains Main memory address of data and instructions.
      - **Index register**: Index to base value
      - **Segment pointer**: It contains a reference to a particular segment.
      - **Stack pointer**: Points top of the stack.
  - **Control and status registers**: These are employed to control the operation of the processor. Differ from machine to machine.
  - MAR, MBR,I/O AR, and I/O BR
  - **Program Counter:** Contains the address of the instruction to be fetched.
  - **Instruction Register**: Contains the instruction most recently fetched.
  - **PSW:** Program Status word: It is a register or a set of registers.

Sample Computer…

- **PSW:** Program Status word: It is a register or a set of registers.
  - ✔ **Sign:** contains sign bit of last arithmetic operation
  - ✔ **Zero**: it is set if the result of arithmetic operation is zero
  - ✔ **Carry:** It is set if there is a carry or borrow.
  - ✔ **Equal:** If the compare result is equality
  - ✔ **Overflow:** It is set if the result is overflow.
  - ✔ **Interrupt enable/ disable:** Used to disable or enable interrupts.
  - ✔ **Supervisor:** Indicates whether the processor is executing in supervisor or user mode.



- **Instruction execution:**
  - Program execution is the main function of the computer.
  - Instruction fetch and execute

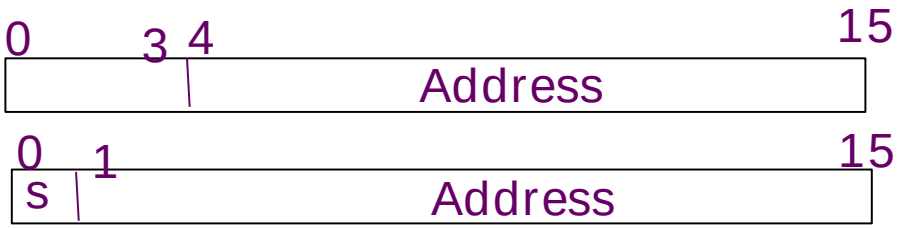
Sample computer…

- **Fetching:** Bringing the instructions from the main memory.
  - PC: Contains the address of the next instruction to be fetched.
    - Incremented unless told otherwise
- **Execute:**
  - The processor interprets the instructions and performs the required action.
    - **Processor- memory:** Transfer data from the memory
    - **Processor- I/ O-** transfer data from peripheral device from the memory.
    - **Data processing:** Arithmetic and logic operations
    - **Control:** The instructions may specify the sequence of the next instruction to be fetched.

Sample Computer..

An Example:

Instruction format



Integer format

PC: address of the instruction
IR: Instruction being executed
AC: Accumulator: temporary storage

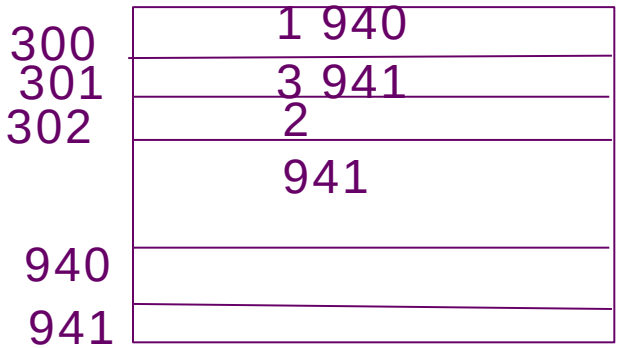
1=0001=LOAD ADDRESS: Load AC from memory

