Tutorial 3

Question 1

Tutorial 3

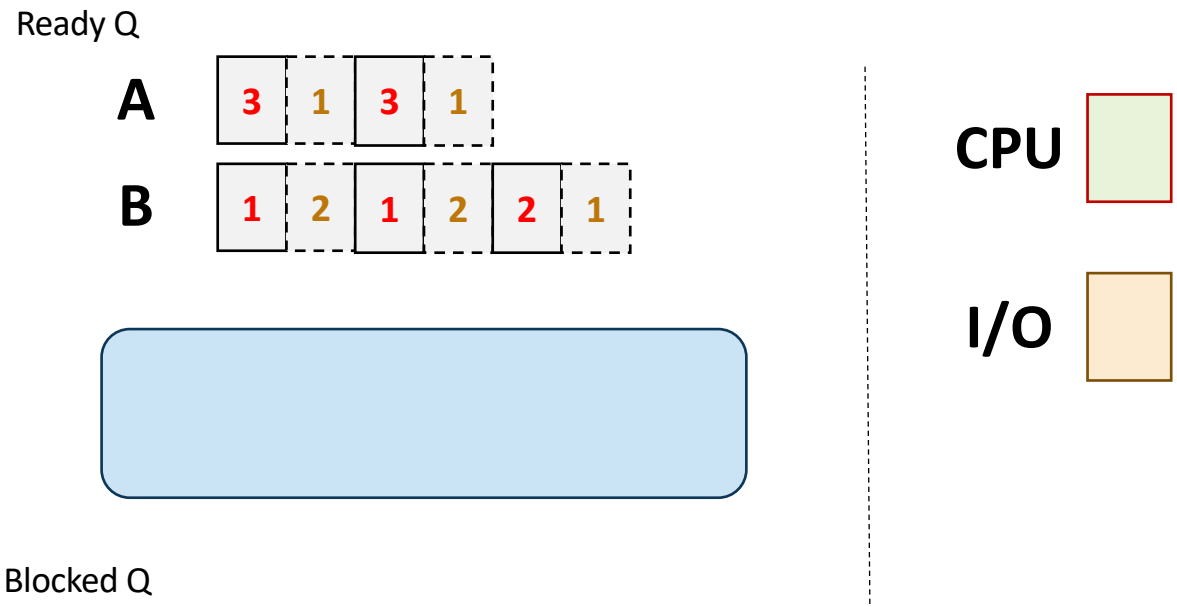
Question 2

First Come First Serve

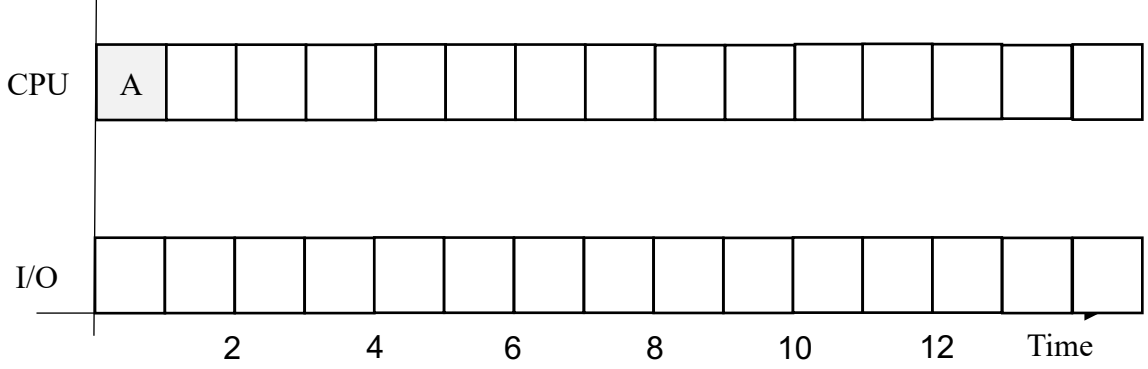
- We assume scheduler kicks in at the beginning of the time step whenever it is triggered

➔Show the result of the scheduling for the time step after scheduler finished its job

