Chapter 1. Introduction

- What is an Operating System?
- Mainframe Systems
- Desktop Systems
- Multiprocessor Systems
- Distributed Systems
- Clustered System
- Real -Time Systems
- Handheld Systems
- Feature Migration
- Computing Environments

Computer System Components

Four components:

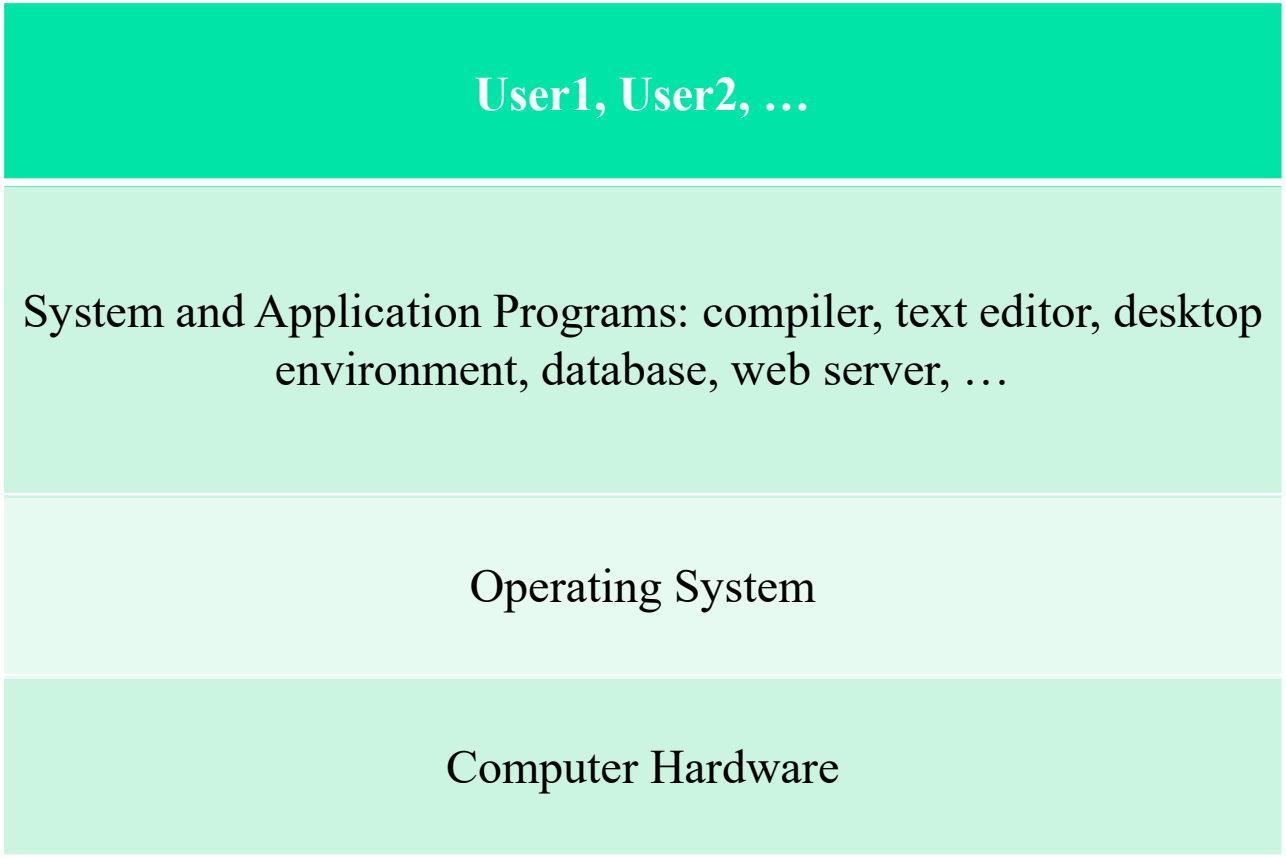
- **Hardware** – provides basic computing resources (CPU, memory, I/O devices)
- **Operating system** – controls and coordinates the use of the hardware among the various application programs for the various users
- **Applications programs** – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs)
- **Users** (people, machines, other computers)

What is an Operating System?

- ☺ *Definition of operating system?*
  - A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  1. Execute user programs.
  2. Make the computer system convenient to use.
  3. Use the computer hardware in an efficient manner.