2=0010=STORE ADDRESS: Store AC to memory

3=0101=ADD ADDRESS: Add AC to memory

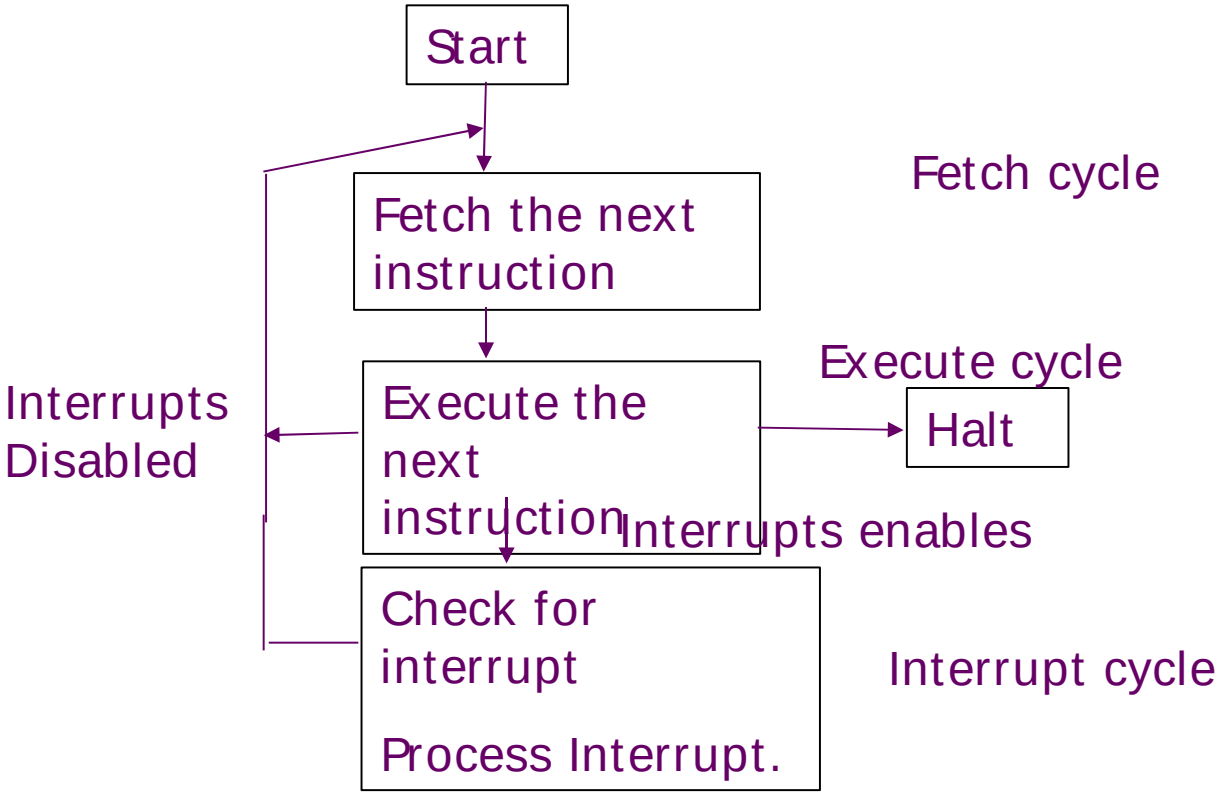
Program

LOAD 940
ADD 941
STORE 941
HLT



Interrupt processing

- To improve the performance, interrupts are provided.
- With interrupts, the processor can be engaged in executing other while an I/O operation is in progress.
- Interrupt cycle is added to the instruction cycle.



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

