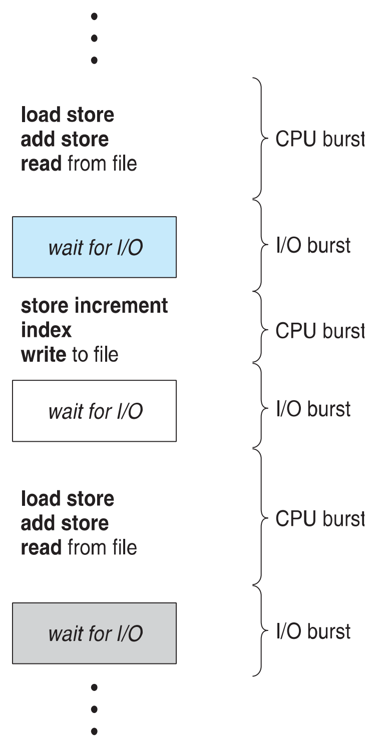
CPU Scheduling

LECTURE 5

CPU Scheduling

- Maximum CPU utilization obtained with multiprogramming
- Process execution consists of a cycle of CPU execution and I/O wait: CPU Burst and I/O Burst Cycle –
- CPU burst followed by I/O burst
- CPU burst distribution is of main concern



CPU Scheduler

- CPU scheduler selects the processes in ready queue, and allocates the CPU
  - Queue may be ordered in various ways
- CPU scheduling decisions may take place when a process:

1. Switches from running to waiting state
2. Switches from running to ready state
3. Switches from waiting to ready
4. Terminates

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program
- Dispatch latency – the time it takes for the dispatcher to stop one process and start another running

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** –Is the time spent when the process is in the ready state [requested a CPU] and gets the CPU for the first time.

Scheduling Algo Optimization Criteria

- **Max** CPU utilization
- **Max** throughput
- **Min** turnaround time
- **Min** waiting time
- **Min** response time

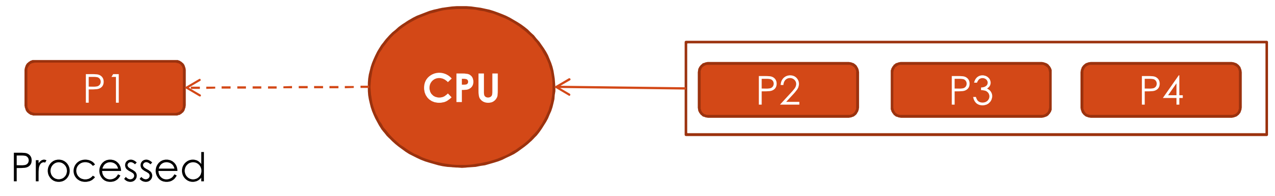
CPU Scheduling Algorithms

First Come First Served [FCFS]

- The processes are served as they enter the queue.
- The process that enters first will run first.
- It uses Non-Preemptive approach
- Processes are run until they are finished
- After completion of one process, next process from the queue is selected.
- Disadvantage of this approach is it takes a lot of time to run.