Initialization



Time: 1



Ready Q

B 1 2 1 2 2 1

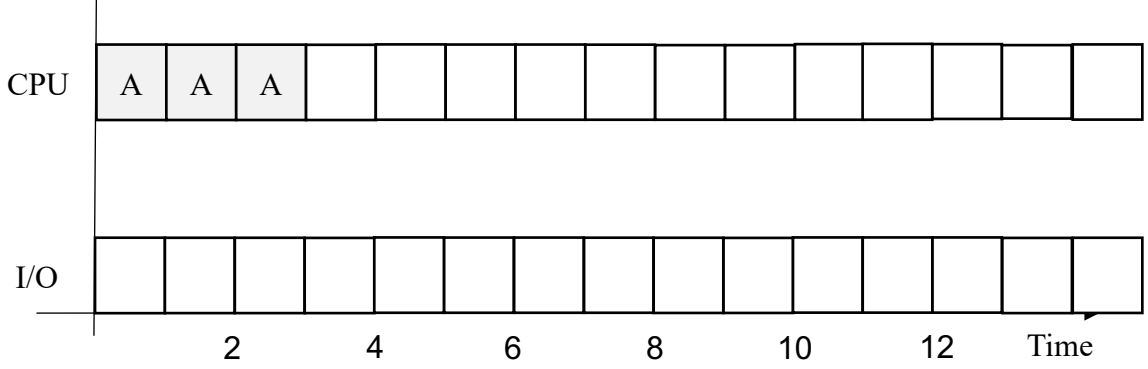
CPU **A** 3 1 3 1

I/O

A is chosen as it is the first in Q

Blocked Q

Time: 3



Ready Q

B 1 2 2 1

C 2

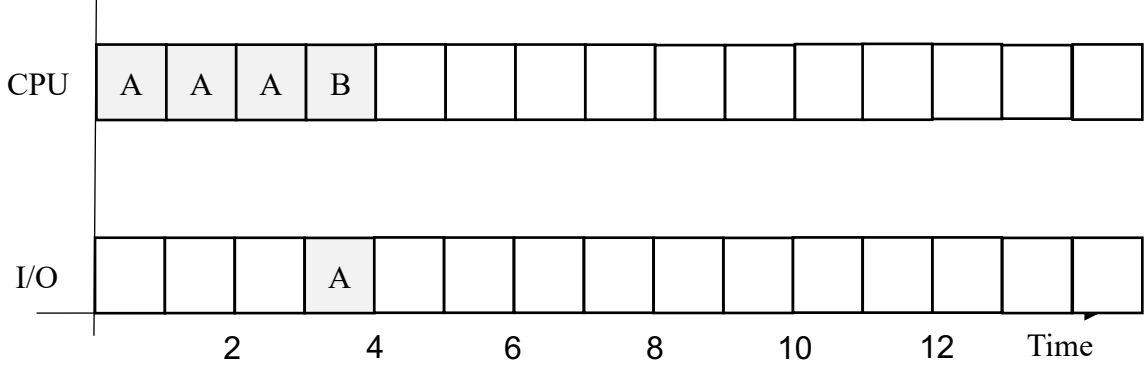
CPU **A** 1 1 3 1

I/O

A still working, C arrives

Blocked Q

Time: 4



Ready Q

C 2

CPU **B** 1 2 1 2 2 1

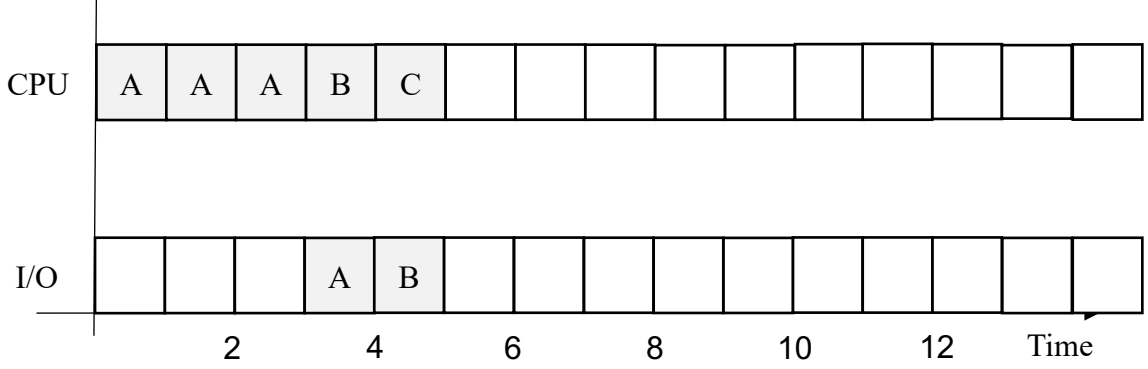