- To start I/O operation, CPU loads the appropriate registers within the device controller
- The device controller examines the contents of these registers to determine what action to take.
- If it s a read operation the controller transfers the data into local buffer.
- Then it informs the CPU through interrupt.
- The operating system preserves the state of the CPU by storing registers and the program counter.

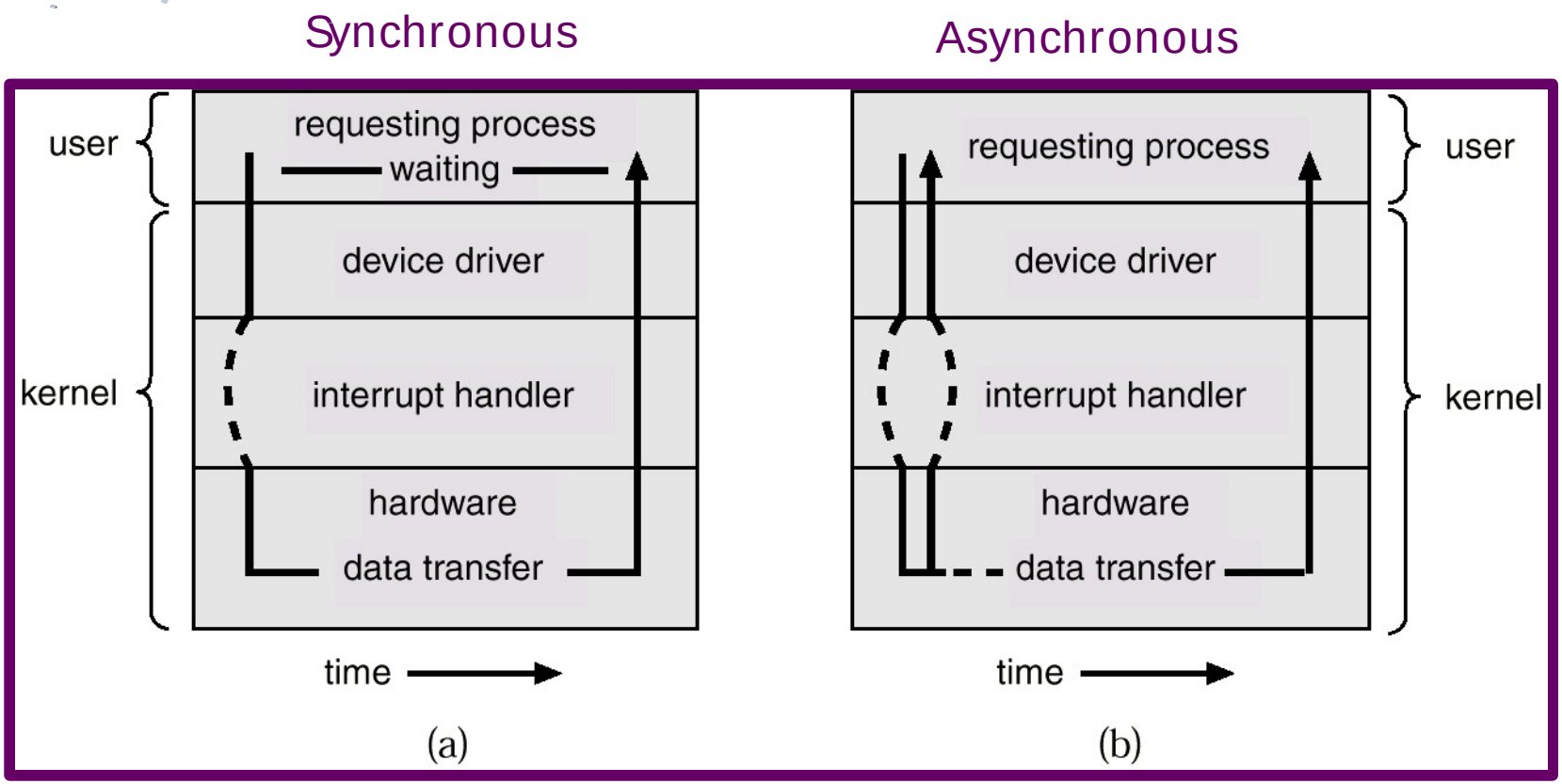
Interrupt Handling...