View of System Components

☺ *4 components?*



Operating System Definitions

- Other definitions of operating system:
  1. Resource allocator – manages and allocates resources
  2. Control program – controls the execution of user programs and operations of I/O devices

- **Kernel** – the one program running at all times (all else being application programs) in the **kernel mode**
- ☺ What is the kernel mode? See Figure 1.10.

Types of Computer Systems

- Mainframe Systems
- Desktop Systems
- Multiprocessor Systems
- Distributed Systems
- Clustered System
- Real -Time Systems
- Handheld Systems

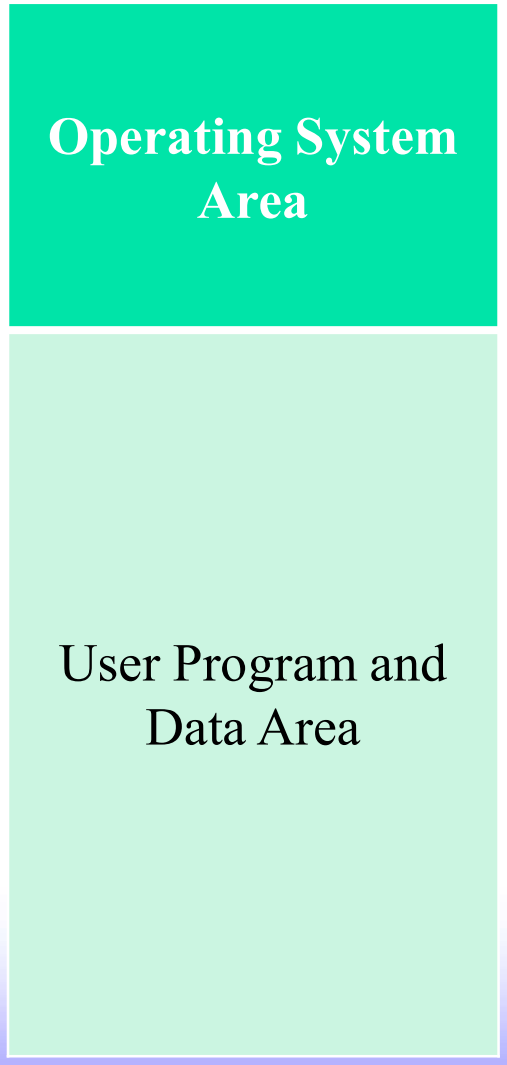
Mainframe Systems

- **Batch** systems
  - Punch cards – program, data, control information about the job
  - Reduce setup time by batching similar jobs
    - Card reader; magnetic tape; magnetic drum; hard disk
  - Automatic job sequencing – automatically transfers control from one job to another.
  - First rudimentary operating system

Mainframe Systems - continued

- The OS was called **resident monitor**
  - Initial control in monitor
  - Control transfers to job
  - When job completes, control transfers back to monitor
  - ☺ ***What is the meaning of the control?***

Simple Batch Systems



The speed of CPU >> the speed of I/O devices
The user program might print the result.

=> During the printing, CPU isn't at work.

=> Inefficient

=> ☺ *Then,* ***how to improve?***

Multiprogrammed Batch Systems

Several jobs are kept in the main memory at the same time, and the CPU is multiplexed among them.

=> Efficient

But, one job after one job, not concurrently yet.

=> Non-interactive
=> ☺ *Then,* ***how to improve?***

Features Needed for Multiprogramming

- **Job scheduling** – for multiple jobs on disk
- **Memory management** – The system must allocate the main memory to several jobs.
- **CPU scheduling** – The system must choose a job among several jobs ready to run.
- **I/O management**
  - I/O routine supplied by the system
  - Allocation of i/o devices

Time-Sharing Systems

- But interactive computing requires less response time,
- E.g., CLI and GUI on terminal, with keyboard [and mouse]
- Multiprogrammed batch systems are not appropriate.
- ☺ *Then, how to improve?*
- **Time-sharing**
  - The CPU is multiplexed among several jobs that are kept in memory and on disk. (The CPU is allocated to a job only if the job is in memory.)
  - That is, **the CPU is allocated to one job for a very short time and then switched to other job.**

    => Illusion of concurrent processing of several jobs
  - A job might be **swapped** in and out of memory to the disk.
  - ☺ ***How to implement?***
  - Using timer and interrupt
- On-line communication between the user and the system is provided.
  - When the operating system finishes the execution of one command, it seeks the next "control statement" from the user's keyboard.
- On-line system must be available for users to access data and code.