I/O **A**

A blocks, B get CPU

Blocked Q

A 1 3 1

Time: 5



Ready Q

A 3 1

CPU **C** 2

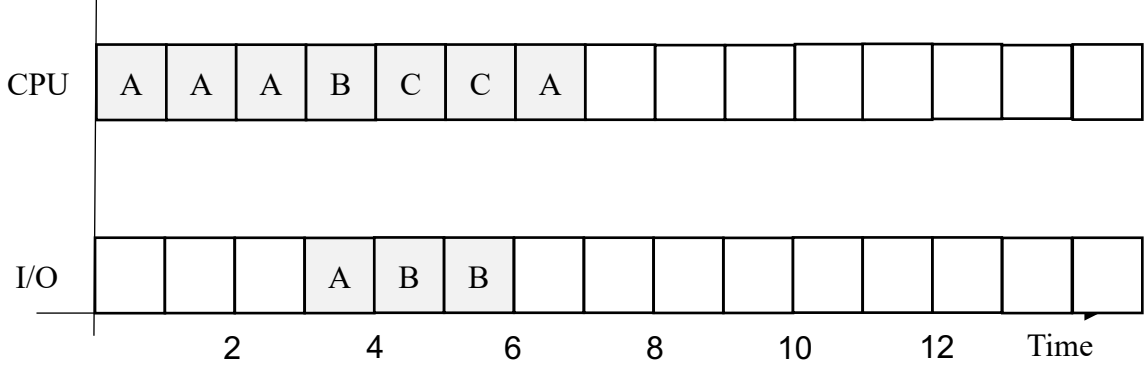
I/O **B**

B blocks, C get CPU

Blocked Q

B 2 1 2 2 1

Time: 7



Ready Q

B 1 2 2 1

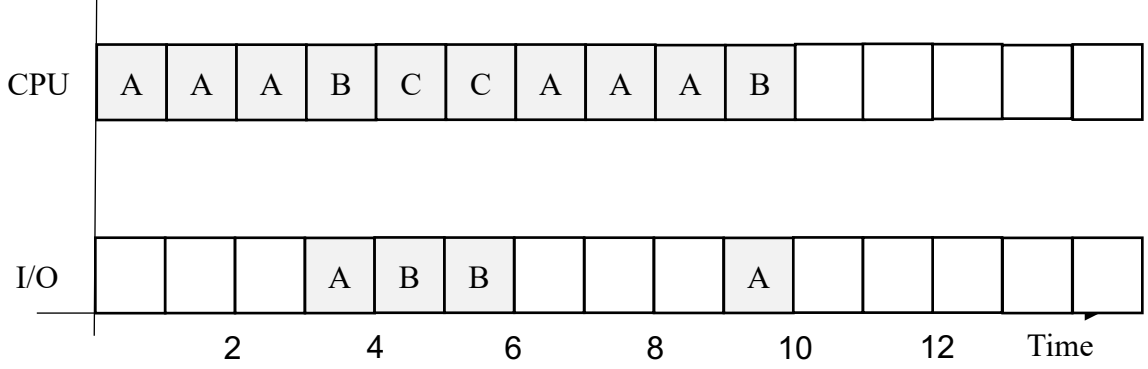
CPU A 3 1

I/O

C done; A get CPU; B unblocks

Blocked Q

Time: 10



Ready Q

B 1 2 2 1

CPU **B** 1 2 2 1

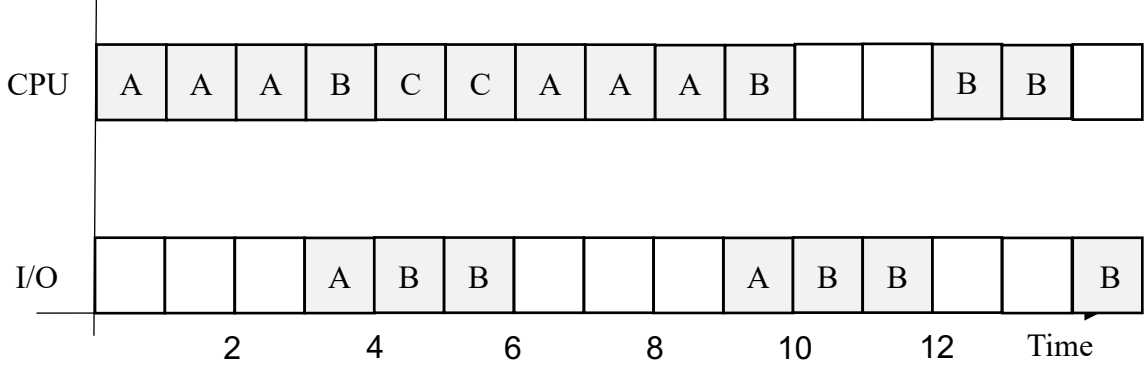