CPU Scheduling Algorithms

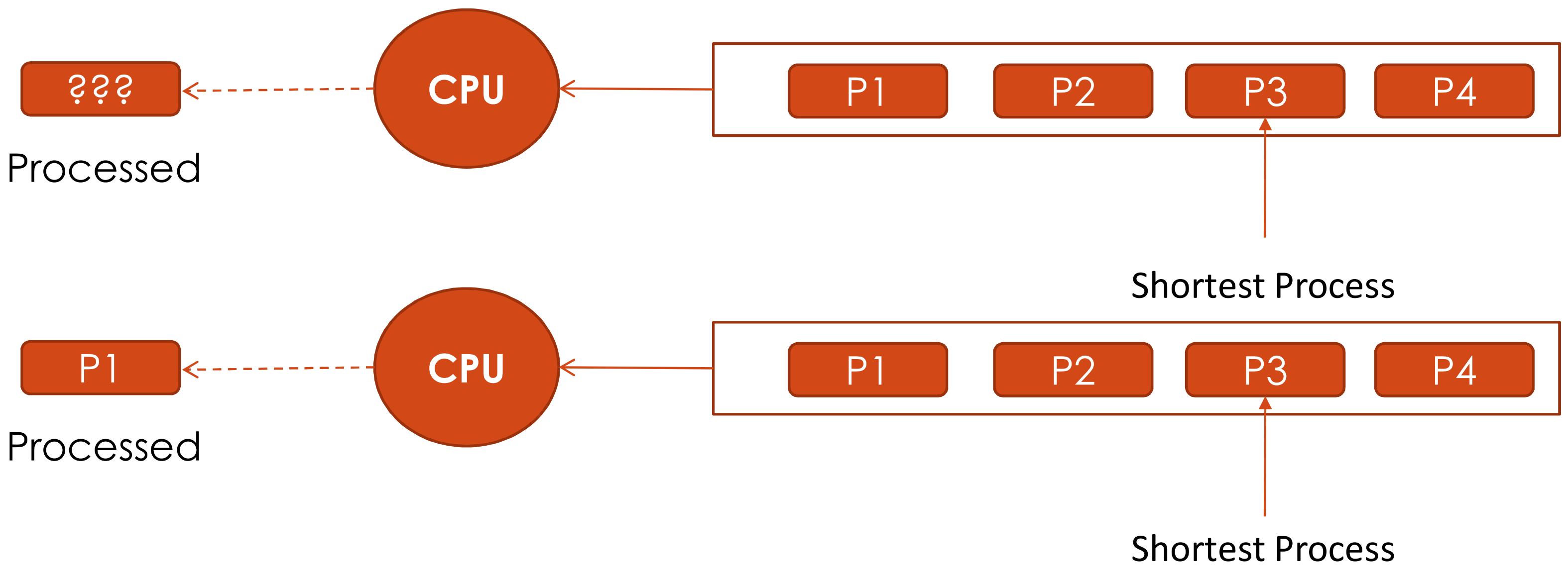
First Come First Served



CPU Scheduling Algorithms

First Come First Served

Which process runs first?



CPU Scheduling Algorithms

Exercise 1

- Assume there are five processes (P1, P2, P3, P4, P5) with the following burst times. The processes are assumed to have arrived in the order P1 , P2 , P3 , P4 , P5 , all at time 0.

Process	Burst Time
P1	2
P2	1
P3	8
P4	4
P5	5

1) What is the order of execution for the FCFS scheduling algorithm?
2) What is the turnaround time of each process?
3) What is the waiting time for each process

CPU Scheduling Algorithms

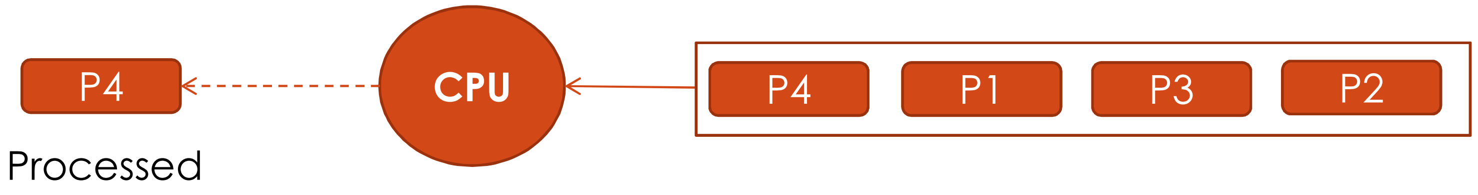
Shortest Job First [SJF]

- The process which has the shortest job will be served first.
- For this technique, CPU Burst Time of the processes should be known

CPU Scheduling Algorithms

Shortest Job First

Process	CPU Burst Time
P1	6 millisecond
P2	8 millisecond
P3	7 millisecond
P4	3 millisecond



CPU Scheduling Algorithms

Exercise 2

Process	CPU Burst Time
P1	6 milliseconds
P2	8 milliseconds
P3	7 milliseconds
P4	3 milliseconds

1) **What is the average waiting time**

CPU Scheduling Algorithms

Priority Scheduling [PS]