Desktop Systems

- *Personal computers* – computer system dedicated to a single user
- I/O devices – keyboards, **mice**, display screens, small printers
- **User convenience and responsiveness are important.**
- Can adopt technology developed for larger operating system.
  - Often individuals have sole use of computer and do not need advanced CPU utilization of protection features.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux).

- See Figure 1.2.

Parallel Systems

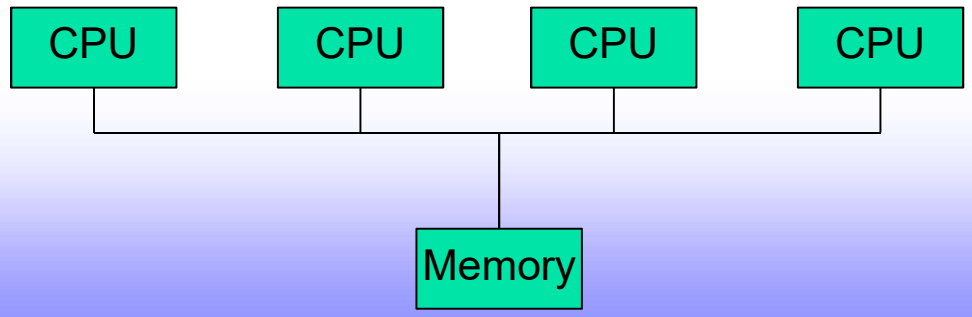
- Systems with more than one CPU in close communication.
  - Also known as *multiprocessor systems*.
- ☺ *How about multi-core CPUs?*

- ***Tightly coupled system***
  - Processors share memory and a system clock.
  - Communication usually takes place through the shared memory.
- Advantages of **parallel** system:
  - Increased *throughput*
  - Economical
  - Increased reliability (in some cases)
    - graceful degradation
    - fail-soft systems

Parallel Systems - continued

- **Parallel processing**, or called **multiprocessing**
  - One job could run over several processors to speed up.

- *Asymmetric multiprocessing*
  - Each processor is assigned a specific task; master processor schedules and allocated work to slave processors.
  - More common in extremely large systems
- ***Symmetric multiprocessing (SMP)***
  - Each processor runs and identical copy of the operating system.
  - Many processes can run at once without performance deterioration.
  - Most modern operating systems support SMP.



Distributed Systems

- Distribute the computation among several physical processors.
- ***Loosely coupled system***
  - Each processor has its own local memory.
  - Processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- **Distributed computing**
  - Sub-jobs for a job
  - Each sub-job running on a system
- Advantages of distributed systems
  - Resources sharing
  - Computation speed up – load sharing
  - Reliability

Distributed Systems - continued

- Requires networking infrastructure.
- Local area networks (*LAN*) or Wide area networks (*WAN*)
- May be either *client-server* or *peer-to-peer* systems

Clustered Systems

- Clustering allows two or more computer systems to share storage, over a network.
- Provides high reliability.
- **Parallel processing over a LAN**

- *Asymmetric clustering*: one server runs the application or applications while other servers standby.
- ***Symmetric clustering***: all *N* hosts are running the application or applications.

Real-Time Systems

- Often used as a **control device in a dedicated application** such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints
- **Very fast response time**
- Real-Time systems may be either *hard* or *soft* real-time.

Real-Time Systems - continued

- Hard real-time:
  - Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems
- Soft real-time
  - Using **priority** of processes
  - Limited utility in industrial control of robotics
  - Integrate-able with time-share systems
  - Useful in applications (multimedia, virtual reality) requiring tight response times

Handheld Systems

- Personal Digital Assistants (PDAs)
  - Tablet computers
- Cellular telephones
- Smart phones
- Issues:
  - Limited memory
  - Slow processors
  - Small display screens
  - Short-lived battery

Computing Environments

- Traditional computing
  - PCs, Servers, limited remote access
- Client-server computing
  - Client-server and web services, convenient remote access, location-less servers
  - See Figure 1.13.
- Peer-to-peer (P2P) computing
- **Embedded** computing
  - Most computers (auto engine controllers, microwaves)
  - Very limited operating system features
  - Little or no user interface, remote access

Review: Computer-System Structures

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

A Modern Computer System

- See Figure 1.2.

