# Operating System
# Lec-02
# Process Abstraction

Tahira Alam

University of Asia Pacific

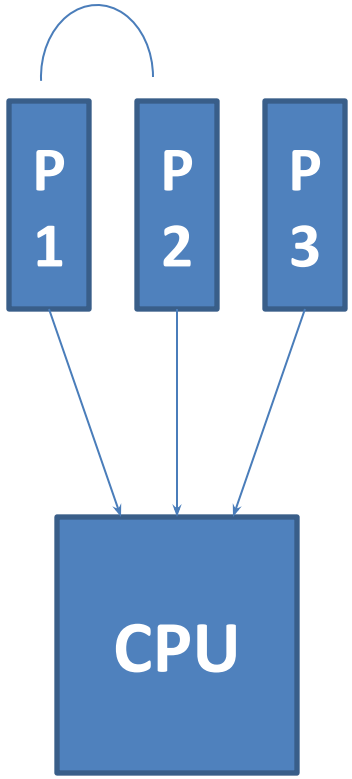
# OS manages CPU

- OS provides the process abstraction

– Process: a running program

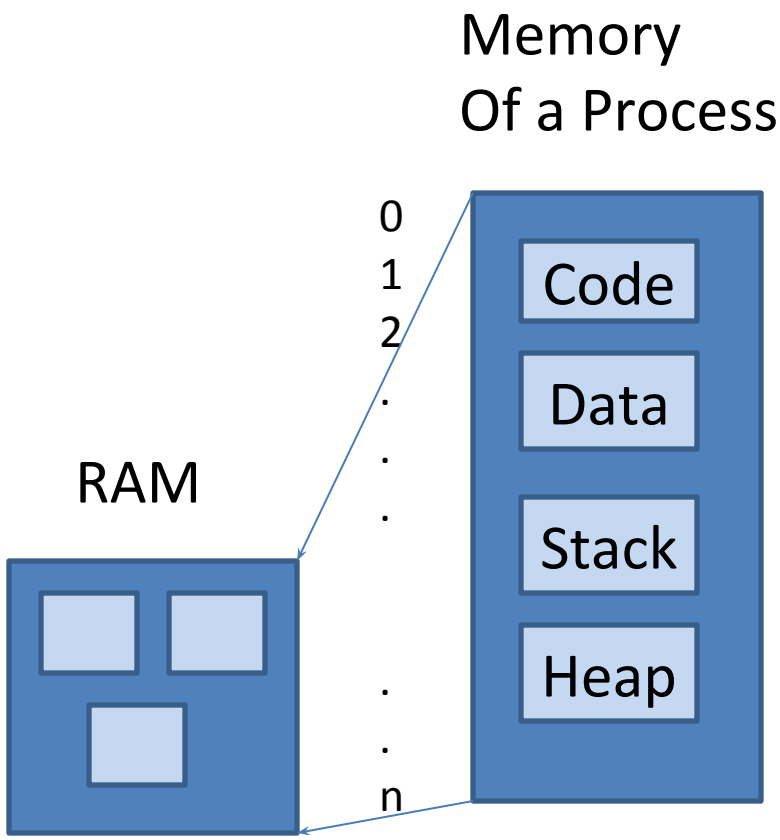
– OS creates and manages processes

- Each process has the illusion of having the complete CPU, i.e., OS virtualizes CPU
- Timeshares CPU between processes
- Enables coordination between processes



# OS manages memory

- OS manages the memory of the process: code, data, stack, heap etc
- Each process thinks it has a dedicated memory space for itself, numbers code and data starting from 0 (virtual addresses)
- OS abstracts out the details of the actual placement in memory, translates from virtual addresses to actual physical addresses