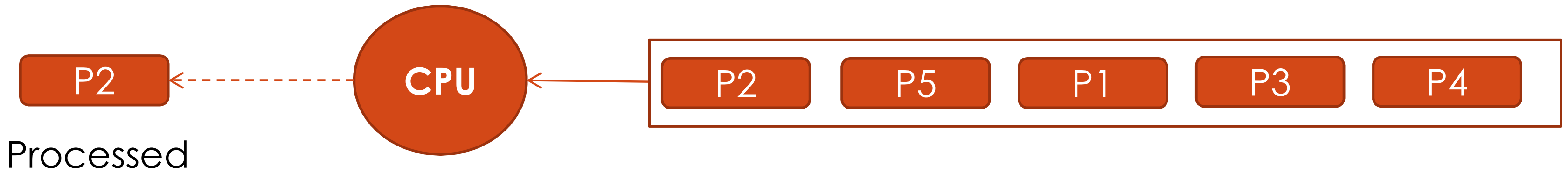
- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer means highest priority)
  - Preemptive
  - Non-preemptive
- **Problem:** Starvation – low priority processes may never execute
- **Solution:** Aging – as time progresses increase the priority of the process

CPU Scheduling Algorithms

Priority Scheduling

Process	CPU Burst Time	Priority
P1	10 millisecond	3
P2	1 millisecond	1
P3	2 millisecond	4
P4	1 millisecond	5
P5	5 millisecond	2

Process	Wait Time
P1	……..
P2	……..
P3	………
P4	………
P5	…………

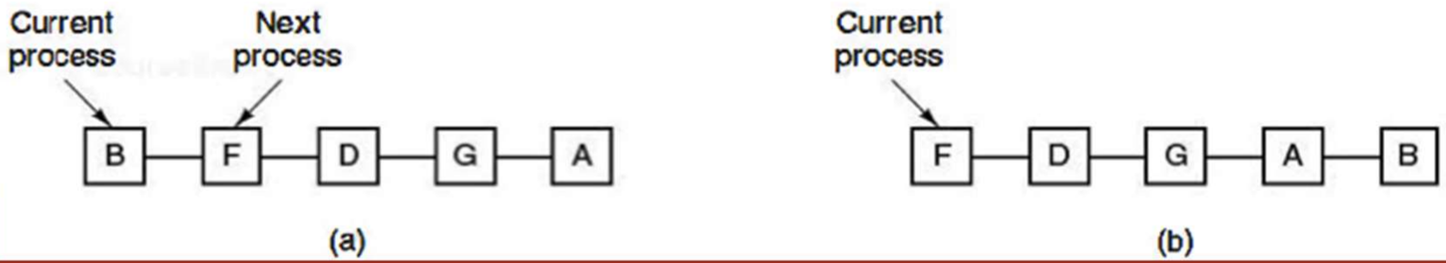


Avg. Wait Time = **…**

CPU Scheduling Algorithms

Round Robin Scheduling [RR]

- Each process gets a small unit of CPU time (time quantum q), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time in chunks of at most q time units at once. No process waits more than (n-1)q time units.
- Timer interrupts every quantum to schedule next process
- A blocked process will be shifted to the end of the list containing all the running process.
- The list is maintained by the scheduler.