- Interrupt handler: The CPU hardware has a wire called interrupt request line; that CPU senses after executing every instruction.
- When a CPU senses a signal, it saves the PC, and PSW on a stack and jump to the interrupt handler routine at a fixed address in memory.
- Interrupt vector: It contains the memory addresses of specialized interrupt handlers.

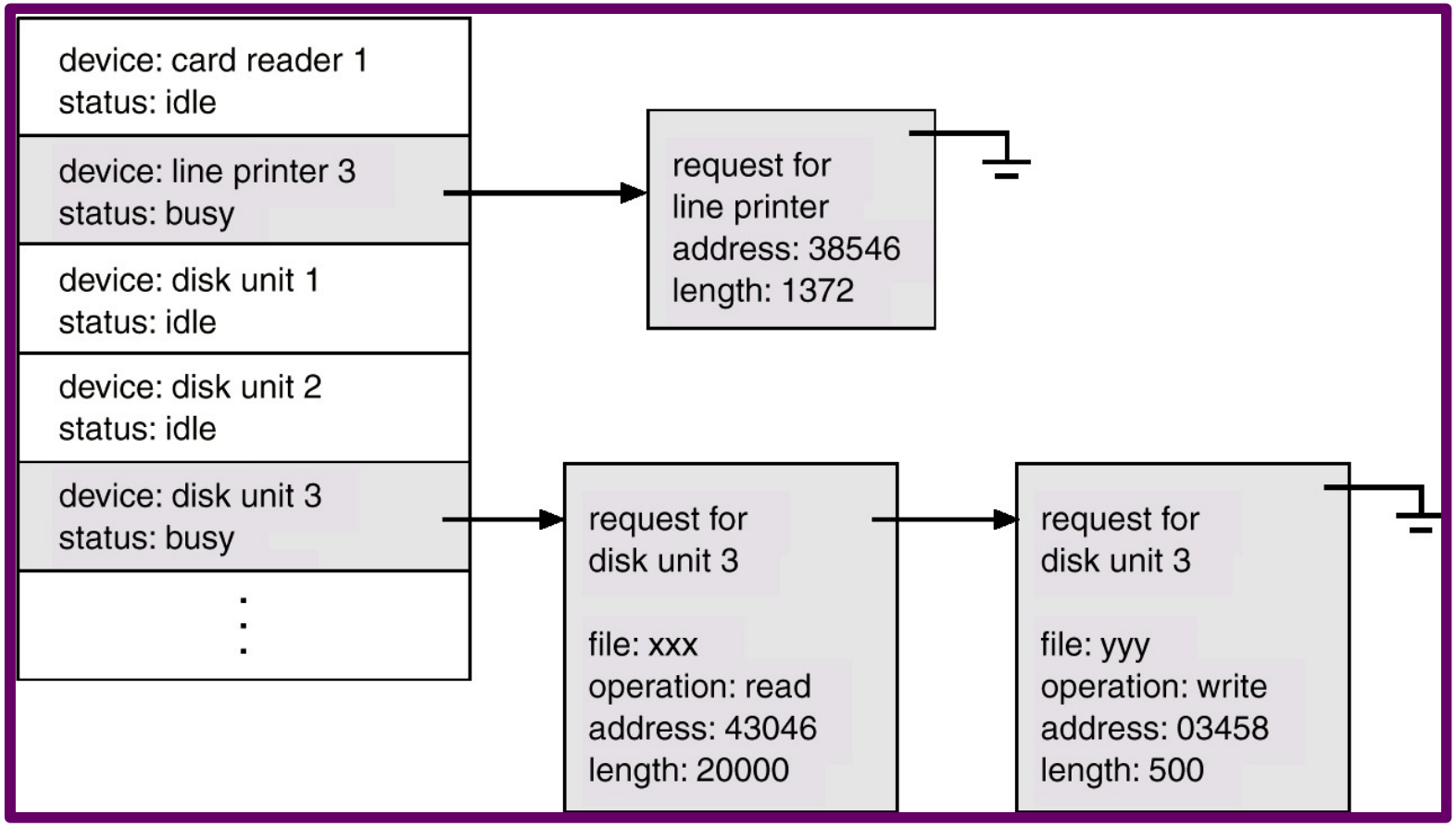
I/O Structure

- Two types of I/O
- **Synchronous I/O**: 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.
- **Asynchronous I/O**: 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.
- Waiting for I/O operation
  - **Loop:** Jump Loop
  - The above loop continues until interrupt occurs.

Two I/O Methods



Device- Status Table



I/O communication techniques

- Three kinds of data transfer
  - **Program data transfer**: CPU checks the I/O status
    - The I/O module does not interrupt the processor.
    - Processor is responsible for extracting the data from main memory and shifting data out of main memory.
    - It is a time-consuming process that keeps the processor busy unnecessarily.
  - **Interrupt driven data transfer:** when I/O is ready it interrupts the CPU
    - In programmed I/O the processor repeatedly interrogate the status of the I/O module.
    - In Interrupt driven data transfer, the I/O module will interrupt the processor to request service when it is ready to exchange data with the processor.