I/O **A**

A blocks; B get CPU

Blocked Q

A 1

Time: 15



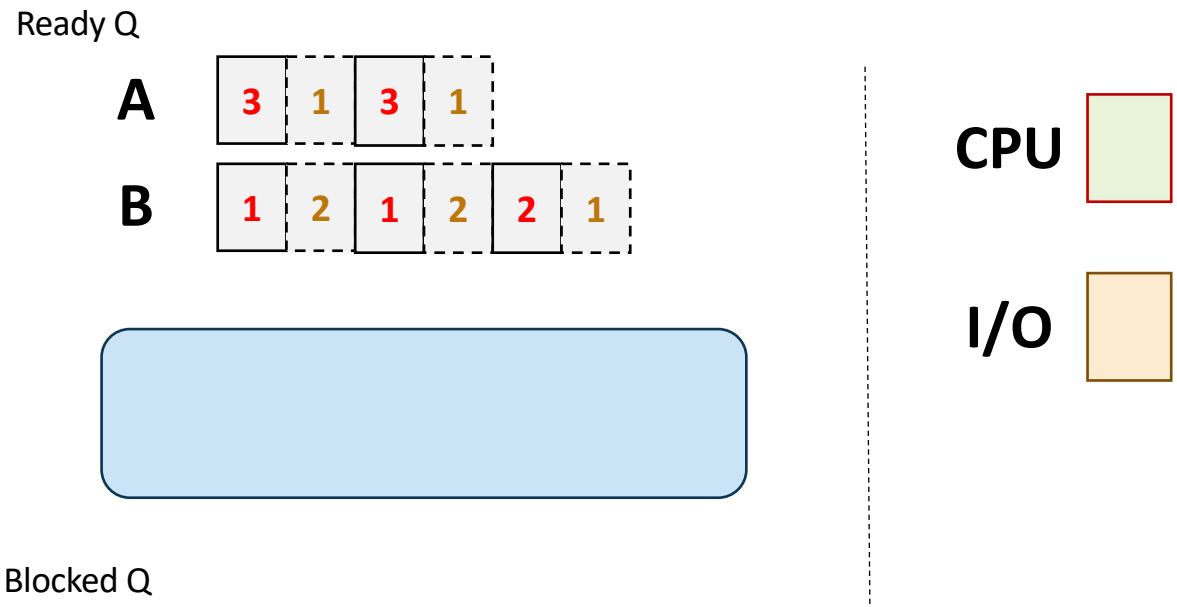
	Turnaround Time	Waiting Time
A		
B		
C		

Round Robin

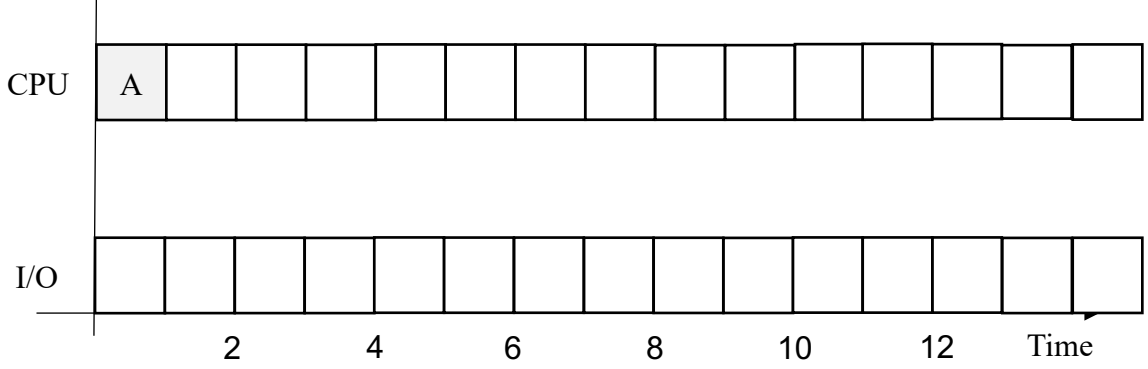
- We assume scheduler kicks in at the beginning of the time step whenever it is triggered

➔Show the result of the scheduling for the time step after scheduler finished its job