Computer-System Operation

- I/O devices
  - 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.
  - ☺ ***What is a device driver?***
  - CPU and DMA move data from/to main memory to/from local buffers, according to instructions in main memory.
  - I/O is from/to the device to/from local buffer of controller.
  - Device controller informs CPU that it has finished its operation by causing an *interrupt*.

Computer-System Operation – cont.

- Bootstrap
  - When a computer starts
  - An hardware interrupt
    - Electric signal to CPU
    - PC <- the address of the first instruction in bootstrap
  - Stored in ROM
  - Role
    - Initialize I/O devices, CPU registers, memory
    - Load the operating system kernel
    - Transfer CPU to the operating system kernel

- Then the kernel starts the first process and waits for some event to occur

Computer-System Operation – cont.

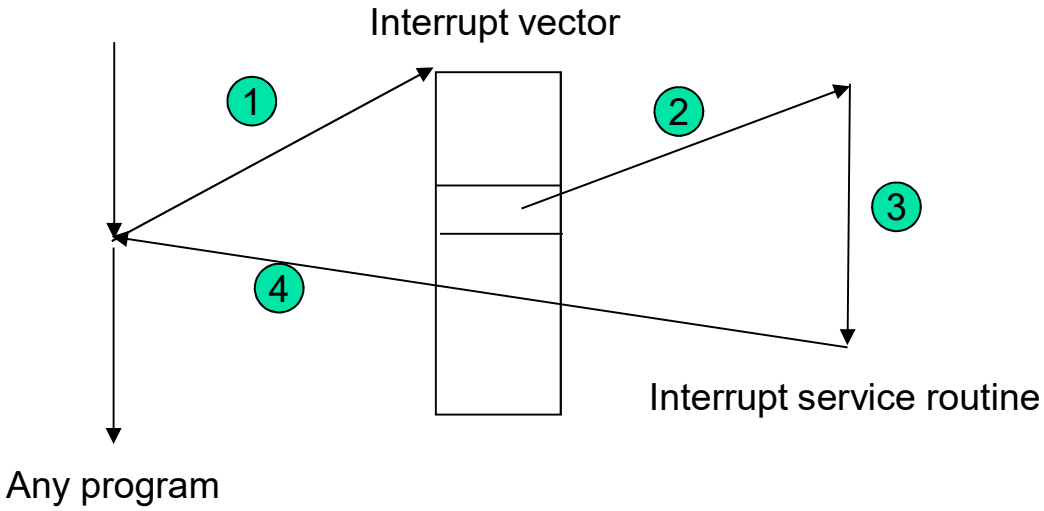
- Interrupt
  - When an **event** occurs, while the OS kernel or an application program is running.
  - Called **interrupt**
  - Interrupt transfers CPU control to the ***interrupt service routine*** generally.
  - Types
    - **Hardware interrupts**
      - Example:
        - A character typed at a keyboard.
        - Completion of an I/O operation: device is ready to do more jobs.
        - Timer: to make sure the OS eventually gets control in an unusual situation.
    - **Software interrupts**
      - Called **trap**
      - Software-generated interrupt caused either by an error or a user request:
        - **System call**
        - **Error (illegal instruction, addressing violation, division of zero, …)**
  - An operating system is ***interrupt driven.***

Computer-System Operation – cont.

- Interrupt handling
  - Electric signal to CPU
  - Suspend execution of current program
    - Save the **context** into the program's PCB (Process Control Block) that is mainted by OS - all the registers, especially PC.
  - Jump to **interrupt handler routine (IHR)** ☺ *What does jumping mean?*
    - Save PC (Program Counter).
    - Set PC to the first address of IHR, which is fixed.
  - Process interrupt
    - Check the interrupt type and process.
    - ***Polling***
    - ***Vectored* interrupt system**
      - PC <- the specific address depends on the type of interrupt
        - Interrupt vector that contains the addresses of all the service routines
        - Sort of indexed and based addressing
        - E.g., PC <- 0xffff + 2 for the interrupt type 2
  - Resume interrupted program
    - Restore the context back to registers,
    - especially PC ☺ *Why is this important ???*

Computer-System Operation – cont.