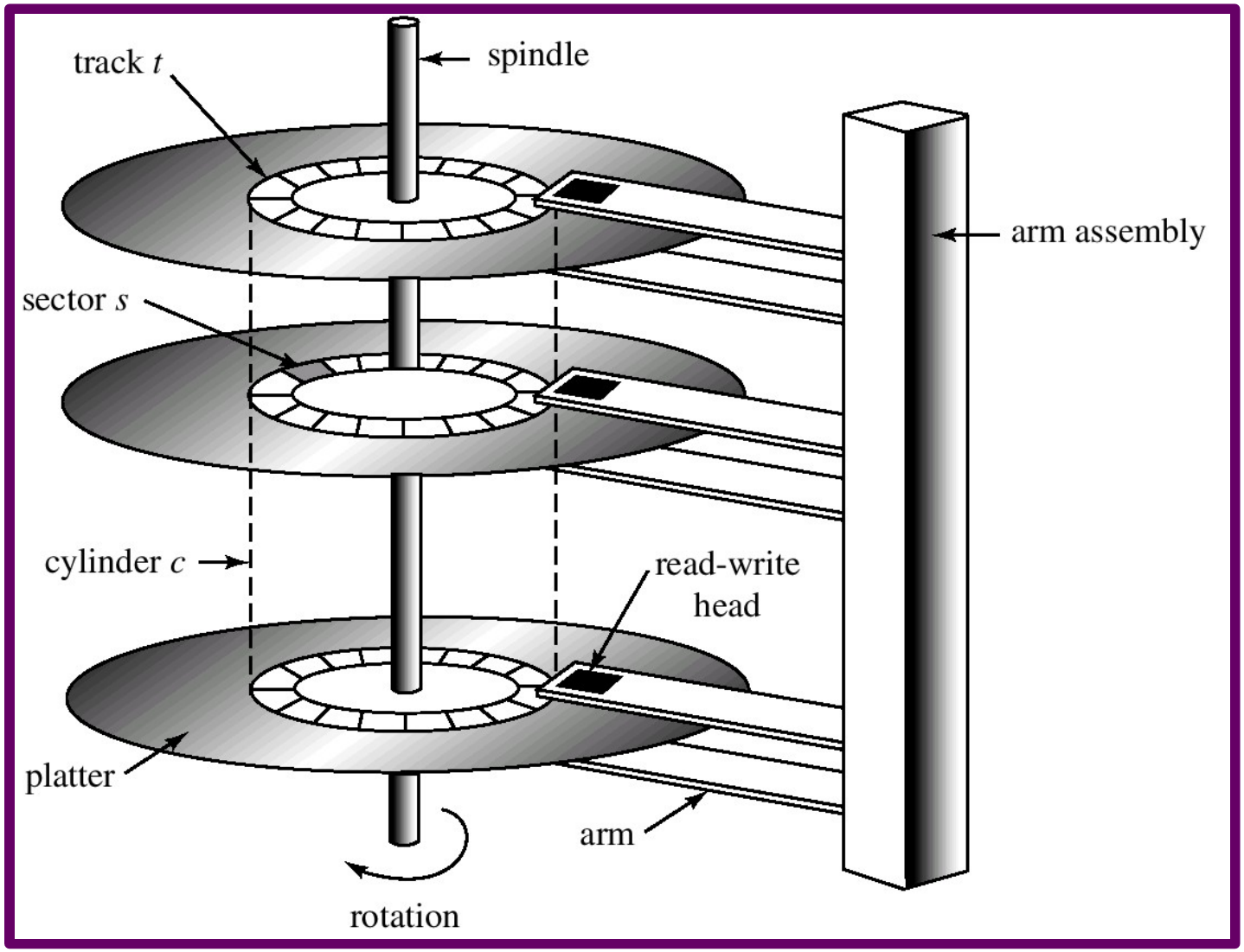
Direct Memory Access (DMA) Structure

- **DMA:**
  - **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 on interrupt is generated per block, rather than the one interrupt per byte.
- **DMA:** Processor issues a command to DMA module by sending the following information.
  - Whether read or write is requested.
  - The address of the I/O device involved.
  - The starting location of the memory to read from or write to.
  - The number of words to be read or written.
- The processor continues other work.
- The DMA module transfers the data and interrupts the processor.
- It takes the control of the bus to transfer data from memory.
- The processor becomes slow, or must wait from the bus.

Storage Structure

- **Main memory** – only large storage media that the CPU can access directly.
  - Two kinds of I/O
    - ✔ Memory mapped I/O:
      - A range of memory addresses are set aside and are mapped to device registers. Reads and writes are transferred to and from the registers.
    - ✔ Programmed I/O: CPU constantly loops to see whether the device is ready
    - ✔ Interrupt driven: I/O device interrupts
    - ✔ DMA:
- **Cache memory**
  - Registers are compatible to CPU: register access may take single clock cycle.
  - Main memory is not; CPU should access MM through bus.
  - Memory access may take many clock cycles
  - A memory buffer that is used to accommodate the speed difference.
- **Secondary storage** – extension of main memory that provides large nonvolatile storage capacity.
- **Electronic disks**
  - Can be volatile or non-volatile
  - Contains DRAM and and magnetic disk and a battery for backup power.
- **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.
  - The host controller is a controller at the computer and the bus.