Initialization



Time: 1



Ready Q

B 1 2 1 2 2 1

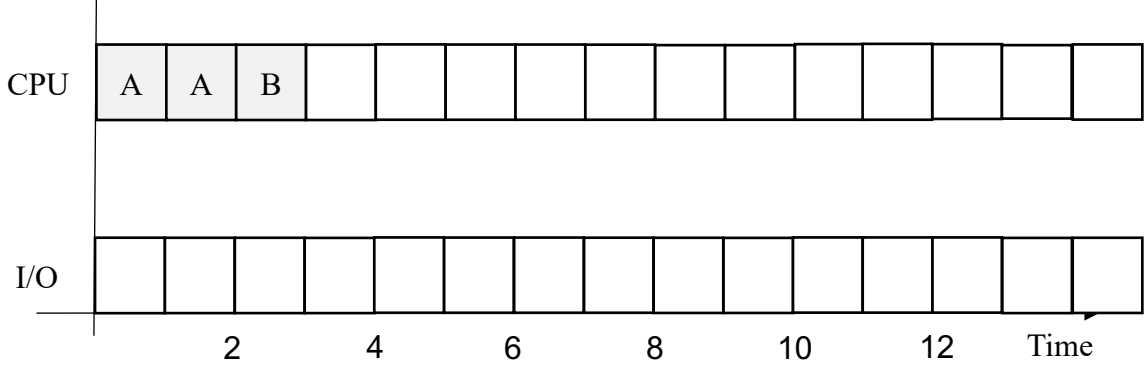
CPU **A** 3 1 3 1

I/O

A is chosen as it is the first in Q

Blocked Q

Time: 3



Ready Q

A 1 1 3 1

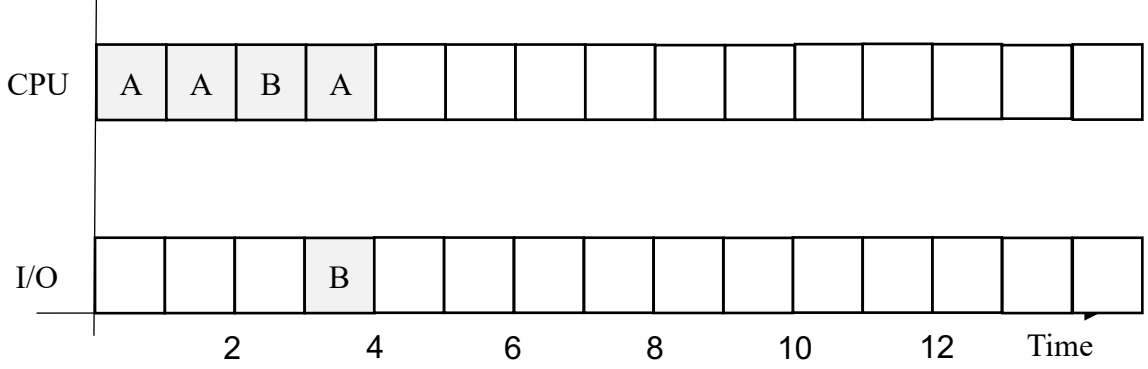
A is removed from CPU after 2 TUs.

