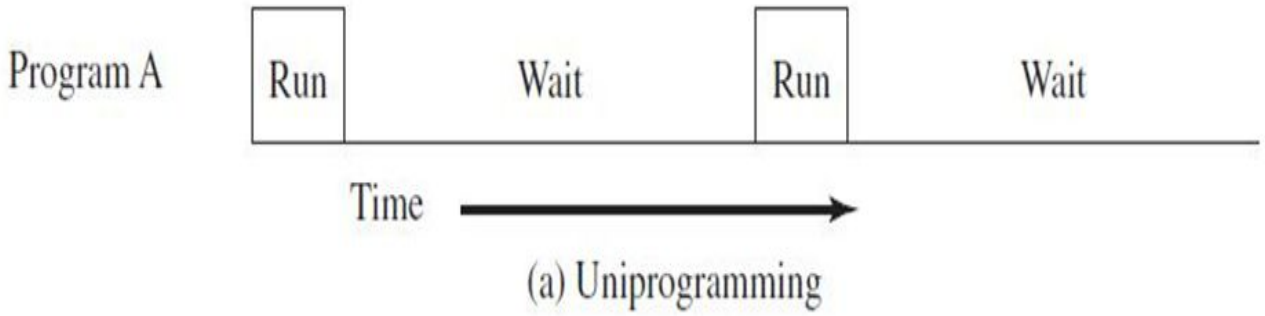
# Operating Systems
# Lecture 3
## Multi Program Batch Operating Systems

***Ms. Minal Rajwar***

# Introduction to Multiprogramming

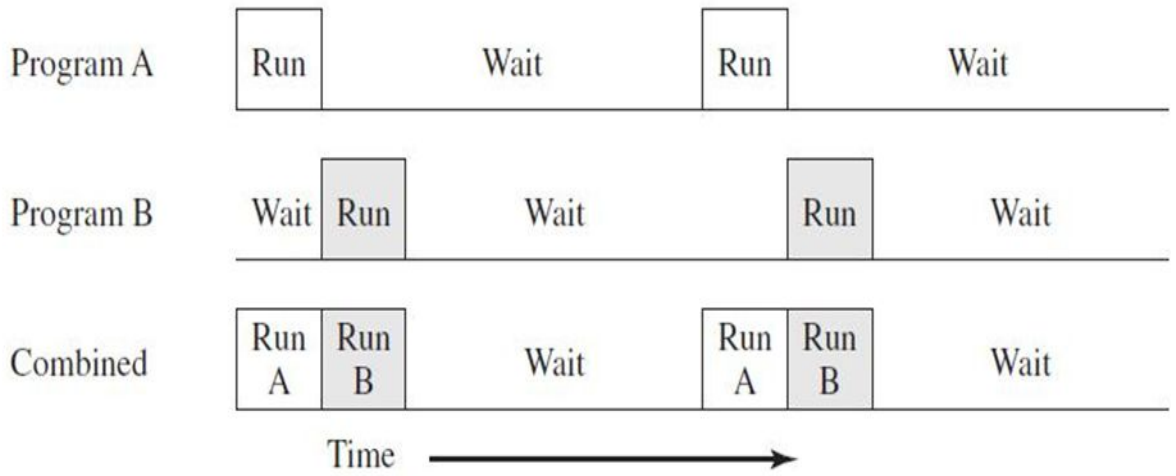
- Even with the automatic job sequencing provided by a simple batch operating system, the processor is often idle. The problem is that I/O devices are slow compared to the processor. Figure below illustrates a situation where we have a single program, referred to as uniprogramming. The processor spends a certain amount of time executing, until it reaches an I/O instruction. It must then wait until that I/O instruction concludes before proceeding.