Moving- Head Disk Mechanism

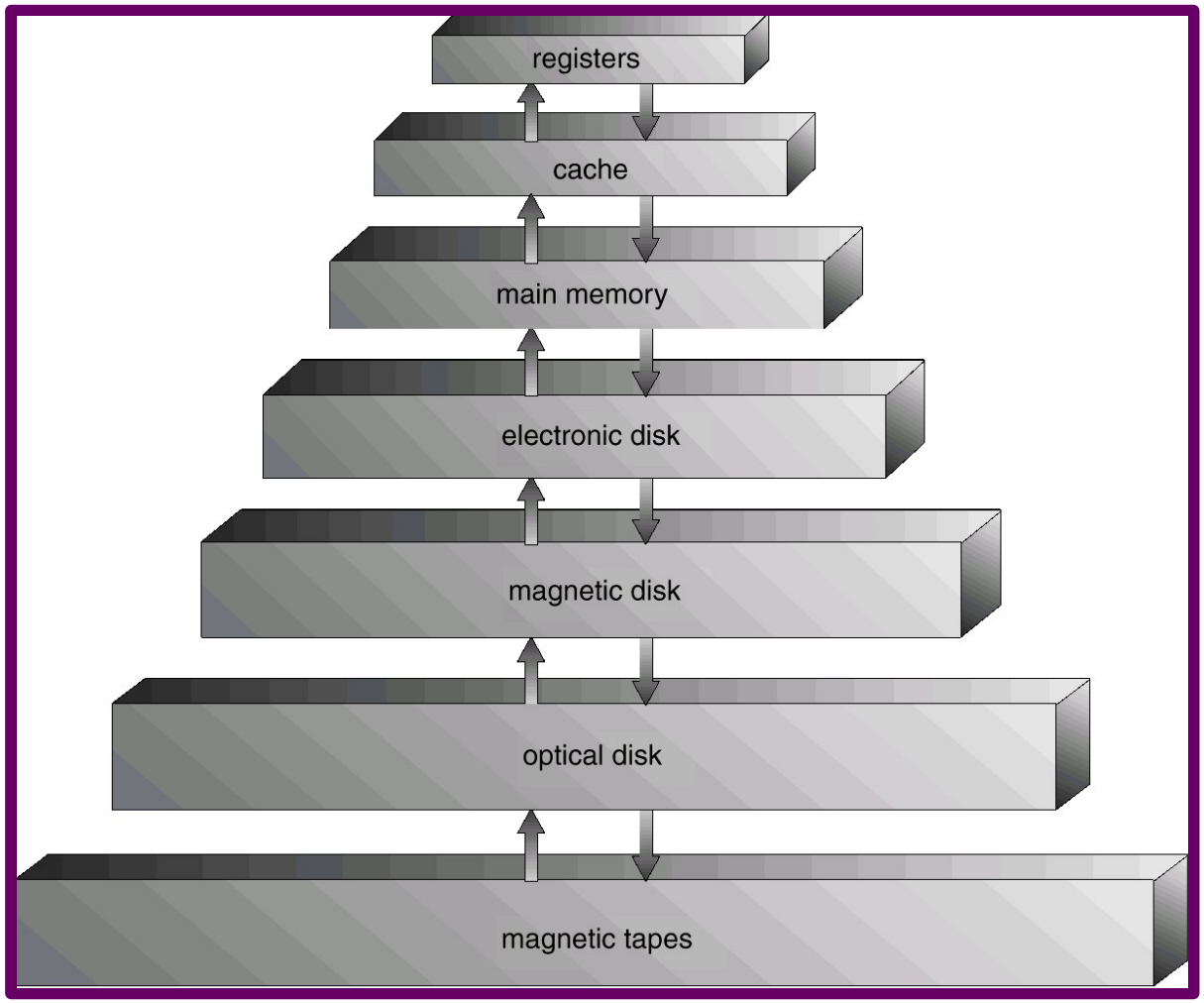


Storage Hierarchy

- 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

High
Cost
And fast

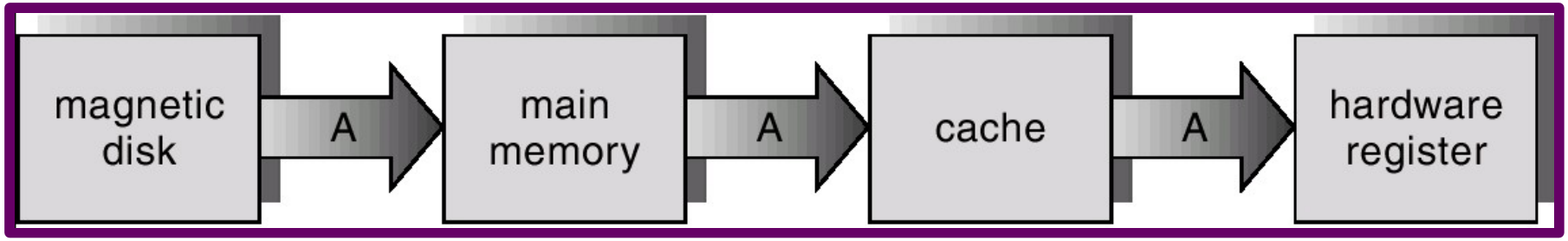


Less
Cost
and slow

Caching: Issue of Coherency and consistency

- Information is copied from secondary storage to cache memory on a temporary basis.
- Use of high-speed memory to hold recently-accessed data.
- Data transfer from cache to CPU is an hardware function: No OS is involved.
- Transfer of data from disk to OS is controlled by OS.
- **Cache coherency** is the discipline that ensures that changes in the values of shared operands are protected throughout the system in a timely fashion.
- 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*.
  - Suppose integer A located in file B is to be incremented by 1 and B resides on disk.
  - OS issues an I/O operation to copy disk block
  - Copy A to cache.
  - So the copy of A appears at several places.
  - In multi-programming environment, several programs may access A
  - In multi processor environment it is still more complex.

Migration of A From Disk to Register



Data transfer between cache to CPU is hardware function without OS interventio

Tranfer from disk to memory is usually controlled by OS.