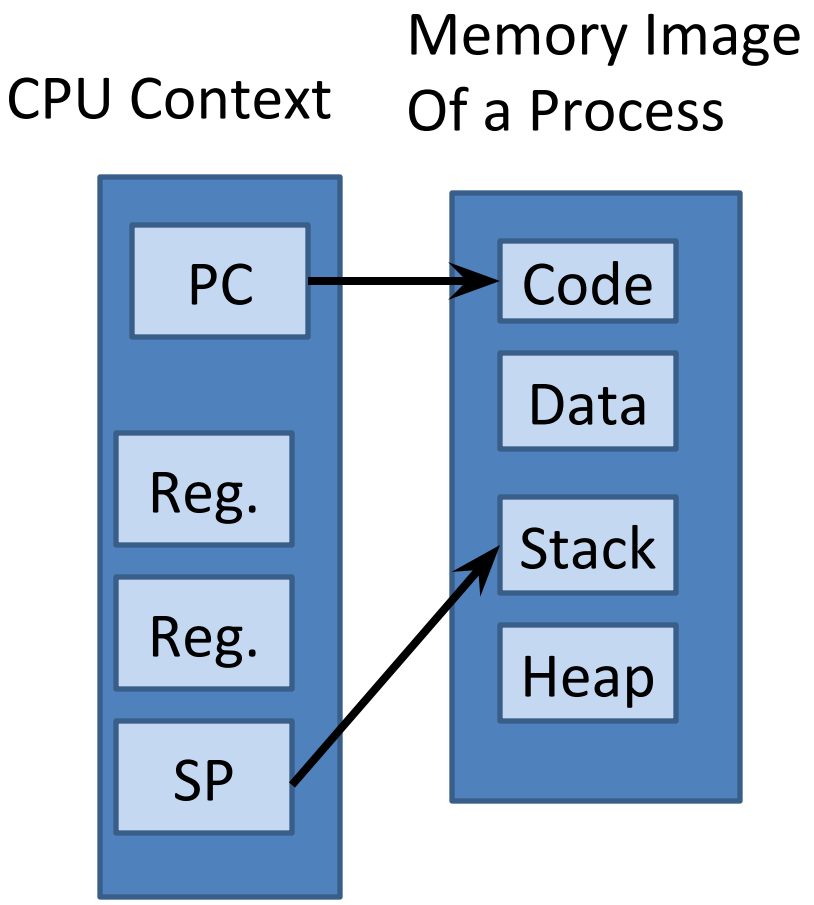


# OS provides process abstraction

- When you run an exe file, the OS creates a process = a running program
- OS timeshares CPU across multiple processes: virtualizes CPU
- OS has a CPU scheduler that picks one of the many active processes to execute on a CPU
  - **Policy:** which process to run
  - **Mechanism:** how to “context switch” between processes

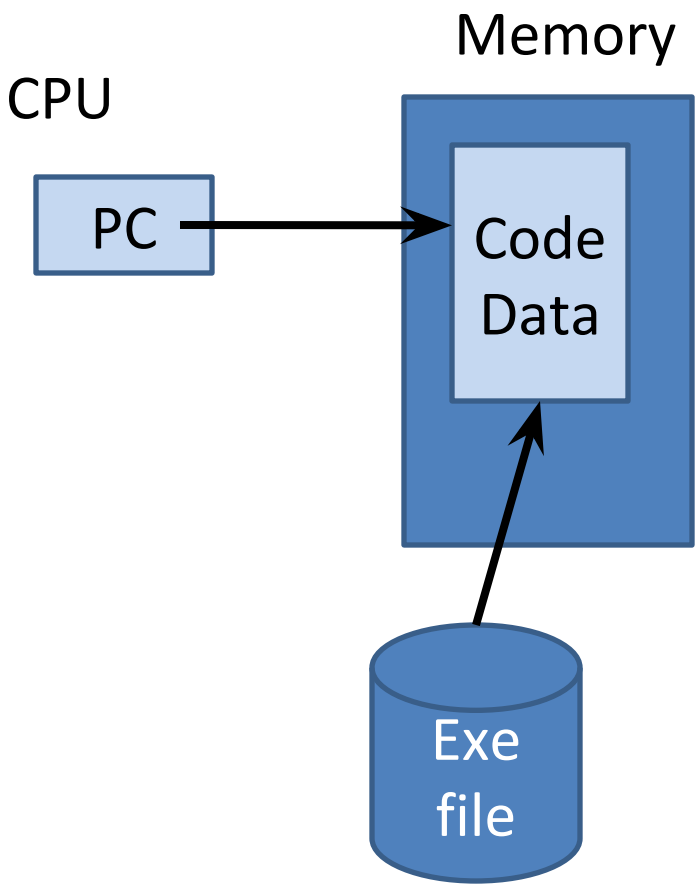
# What constitutes a process?