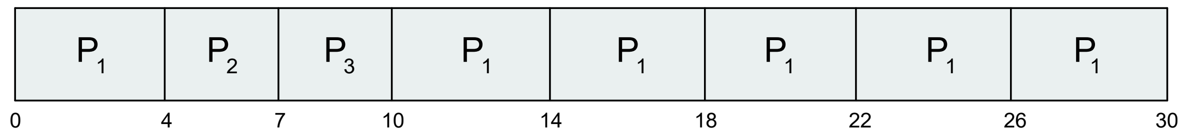


CPU Scheduling Algorithms

Round Robin Scheduling

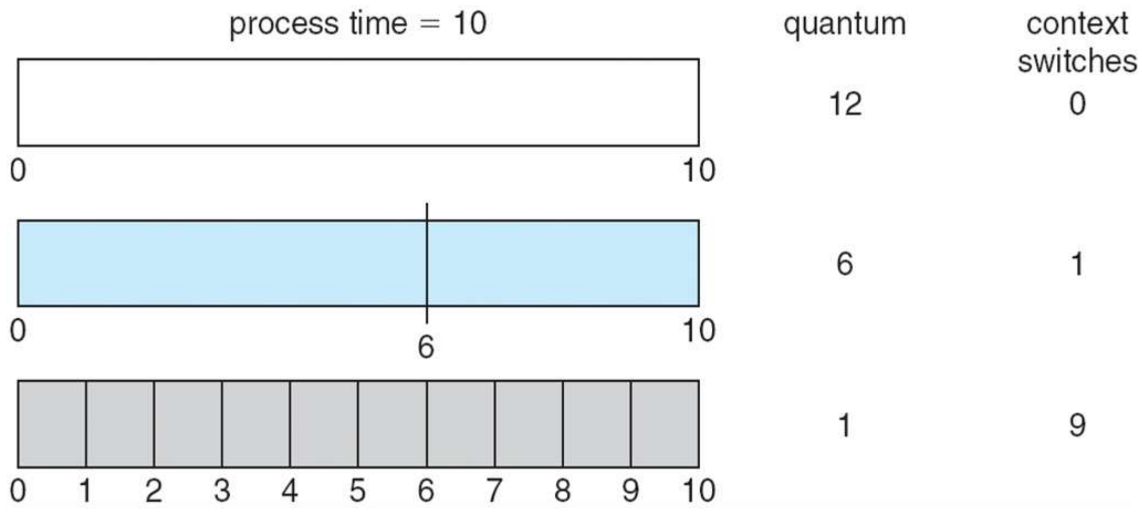
Quantum [Q] here is 4

Process	CPU Burst Time
P1	24 millisecond
P2	3 millisecond
P3	3 millisecond



CPU Scheduling Algorithms

Round Robin Scheduling



CPU Scheduling Algorithms

Exercise 3

Process	Burst	Priority	Arrival
P1	20	40	0
P2	25	30	25
P3	25	30	30
P4	15	35	60
P5	10	5	100
P6	10	10	105

The following processes are being scheduled using a preemptive, round-robin scheduling algorithm. Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed above, the system also has an idle task (which consumes no CPU resources and is identified as idle). This task has priority 0 and is scheduled whenever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

a. Show the scheduling order of the processes using a Gantt chart.
b. What is the turnaround time for each process?
c. What is the waiting time for each process?
d. What is the CPU utilization rate?

CPU Scheduling Algorithms

Multi-Level Queue Scheduling

- Ready queue is partitioned into separate queues, eg:
  - foreground
  - background
- If a process is allocated to a given queue it stays there permanently
- Each queue has its own scheduling algorithm:
  - foreground – RR [Round Robin Scheduling]
  - background – FCFS [First Come First Served]

CPU Scheduling Algorithms

Multi-Level Queue Scheduling

- Scheduling must be done between the queues as well:
  - Fixed priority scheduling; (i.e., serve all from foreground then from background).
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR and 20% to background in FCFS

CPU Scheduling Algorithms

Multi-Level Queue Scheduling

Exercise 4

- Consider the below table of four processes under Multilevel queue scheduling. The queue number denotes the queue of the process.

Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1

The priority of Queue 1 is greater than Queue 2. Queue 1 uses Round Robin (Time Quantum = 2) and Queue 2 uses FCFS. Show the scheduling order of the processes using MLQS ?