Blocked Q

CPU **B** 1 2 1 2 2 1

I/O

Time: 4



Ready Q

C 2

CPU **A** 1 1 3 1

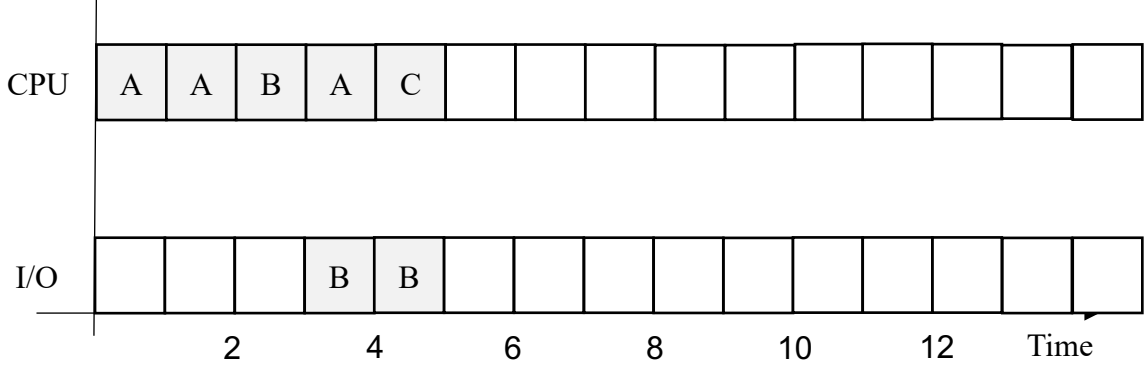
I/O **B**

A get chosen (first in Q);
B blocks; C arrives

Blocked Q

B 2 1 2 2 1

Time: 5



Ready Q

CPU C 2

I/O B

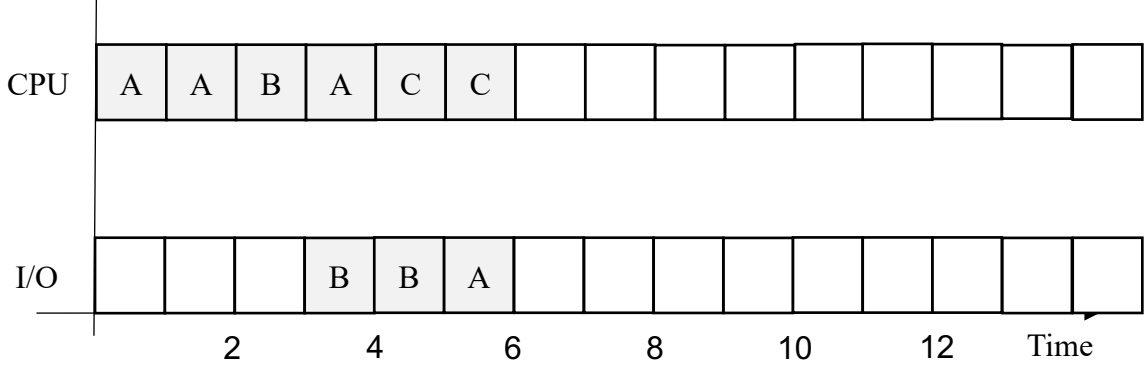
C is the only ready process

Blocked Q

B 1 1 2 2 1

A 1 3 1

Time: 6



Ready Q

B 1 2 2 1

B unblocks; C continues

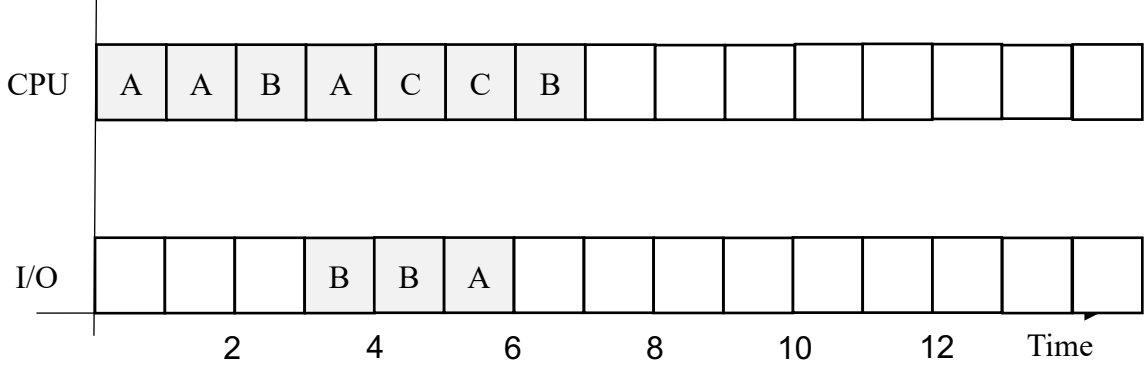
Blocked Q

A 1 3 1

CPU **C** 1

I/O **A**

Time: 7



Ready Q

A 3 1

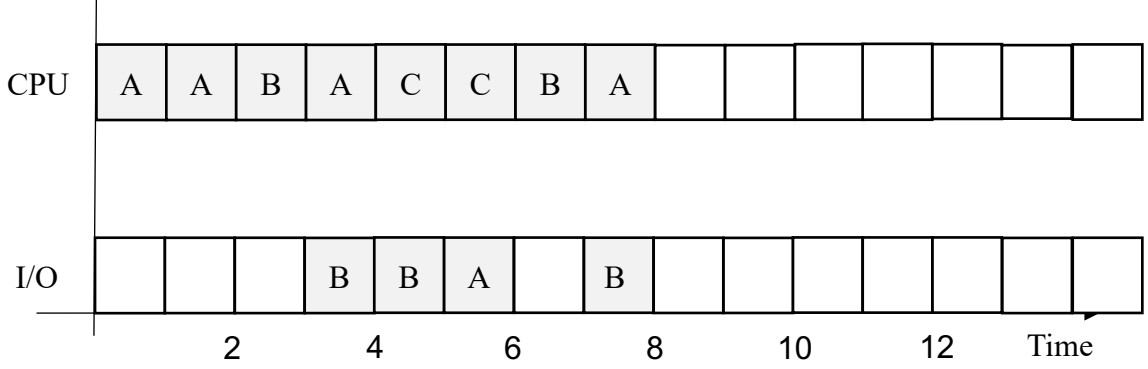
CPU **B** 1 2 2 1

I/O

B is chosen. A unblocks.

Blocked Q

Time: 8



Ready Q

CPU **A** 3 1

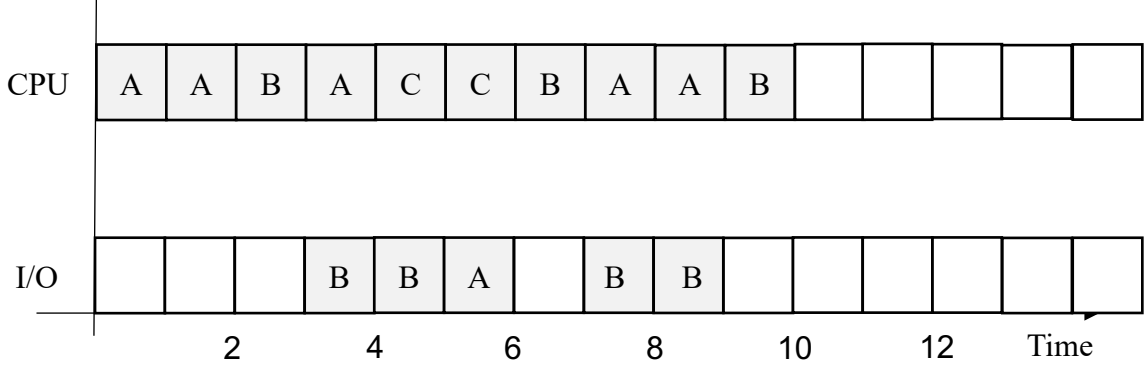
I/O **B**

B blocks.