- A unique identifier (PID)
- Memory image
  - Code & data (static)
  - Stack and heap (dynamic)
- CPU context: registers
  - Program counter
  - Current operands
  - Stack pointer
- File descriptors
  - Pointers to open files and devices



# How does OS create a process?

- Allocates memory and creates memory image
  – Loads code, data from disk exe
  – Creates runtime stack, heap
- Opens basic files
  – STD IN, OUT, ERR
- Initializes CPU registers
  – PC points to first instruction



# States of a process(1)

- **Running:** currently executing on CPU
- **Ready:** waiting to be scheduled
- **Blocked:** suspended, not ready to run

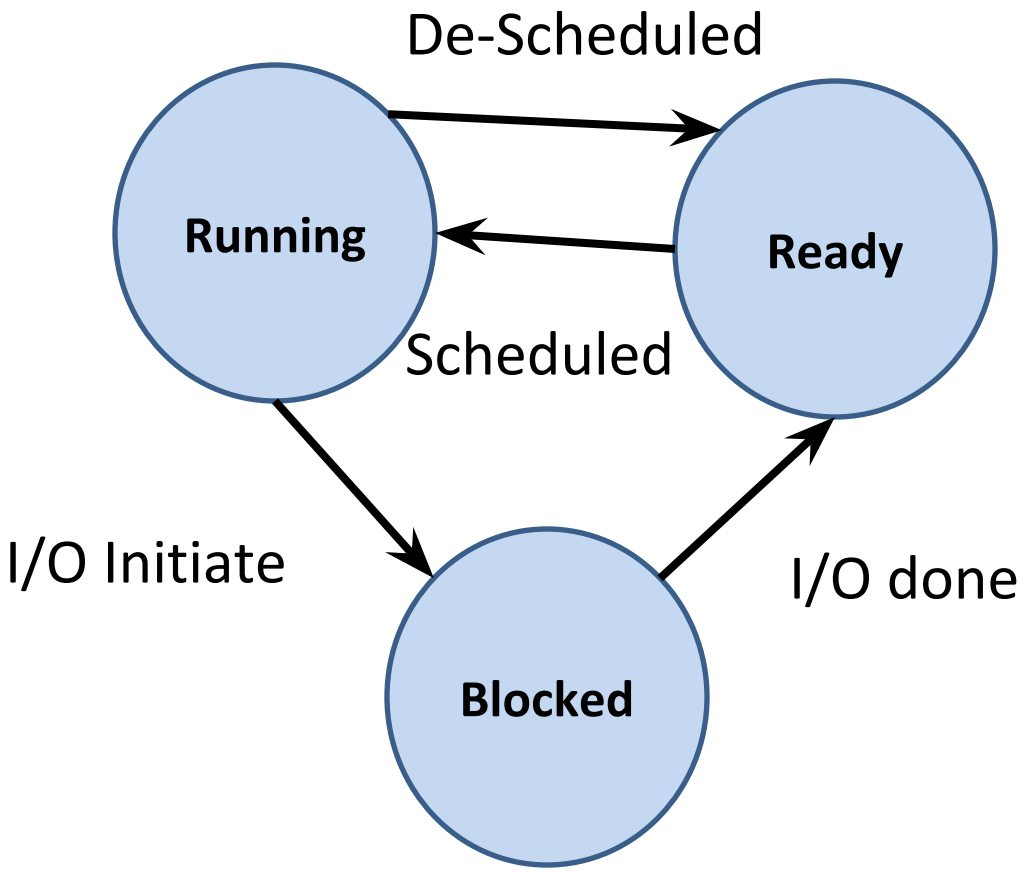
– Why? Waiting for some event, e.g., process issues a read from disk