CPU Scheduling Algorithms

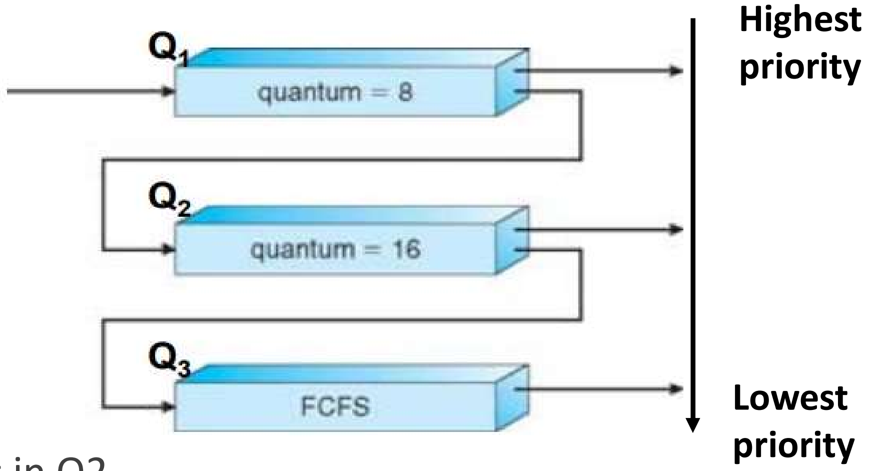
Multilevel Feedback Queue

- A process can move between the various queues; aging can be implemented this way
- Fixed priority scheduling; (i.e., serve all from foreground then from background).
- Multilevel-feedback-queue scheduler defined by the following parameters:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service

CPU Scheduling Algorithms

Multilevel Feedback Queue

- Three queues:
  - Q1 – RR with time quantum 8 milliseconds
  - Q2 – RR with time quantum 16 milliseconds
  - Q3 – FCFS
  - The scheduler first executes all processes in Q1.
  - Only when Q1 is empty will it executes processes in Q2.
  - Similarly, processes in Q3 will be executed only if Q1 and Q2 are empty.
  - A process that arrives for Q2 will preempt a process in Q3.



CPU Scheduling Algorithms

Multilevel Feedback Queue

- Scheduling
  - A new process enters queue Q1
    - When it gains CPU, the process receives 8 milliseconds
    - If it does not finish in 8 milliseconds, the process is moved to queue Q2
  - At Q2, the process receives 16 additional milliseconds
    - If it still does not complete, it is preempted and moved to queue Q3
  - If a process does not use up its quantum in the current level, it will keep its current queuing level and be put into the end of the queue.

CPU Scheduling Algorithms

Multilevel Feedback Queue

Exercise 5

- Consider the below table of four processes under Multilevel Feedback scheduling. The queue number denotes the queue of the process.

Process	Arrival Time	CPU Burst Time
P1	0	53
P2	0	17
P3	0	68
P4	0	24

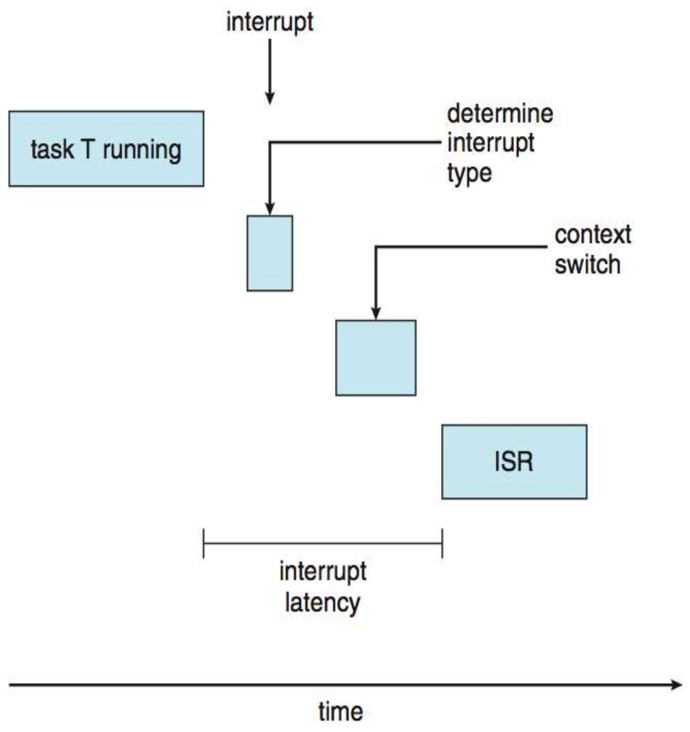
- Three Queues
- Priority of Queues are :
  - Q1 : Highest Priority
  - Q2:
  - Q3 :Lowest Priority
- Q1 Uses RR (TQ=17)
- Q2 Uses RR (TQ=25)
- Q3 uses FCFS