Blocked Q

B 2 2 1

Time: 10



Ready Q

A 1 1

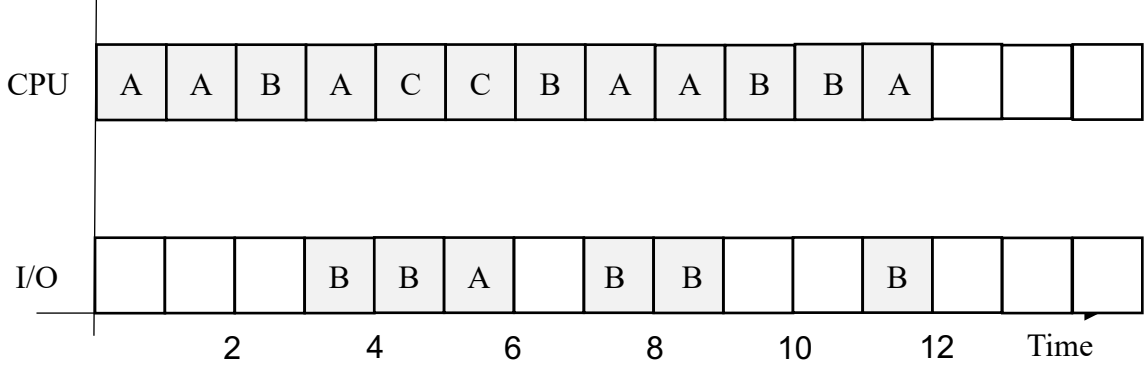
CPU **B** 2 1

I/O

A vacates CPU. B unblocks & get chosen.

Blocked Q

Time: 12



Ready Q

CPU **A** 1 1

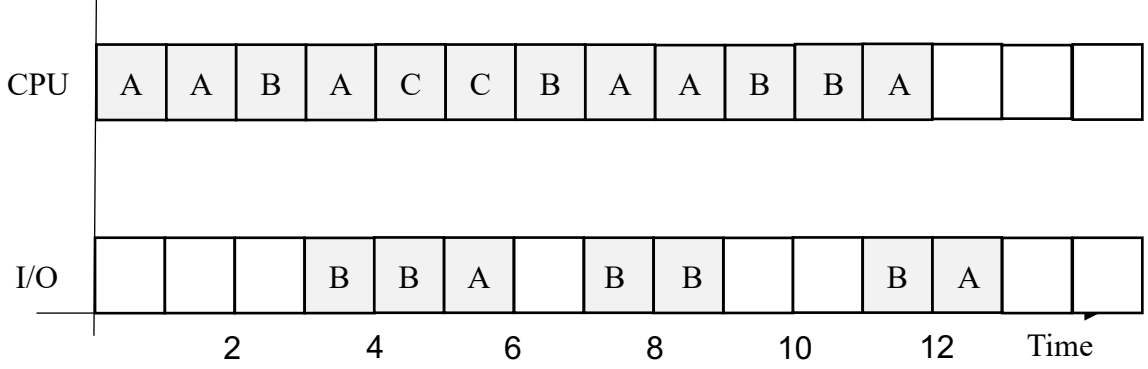
I/O **B**

B blocks.