Hardware Protection measures

- In Early systems, programmers had complete control over the system.
- A OSs developed the control was given to OS.
- OS started performing many functions such as I/O.
- OS started sharing resources among several programs
- Multiprogramming put several programs at same time.
- Without sharing the error may cause problem to only one program.
- With sharing an error may cause problems to many programs.
  - Sometimes to OS itself.
  - OSs have to be protected from such incorrect programs.
- MS-DOS and MAC-OS allow this kind of error.
- Protection measures.
  - 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.

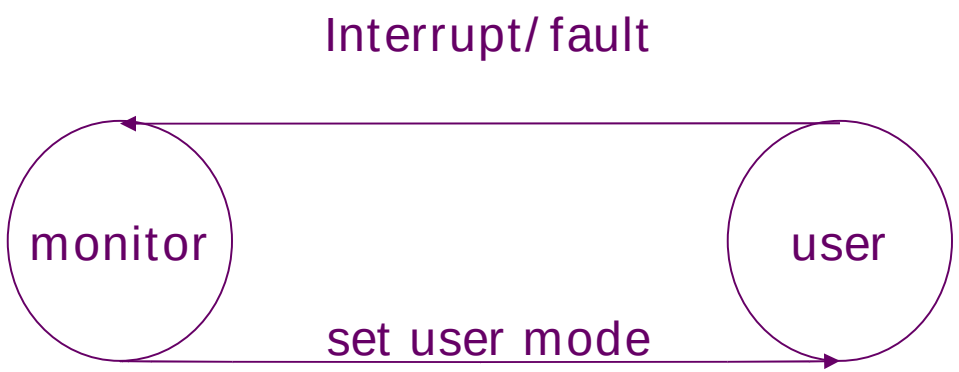
  This architectural enhancement is useful for many aspects of system operation.

  When system starts, hardware starts in monitor mode.

  OS is then loaded and OS starts user processes in user mode.

Dual- Mode Operation (Cont.)

- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1).
- Whenever OS takes control the mode bit is 0.
- When an interrupt or fault occurs hardware switches to monitor mode.
- This dual- mode operation protects computer from errant users and errant users from other.
- This protection can be achieved by designating some of the instructions as privileged instructions.
- ***Privileged instructions* can be issued only in monitor mode.**
  - **MS- DOS was written without mode bit**
  - **80846 provides duel mode operation: so it provides greater protection.**



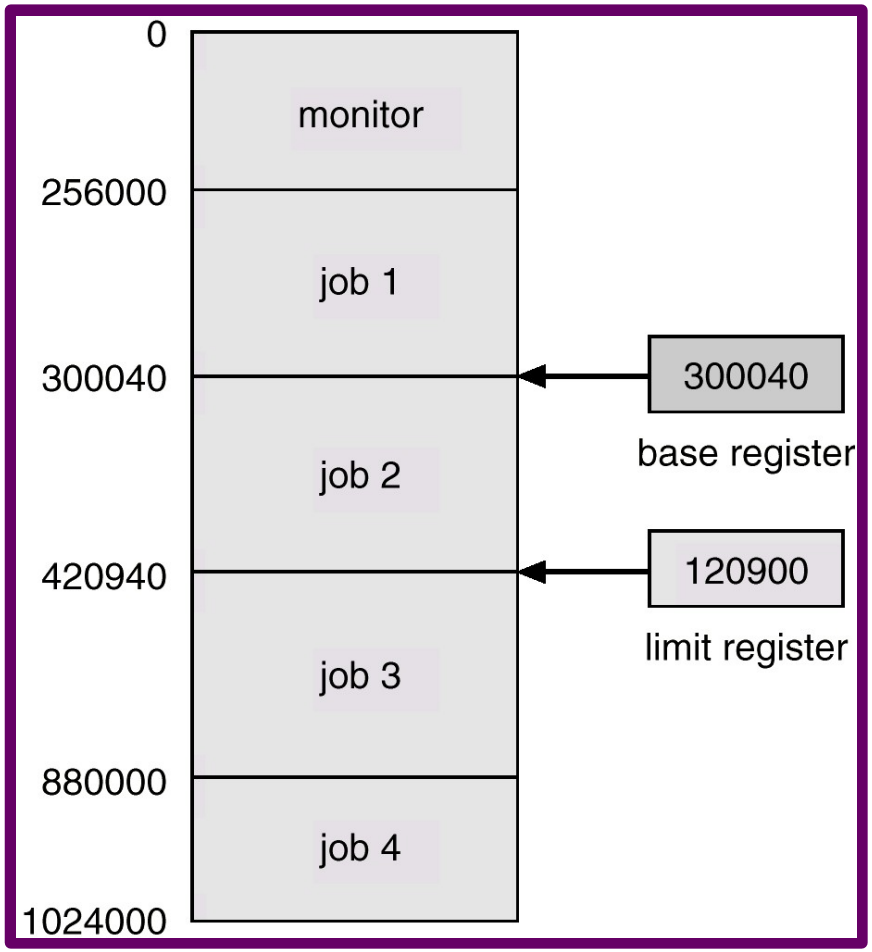
I/O Protection

- A user program may disrupt the normal operation
  - By issuing a illegal I/O operation.
- Solution: All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode
- A user program that, as part of its execution, stores a new address in the interrupt vector. How to protect it ?
- Answer: Consider I/O instructions as privileged instructions.