Find Average TAT and WT

Real Time CPU Scheduling

- Can present obvious challenges
- **Soft real-time systems** – no guarantee as to when critical real-time process will be scheduled
- **Hard real-time systems** – task must be serviced by its deadline
- Two types of latencies affect performance
  - Interrupt latency – time from arrival of interrupt to start of routine that services interrupt
  - Dispatch latency – time for schedule to take current process off CPU and switch to another

Interrupt Service Routine (ISR)



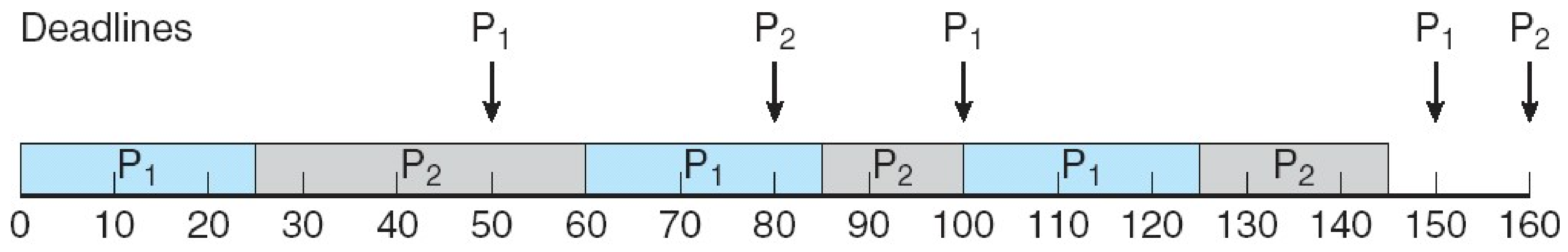
Real Time CPU Scheduling

Earliest Deadline First Scheduling

- Priorities are assigned according to deadlines:
  - the earlier the deadline, the higher the priority;
  - the later the deadline, the lower the priority

P1: Deadline = 50, Burst = 25

P2: Deadline = 80, Burst = 35



CPU Scheduling Algorithms

Earliest Deadline First Scheduling

Exercise 6

- Suppose there are four processes with their arrival time, burst time, and deadline time as follows:

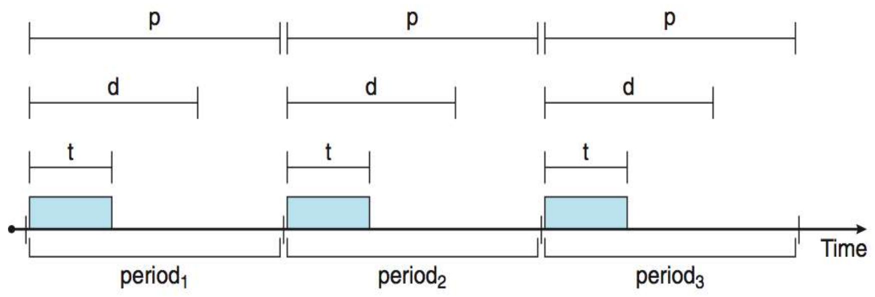
Process	Arrival Time	CPU Burst Time	Deadline
P1	0	10	26
P2	1	3	3
P3	2	5	9
P4	3	7	20

a. Show the scheduling order of the processes using a Gantt chart.
b. What is the turnaround time for each process?
c. What is the waiting time for each process?

Real Time CPU Scheduling

Priority Based Scheduling

- For real-time scheduling, the scheduler must support preemptive, priority-based scheduling
- For hard real-time must also provide the ability to meet deadlines
- Processes have new characteristics: periodic ones require CPU at constant intervals
  - Has processing time t, deadline d, period p
  - $0 \le t \le d \le p$
  - Rate of a periodic task is 1/p



Multiple Processor Scheduling

- CPU scheduling more complex when multiple CPUs are available
- Homogeneous processors within a multiprocessor
- Asymmetric multiprocessing [ASMP]– only one processor accesses the system data structures, alleviating the need for data sharing
- Symmetric multiprocessing [SMP]– each processor is self-scheduling, all processes in common ready queue, or each has its own private queue of ready processes
  - Currently, most common