- Interrupt handling



I/O Structure

- ***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.
  - User program can be informed I/O completion using interrupt.

I/O Structure - continued

- **DMA (Direct Memory Access)** transfer
  - 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.

  - ☺ ***Synchronous or asynchronous I/O?***

- See Figure 1.5, "How a modern computer system works".

Storage Structure

- ***Instruction register***
  - Temporary memory in CPU for the execution of instructions
- ***Cache memory***
  - Temporary memory to overcome the speed difference between CPU and main memory
- ***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.

Storage Hierarchy

- Storage systems organized in hierarchy:
  - Speed
  - Cost
  - Volatility
- See Figure 1.4.

- ***Caching*** – copying information into faster storage system; main memory can be viewed as a *cache* for secondary storage.
  - 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.*
  - Hard disk -> main memory -> cache memory -> CPU registers

Hardware Protection

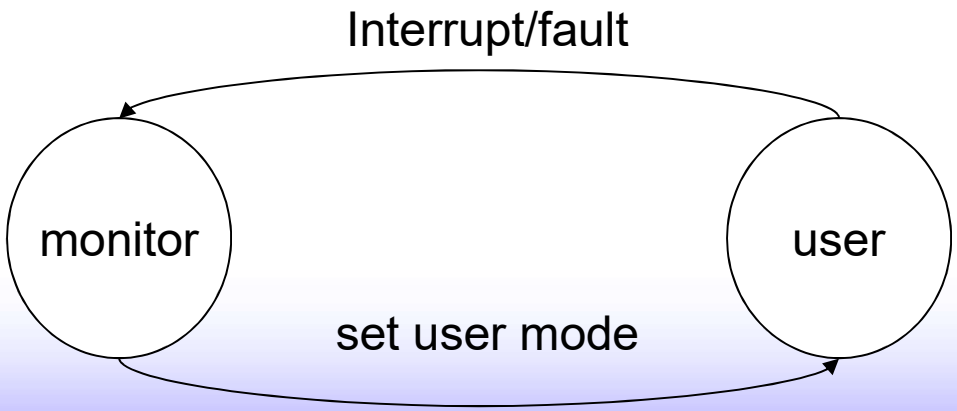
- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

Hardware Protection - continued

- **Dual mode operation**
  - See Figure 1.10.
  - Sharing system resources requires operating system to ensure that an incorrect program or poorly behaving human cannot cause other programs to execute incorrectly.
  - OS must 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

Hardware Protection - continued

- **Dual mode operation**
  - ***Mode bit*** added to computer hardware to indicate the current mode: monitor (0) or user (1).
  - ***Privileged instructions*** can be issued only in monitor mode.
  - ☺ *How?*
  - When an interrupt or fault occurs hardware switches to monitor mode, i.e., sets the mode bit by 0.