– When is it unblocked? Disk issues an interrupt when data is ready

# States of a process(2)

- **New:** being created, yet to run
- **Dead:** terminated

# Process State Transitions



# Example: Process States

| Time | $Process_0$ | $Process_1$ | Notes |
|---|---|---|---|
| 1 | Running | Ready | |
| 2 | Running | Ready | |
| 3 | Running | Ready | $Process_0$ initiates I/O |
| 4 | Blocked | Running | $Process_0$ is blocked, |
| 5 | Blocked | Running | so $Process_1$ runs |
| 6 | Blocked | Running | |
| 7 | Ready | Running | I/O done |
| 8 | Ready | Running | $Process_1$ now done |
| 9 | Running | – | |
| 10 | Running | – | $Process_0$ now done |

# OS data structures

- OS maintains a data structure (e.g., list) of all active processes
- Information about each process is stored in a process control block (PCB)

– Process identifier
– Process state
– Pointers to other related processes (parent)
– CPU context of the process (saved when the process is suspended)
– Pointers to memory locations
– Pointers to open files

# Multiprocessors

- **Multiprocessing** is the use of two or more central processing units within a single computer system.

- Symmetric Multiprocessing
- Asymmetric Multiprocessing

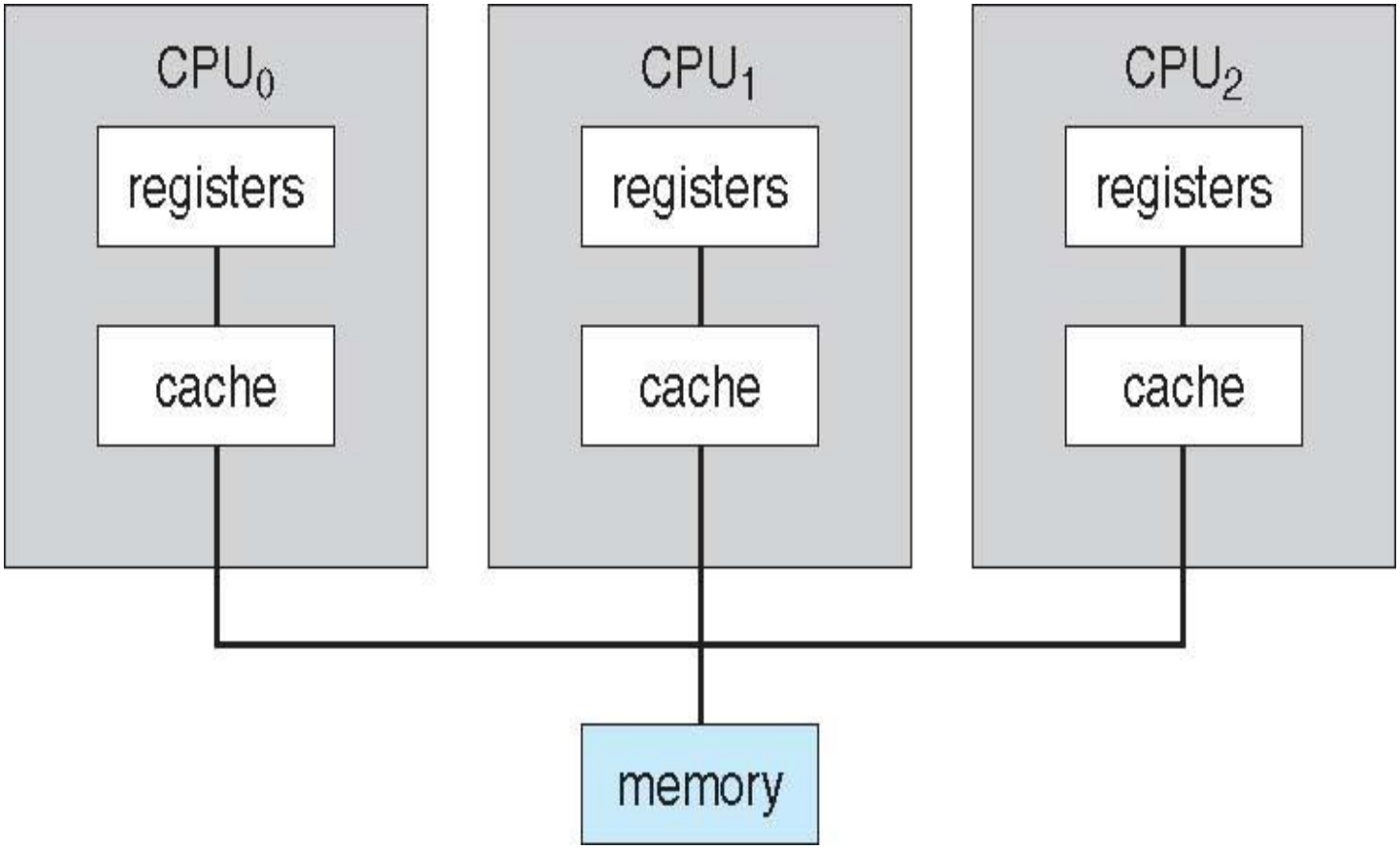
# Asymmetric Multiprocessing

- not all of the multiple interconnected central processing units (CPUs) are treated equally.
- only a master processor runs the tasks of the operating system

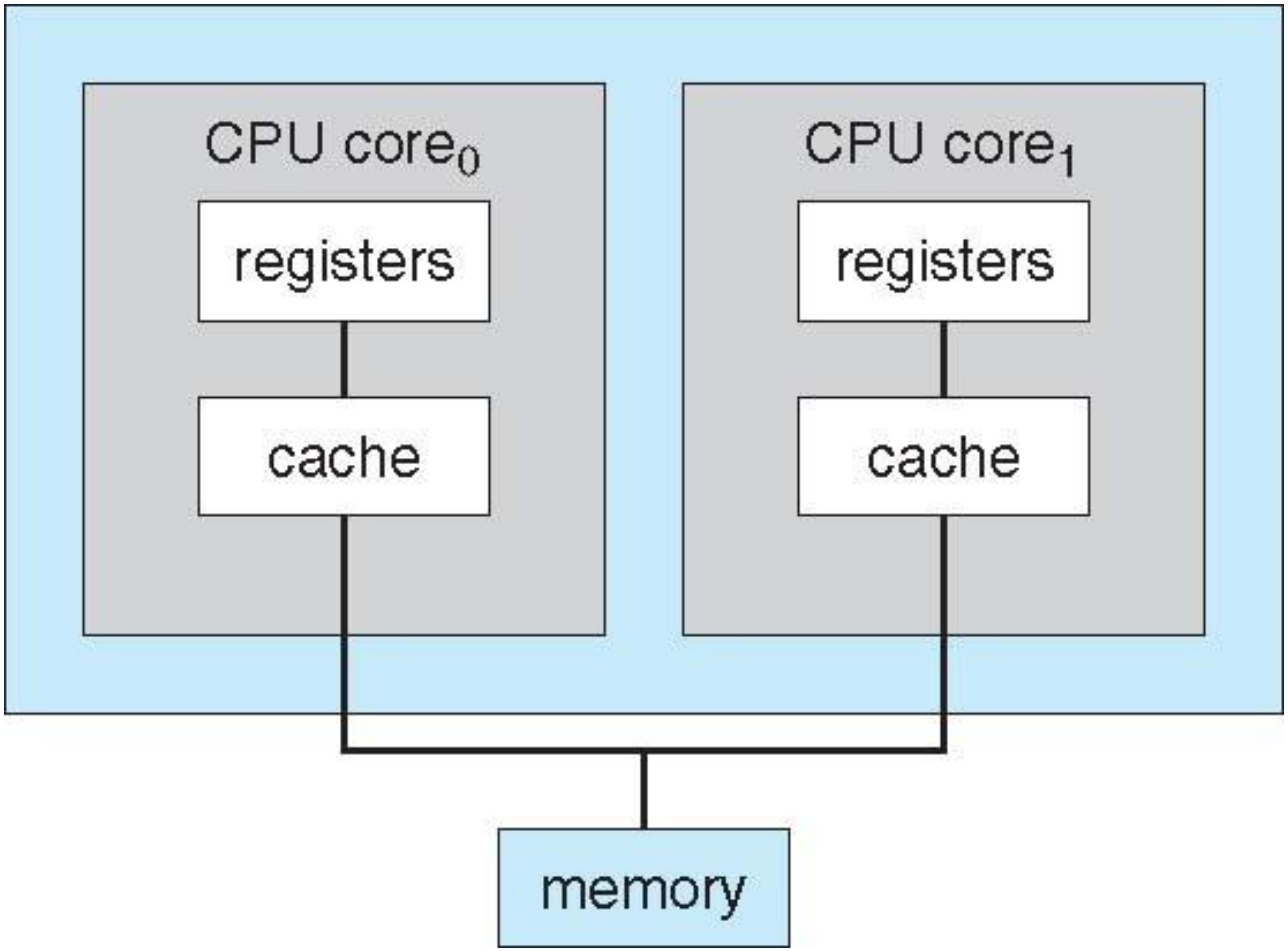
# Symmetric Multiprocessing(1)