Multiple Processor Scheduling

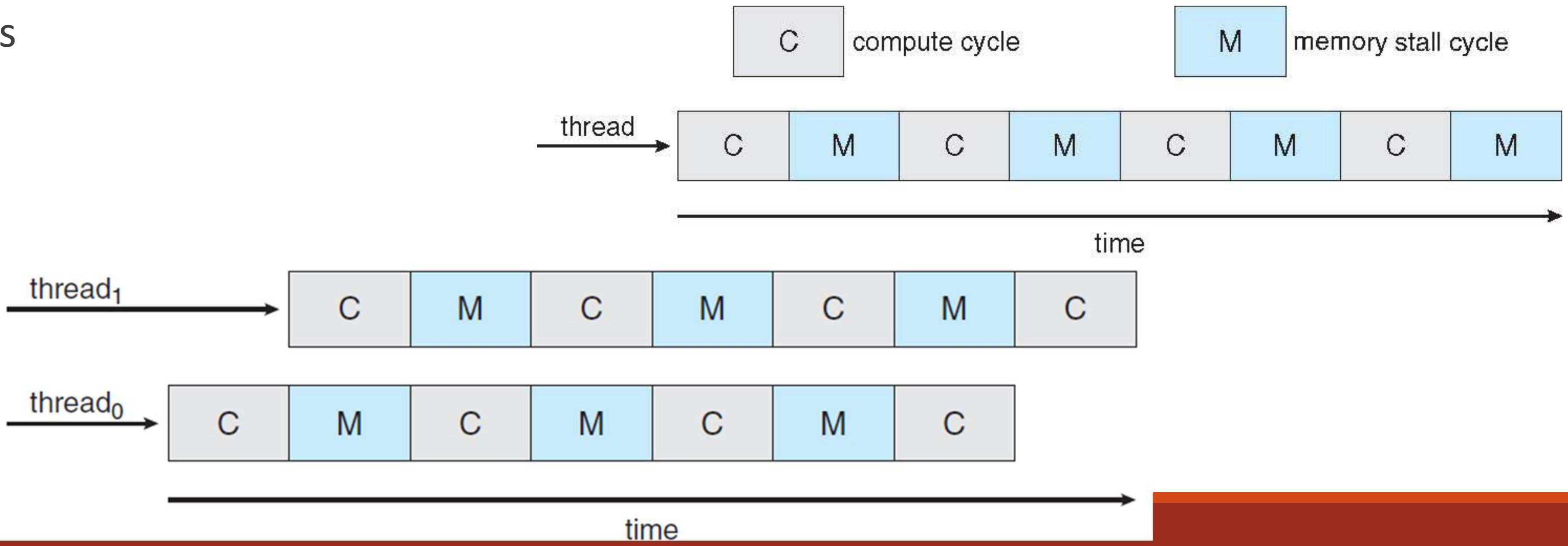
Load Balancing

- Load balancing attempts to keep workload evenly distributed in Multiple processors environments
  - Push migration – periodic task checks load on each processor, and if found overloaded then it pushes the task from the overloaded CPU to other CPUs
  - Pull migration – idle processors pull the waiting tasks from the busy processor