Hardware Protection - continued

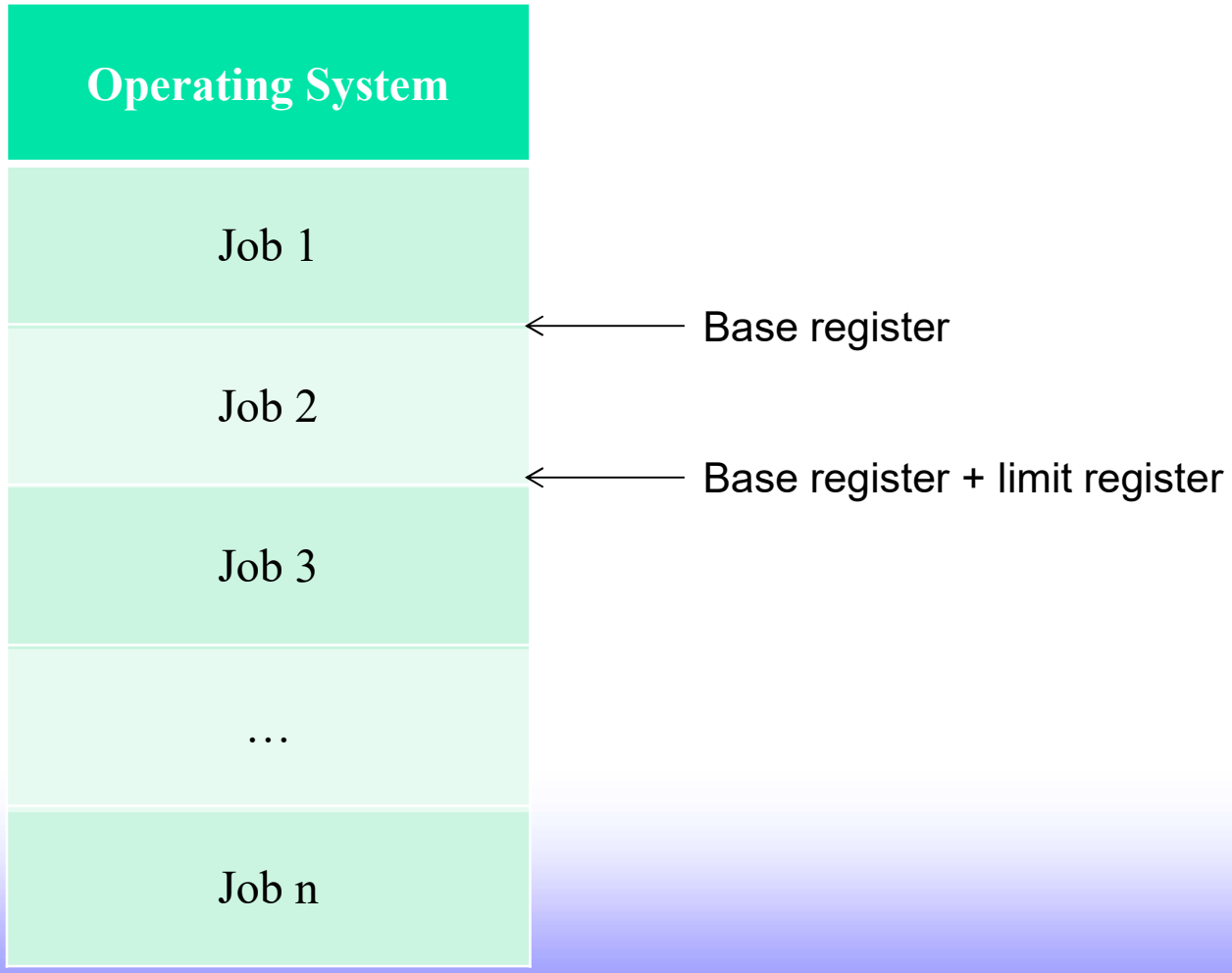
- **I/O protection**
  - All I/O instructions are privileged instructions.
  - Must ensure that a user program could never gain control of the computer in monitor mode, i.e., a user program that, as part of its execution, stores a new address in the interrupt vector.
  - ☺ *Given the I/O instructions are privileged, how does the user program perform I/O?*
  - **System call** – the method used by a process to request action by the operating system.
    - Usually takes the form of a **trap** (i.e., software interrupt) to a specific location in the interrupt vector.
    - Control passes through the interrupt vector to a service routine in the OS, and the mode bit is set to monitor mode.
    - ☺ ***By what ???***
    - The monitor verifies that the parameters are correct and legal, executes the request, and returns control to the instruction following the system call,
    - After setting the mode bit by 1,
    - ☺ ***By what ???***

Hardware Protection - continued

- **Memory protection**
  - Must provide memory protection at least for the interrupt vector and the interrupt service routines.
  - In order to have memory protection, at a minimum add two registers that determine the range of legal addresses a program may access:
    - **Base register** – holds the smallest legal physical memory address for a job.
    - **Limit register** – contains the size of the range.
  - The load instructions for the *base* and *limit* registers are privileged instructions.
  - Memory outside the defined range is protected -> addressing violation, e.g., wrong index in an array variable.

Hardware Protection - continued

- **Memory protection**



Hardware Protection - continued

- **Memory protection**
  - When an instruction is executed,
    - CPU generates an address.
    - If the address < base
      - Then trap to operating system -> addressing error
    - If the address >= base + limit
      - Then trap to operating system -> addressing error
    - Access to memory with the address

Hardware Protection - continued

- **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.