- two or more identical processors are connected to a single, shared main memory, have full access to all input and output devices

# Symmetric Multiprocessing(2)



# Symmetric Multiprocessing(3)



# Cluster Systems

- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)**
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** has multiple nodes running applications, monitoring each other

# Computing Environments(1)

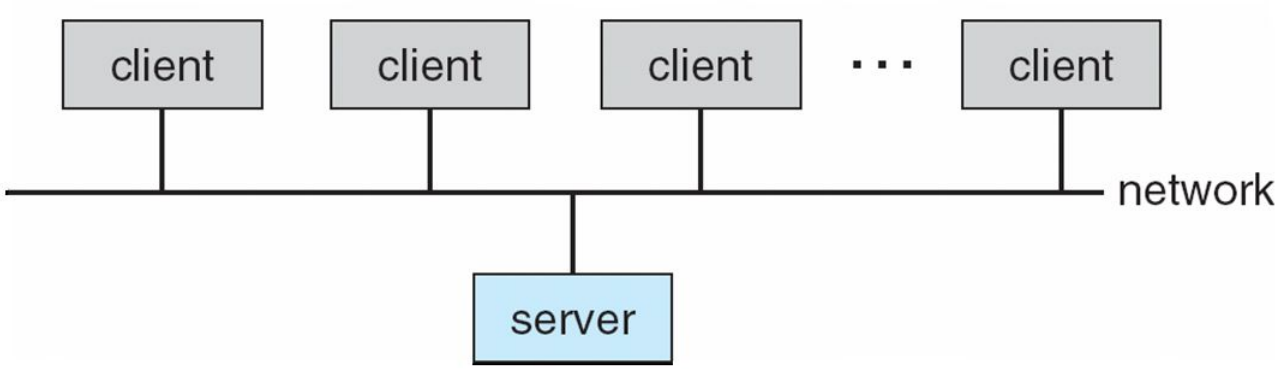
- Traditional computer
  - Blurring over time
  - Office environment
    - PCs connected to a network, terminals attached to mainframe or minicomputers providing batch and timesharing
    - Now portals allowing networked and remote systems access to same resources
  - Home networks
    - Used to be single system, then modems
    - Now firewalled, networked

# Computing Environments(2)

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server** provides an interface to client to request services (i.e., database)
    - **File-server** provides interface for clients to store and retrieve files

# Computing Environments(3)

- Client-Server Computing



# Computing Environments(4)

- Peer to Peer
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network

# Computing Environments(4)

- Web based Computing
- Web has become ubiquitous
- PCs most prevalent devices
- More devices becoming networked to allow web access
- New category of devices to manage web traffic among similar servers: **load balancers**

# Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS.
- **Security** – defense of the system against internal and external attacks.

Thank You!