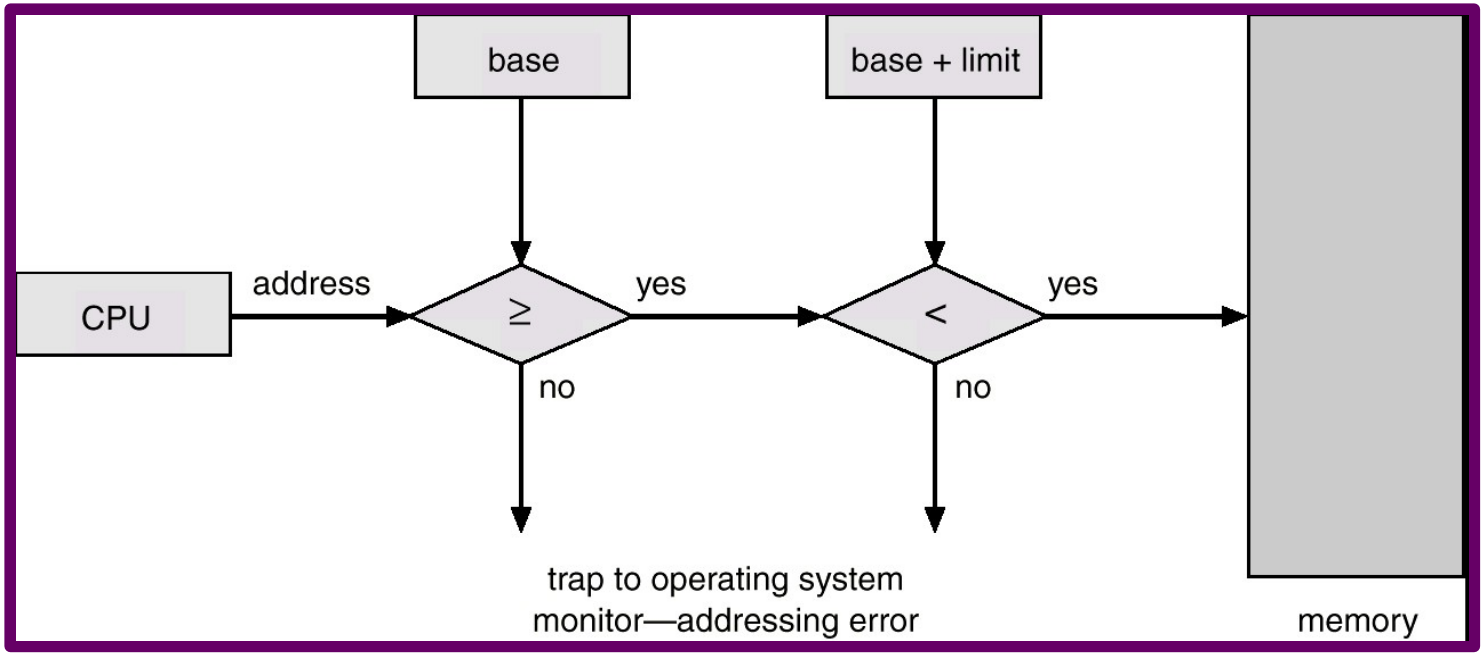
Memory Protection

- 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

- Ensures that OS maintains control.
- *Timer* – interrupts computer after a 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.
- Timer also used to compute the current time.
- Load-timer is a privileged instruction.

General system architecture

- To improve the utilization led to the development of multiprogramming and timesharing.
  - Resources of the computers are shared among many programs and processes
- Sharing led to allow OS to have control over I/O to provide continuous, consistent and correct operation.
- Dual mode of operation is introduced.
- I/O instructions and instructions to modify memory management are privileged instructions.
  - HLT instruction is privileged
  - The instructions to turn-on and turn-off are privileged.
- The instructions to change user mode to monitor mode are privileged.
- User must ask the monitor to do I/O. Such a request is called system call.
- When a system call is executed it is treated by the hardware as a software interrupt.
- Control is passed to interrupt vector to a service routine to the OS by setting the mode bit to monitor mode.
- The OS examines the request , and passes necessary information and checks correctness and executes the request.
- It returns the control to the statement after the system call.

General System Architecture...