Multiple Processor Scheduling

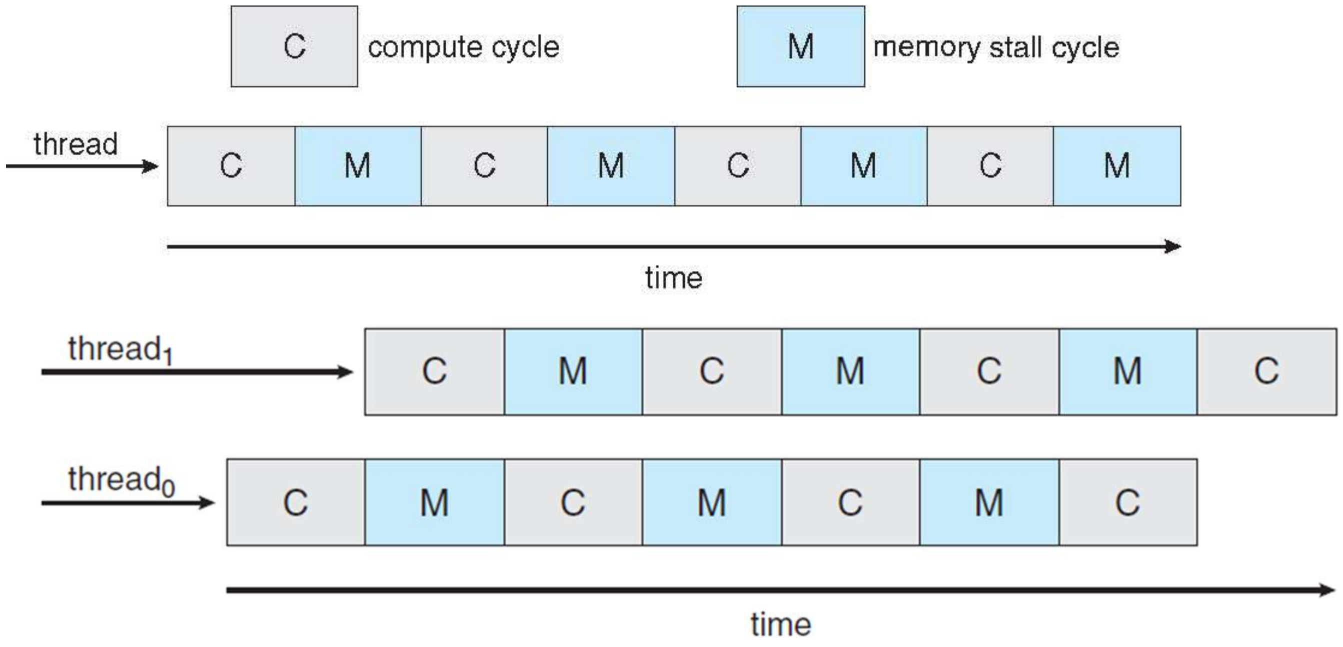
Multicore Processors

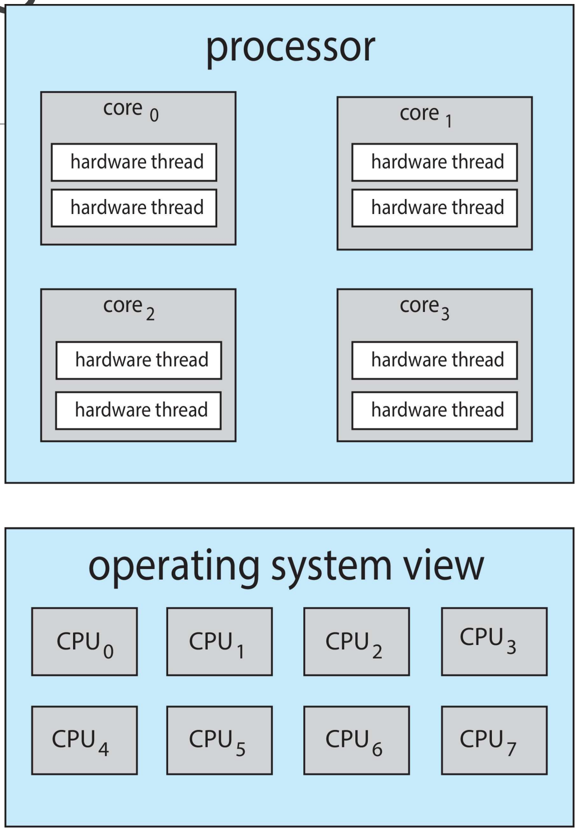
- Recent trend to place multiple processor cores on same physical chip
- Faster and consumes less power
- Multiple threads per core also growing
- Takes advantage of memory stall to make progress on another thread while memory retrieve happens



Multiple Processor Scheduling

Multicore Processors


C
compute cycle
M
memory stall cycle
thread
C M C M C M C M
time
thread$_1$
C M C M C M C
thread$_0$
C M C M C M C
time


processor
core$_0$
hardware thread
hardware thread
core$_1$
hardware thread
hardware thread
core$_2$
hardware thread
hardware thread
core$_3$
hardware thread
hardware thread
operating system view
CPU$_0$
CPU$_1$
CPU$_2$
CPU$_3$
CPU$_4$
CPU$_5$
CPU$_6$
CPU$_7$