(a) Uniprogramming

# Introduction to Multiprogramming

- Suppose that there is enough memory for OS and two user programs. When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O. This is shown below




(b) Multiprogramming with two programs

# Advantages for Multiprogramming

- Early batch systems left CPU **idle** during I/O.
- Multiprogramming emerged to **overlap I/O of one job with CPU of another.**
- **Objective:** Maximize CPU utilization, minimize idle time

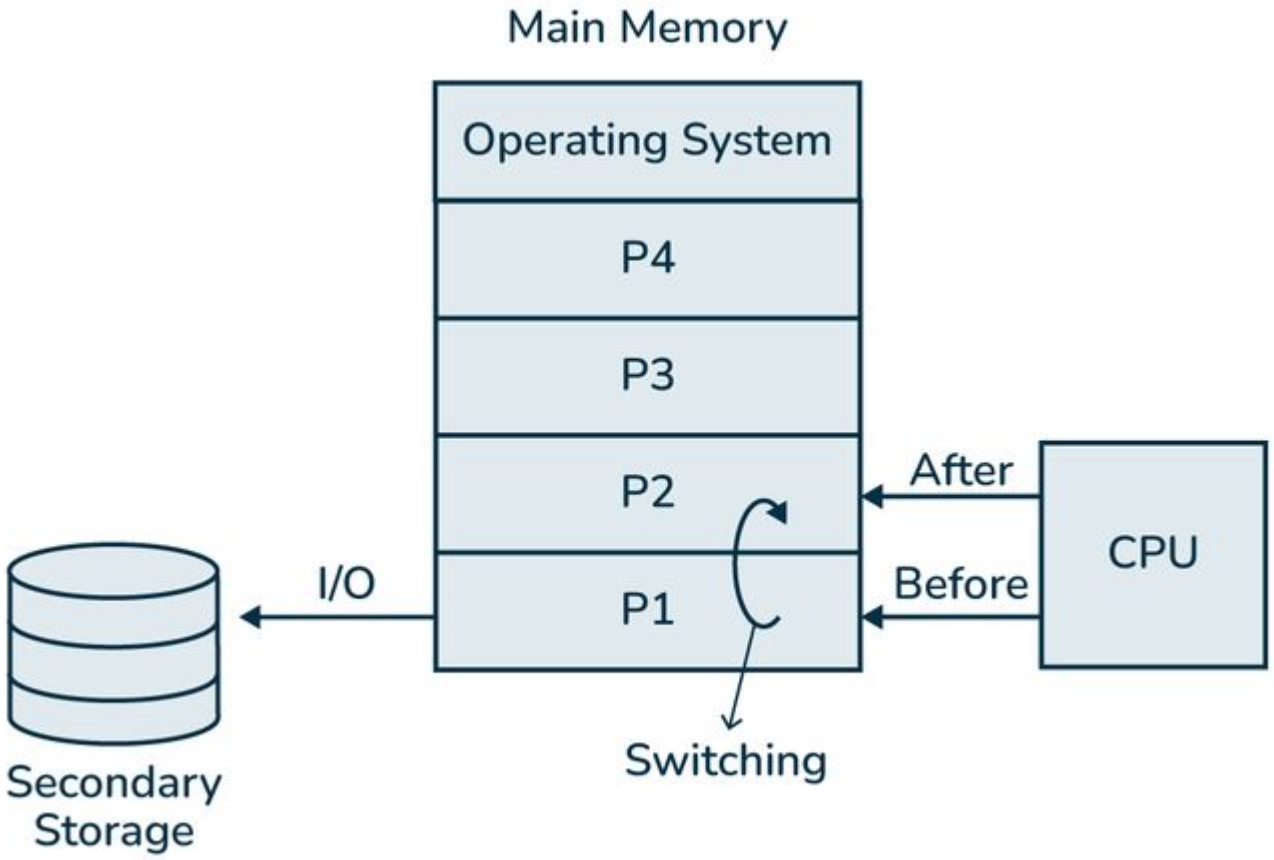
# Characteristics Multiprogramming

- Many jobs in memory.
- **Job Scheduling**: Only one job runs on the CPU at a time
- **Memory Management** is essential
- **State tracking** of each job: ready, running, waiting
- Jobs are usually **independent** and **non-interactive**

