Operating System Structure

- **Multiprogramming** is needed for efficiency
  - A single user cannot keep CPU and I/O devices busy at all times!
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - one job is selected and run via job scheduling
  - When it has to wait (for I/O for example), OS switches to another job

Operating System Structure

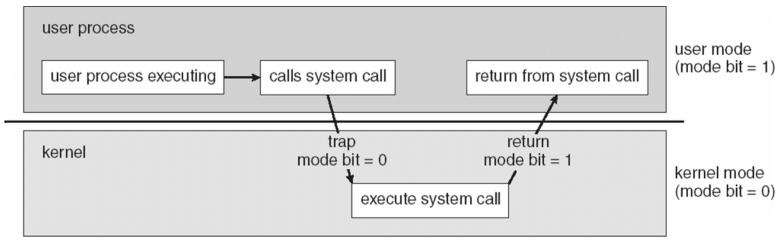
- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
  - Response time should be < 1 second
  - If each user has at least one program executing in memory ⇨ process
  - If several jobs are ready to run at the same time ⇨ CPU scheduling
  - If processes don't fit in memory, swapping moves them in and out to run
  - ***Virtual memory*** allows execution of processes not completely in memory

Hardware Protection – Dual Mode

- Sharing system resources requires OS to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- **Dual-Mode operation**: Provide hardware support to differentiate between at least two modes of operations:
  1. *User mode:* execution done on behalf of the user
  2. *Monitor mode* (also *kernel mode* or *system mode*): execution done on behalf of the OS
  - Instruction set is restricted in user mode
  - A program, running in user mode, attempting to execute a privileged instruction will cause a trap

System Calls

- **System calls** are used to request services from the OS
  - E.g. give me the current date/time, open a file for reading, etc.
- Executing a system call changes mode to kernel, return from call resets it to user

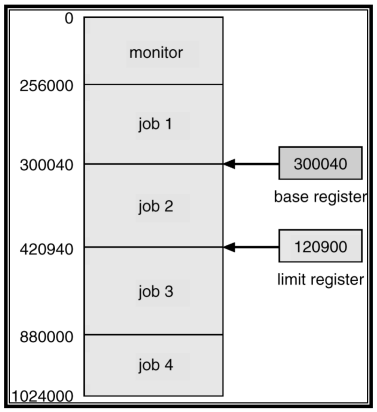


Hardware Protection – I/O & Memory Protection

- We must ensure that I/O devices are protected as well, to have *I/O protection* we ensure:
  - all I/O instructions are privileged instructions
- We must also ensure that processes (jobs) are not able to access each other's memory space
  - User jobs must also not be able to access the interrupt handlers or interrupt vectors

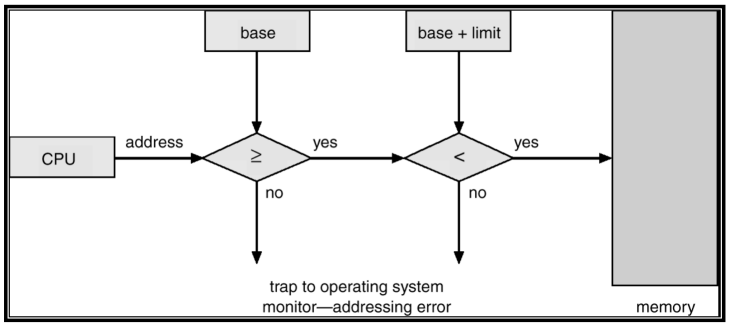
Hardware Protection – Memory Protection

- 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 physical memory address
  - limit register: holds the size of the job.



Hardware Protection – Memory Protection

- Checking memory addresses in hardware:



Hardware Protection – CPU Protection

- We must also ensure that no one job is using up all the CPU cycles.
- *Timer* – interrupts computer after specified period to ensure operating system maintains control.
  - timer is decremented every clock tick
  - when timer reaches 0, interrupt occurs
- The timer is commonly used to implement *time sharing*.