Blocked Q

B 1

Time: 13



Ready Q

CPU

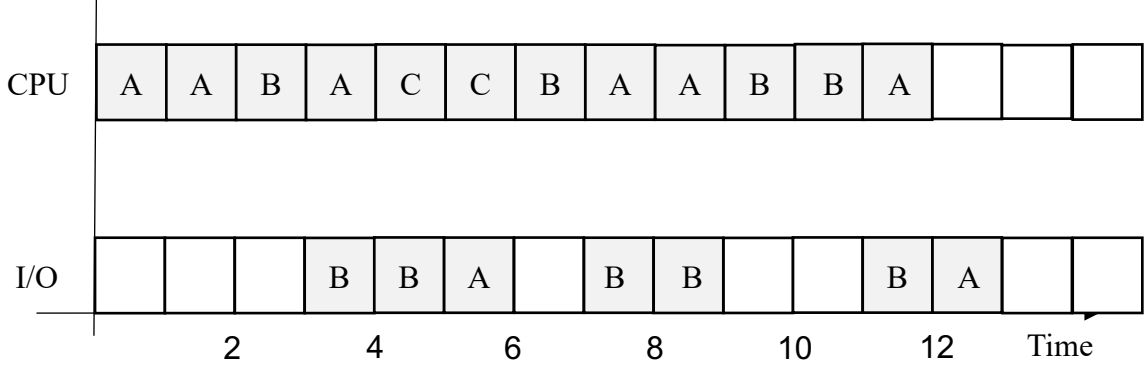
I/O **A**

B done; A blocks

Blocked Q

A 1

Time: 14



Tutorial 3

Question 1

Tutorial 3

Question 3

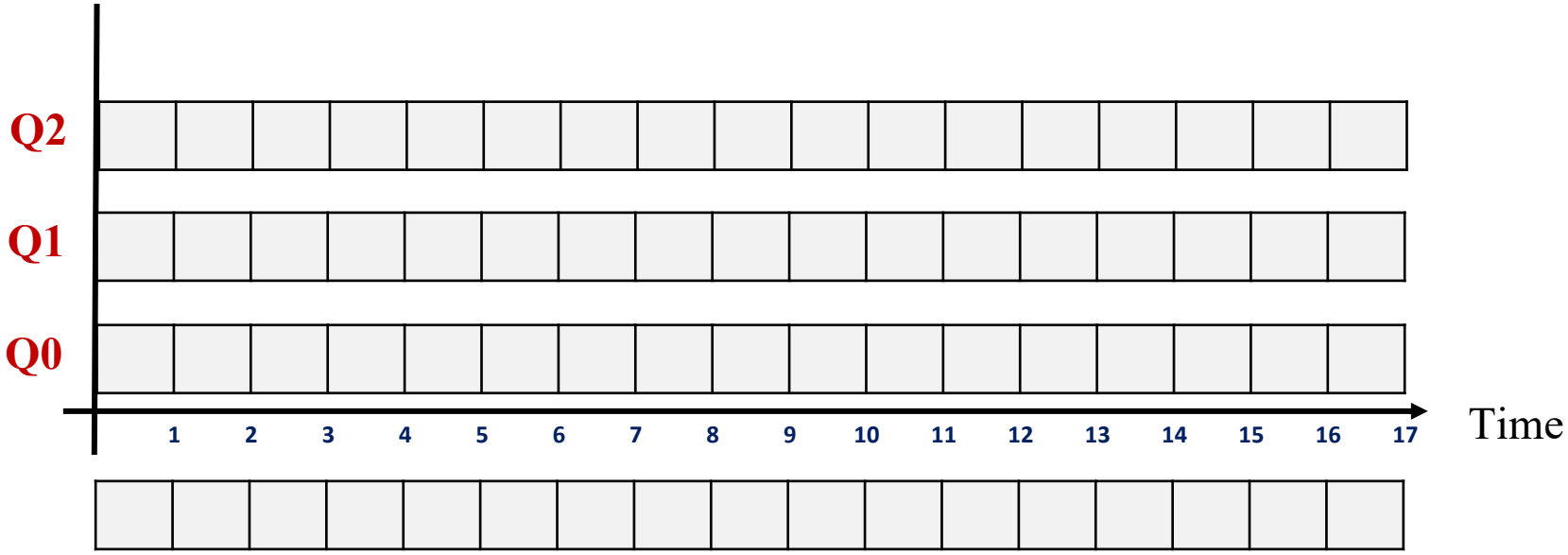
MLFQ: Rules?

- **Basic rules:**
  1. If Priority(A) > Priority(B) ➔
  2. If Priority(A) == Priority(B) ➔

- **Priority Setting/Changing rules:**
  1. New job ➔
  2. If a job fully utilized its time slice ➔
  3. If a job give up / blocks ➔

Q3. MLFQ (TQ=2, ITI=1)

Process	Behavior	Priority
A	**C3**, IO1, **C3**	
B	**C1**, IO1, **C1**, IO1, **C1**, IO1, **C1**	



Tutorial 3

Question 4

Pseudo-Code for RR Scheduler

```
RunningTask.TQLeft--;
if (RunningTask.TQLeft > 0) done!
//Check for another task to run
if ( ReadyQ.isEmpty() )
     //renew time quantum
   RunningTask.TQLeft = TimeQuantum;
     done!

//Need context switching
TempTask = ReadyQ.dequeue();
//current task goes to the end of queue
ReadyQ.enqueue( RunningTask );

TempTask.TQLeft = TimeQuantum;
SwitchContext( RunningTask, TempTask );
```