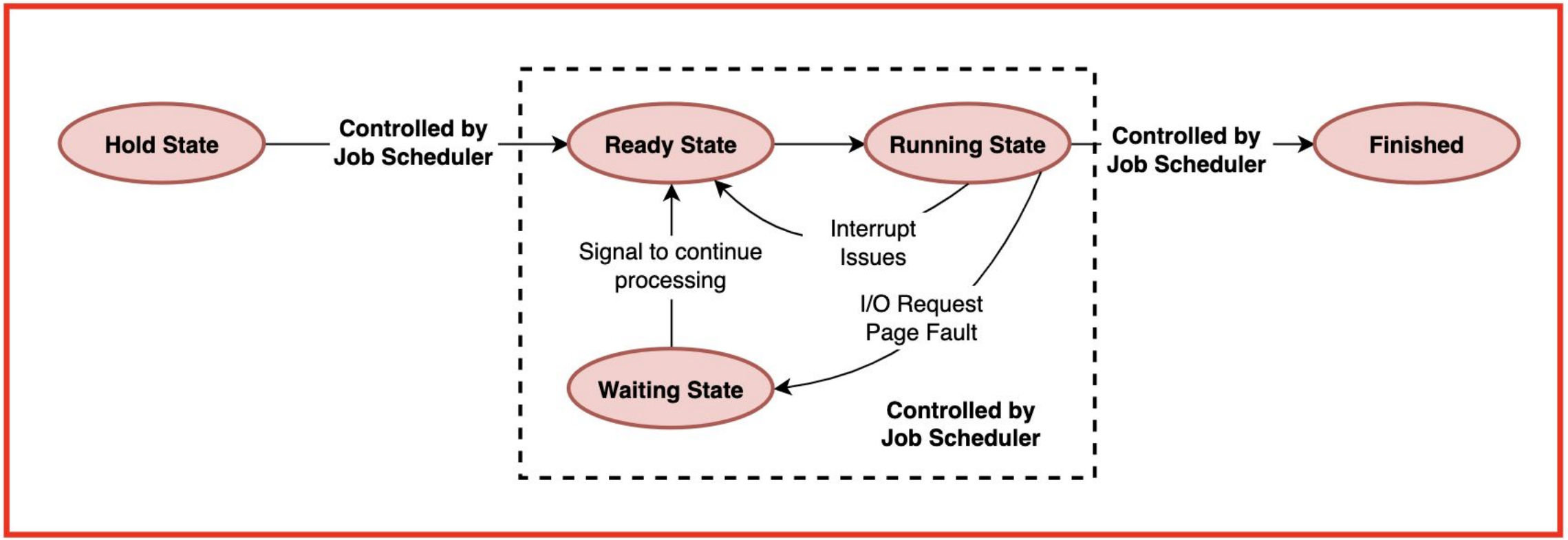
# Multiprogramming vs. Simple Batch Systems

| Feature | Simple Batch | Multiprogramming |
|---|---|---|
| Jobs in memory | One at a time | Multiple jobs |
| CPU Utilization | Often idle | High utilization |
| Scheduling | Sequential | Based on algorithm |
| Memory use | Not optimized | Efficiently shared |

# Architecture of Multiprogram Batch OS



# Job States and Transitions



# Job States and Transitions

**Ready**: Waiting for CPU

**Running**: Using CPU

**Waiting**: Waiting for I/O

**Terminated**: Finished execution

# Scheduling in Multiprogramming OS

- **Scheduling** is crucial to decide which job gets the CPU
- Scheduler selects a job from the **ready queue**
- Happens when:
  - Running job finishes
  - Job requests I/O
  - Another job with higher priority arrives

# Types of CPU Scheduling Algorithms

- FCFS (First-Come-First-Served)
- SJF (Shortest Job First)
- Round Robin

# Advantages of Multiprogramming

- **Better CPU utilization** – keeps CPU busy
- Overlaps I/O and computation
- **Higher throughput** – more jobs completed per hour
- **Reduces waiting time** for shorter jobs

# Limitations and Challenges

- More **complex OS design**
- Requires **efficient memory allocation and protection**
- **Scheduling overhead** and **context switching**
- Risk of **deadlocks** if multiple jobs wait on resources

# Real-World Usage and Evolution

- Used in **mainframe systems** like **IBM OS/360**
- Forms basis for **modern multitasking OS** (Windows, Linux, UNIX)
- Concepts used in **cloud batch processing** and **job schedulers** (e.g., Kubernetes, cron).

# Quiz

- What is multiprogramming?
- Why do we need CPU scheduling?
- Name two CPU scheduling algorithms.
- What happens when a job waits for I/O?