**1）**Show how can you avoid race conditions with the Load-Linked/Store-Conditional instruction pair.

First of all, race conditions is when threads or processes perform read and write operations on data items in same shared space, trying to change it at the same time. These threads or processes will “race” for operating on the shared memory, causing the result to be completely dependent on the scheduling algorithm. To avoid this, we need mutual exclusion that allows only one thread can get in at one time and others have to wait until it’s completed. This special area is called critical section and it’s where load-linked/store conditional comes in to create condition to set flag for each incoming thread and reset it after thread is done.

**2)** Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

|  |  |  |
| --- | --- | --- |
| Process | Burst Time | Priority |
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 3 |
| P4 | 1 | 4 |
| P5 | 5 | 2 |

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.  
a. Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, a nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1) scheduling.  
b. What is the turnaround time of each process for each of the scheduling algorithms in part a?  
c. What is the waiting time of each process for each of the scheduling algorithms in part a?

FCFS:

-------------------------------------------------------------------------------------------------

| P1 | P2 | P3 | P4.  | P5 |

-------------------------------------------------------------------------------------------------

0 10 11 13 14 19

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Arrival Time | Process | Burst Time | Priority | Finish Time | Turnaround time | Waiting Time |
| 0 | P1 | 10 | 3 | 10 | 10 | 0 |
| 0 | P2 | 1 | 1 | 11 | 11 | 10 |
| 0 | P3 | 2 | 3 | 13 | 13 | 11 |
| 0 | P4 | 1 | 4 | 14 | 14 | 13 |
| 0 | P5 | 5 | 2 | 19 | 19 | 14 |
| Average | | | | | 13.4 | 9.6 |

SJF (non-preemptive):

-------------------------------------------------------------------------------------------------

| P1  | P4 | P3 | P5  | P1 |

-------------------------------------------------------------------------------------------------

0 1 2 4 9 19

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Arrival Time | Process | Burst Time | Priority | Finish Time | Turnaround time | Waiting Time |
| 0 | P1 | 10 | 3 | 19 | 19 | 9 |
| 0 | P2 | 1 | 1 | 1 | 1 | 0 |
| 0 | P3 | 2 | 3 | 4 | 4 | 2 |
| 0 | P4 | 1 | 4 | 2 | 2 | 1 |
| 0 | P5 | 5 | 2 | 9 | 9 | 4 |
| Average | | | | | 7 | 3.2 |

SJF (preemptive, shortest next CPU burst):

------------------------------------------------------------------------------------------------------------

| P2  | P4 | P3 | P5  | P1 |

------------------------------------------------------------------------------------------------------------

0 1 2 4 9 19

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Arrival Time | Process | Burst Time | Priority | Finish Time | Turnaround time | Waiting Time |
| 0 | P1 | 10 | 3 | 19 | 19 | 9 |
| 0 | P2 | 1 | 1 | 1 | 1 | 0 |
| 0 | P3 | 2 | 3 | 4 | 4 | 2 |
| 0 | P4 | 1 | 4 | 2 | 2 | 1 |
| 0 | P5 | 5 | 2 | 9 | 9 | 4 |
| Average | | | | | 7 | 3.2 |

Non-preemptive priority:

--------------------------------------------------------------------------------------------------------------

| P2  | P5 | P1 | P3  | P1. |

--------------------------------------------------------------------------------------------------------------

0 1 6 16 18 19

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Arrival Time | Process | Burst Time | Priority | Finish Time | Turnaround time | Waiting Time |
| 0 | P1 | 10 | 3 | 16 | 16 | 6 |
| 0 | P2 | 1 | 1 | 1 | 1 | 0 |
| 0 | P3 | 2 | 3 | 18 | 18 | 16 |
| 0 | P4 | 1 | 4 | 19 | 19 | 18 |
| 0 | P5 | 5 | 2 | 6 | 6 | 1 |
| Average | | | | | 13.4 | 8.2 |

Round Robin:

-----------------------------------------------------------------------------------------------------------------------------------

| P1 | P2  | P3  | P4  | P5  | P1  | P3  | P5  | P1  | P5  | P1  | P5  | P1  | P5  | P1  | P1  | P1  | P1  | P1 |

-----------------------------------------------------------------------------------------------------------------------------------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Arrival Time | Process | Burst Time | Priority | Finish Time | Turnaround time | Waiting Time |
| 0 | P1 | 10 | 3 | 19 | 19 | 9 |
| 0 | P2 | 1 | 1 | 2 | 2 | 1 |
| 0 | P3 | 2 | 3 | 7 | 7 | 5 |
| 0 | P4 | 1 | 4 | 4 | 4 | 3 |
| 0 | P5 | 5 | 2 | 14 | 14 | 9 |
| Average | | | | | 9.2 | 5.4 |

d. Which of the schedules in part a results in the minimal average waiting time (over all processes)?

|  |  |
| --- | --- |
| FCFS | 9.6 |
| SJF (nonpreemptive) | 3.2 |
| SJF (preemptive, shortest next CPU burst) | 3.2 |
| Nonpreemptive Priority | 8.2 |
| Round Robjn | 5.4 |

The answer is SJF either nonpreemptive or preemptive has the minimum average waiting time.

**3)** Explain why spinlocks are not appropriate for uniprocessor systems yet may be suitable for multiprocessor systems**.**

Spinlocks are not appropriate because they consume a large amount of cpu resource and the only condition to break a process out of a spinlock is to start a different process. If the current process haven’t return the control to the processor, other processes won’t be able to set condition for start the first process. Therefore, it is less expensive and more efficient to use spinlocks in a multiprocessor system where processes work on their own processor and modify in order to release the first process from the spinlock.