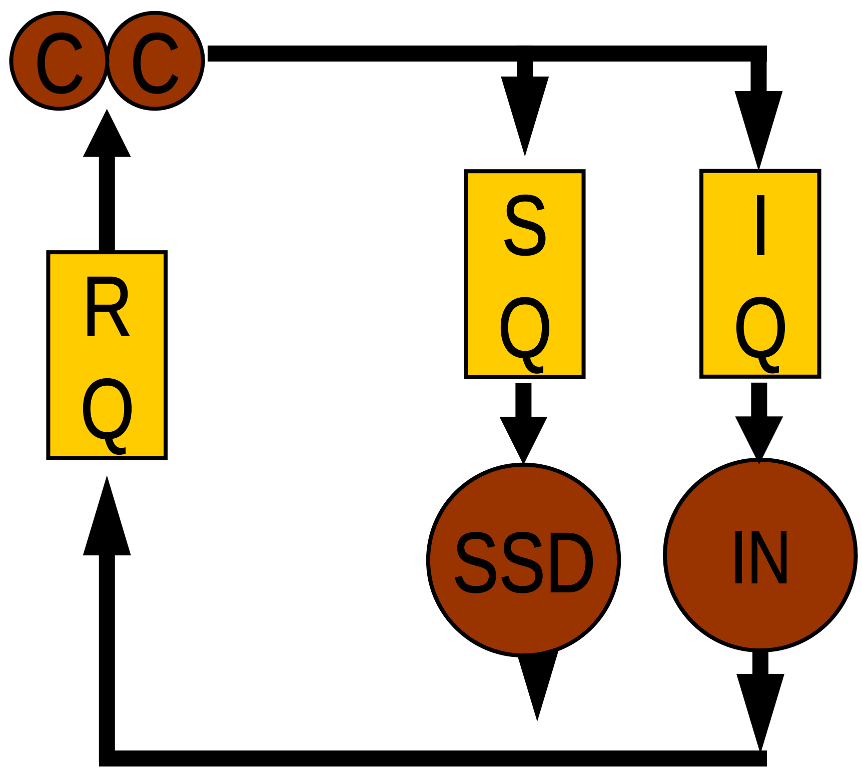
COSC 3360/6310
FIRST ASSIGNMENT

Spring 2018

The model

We have

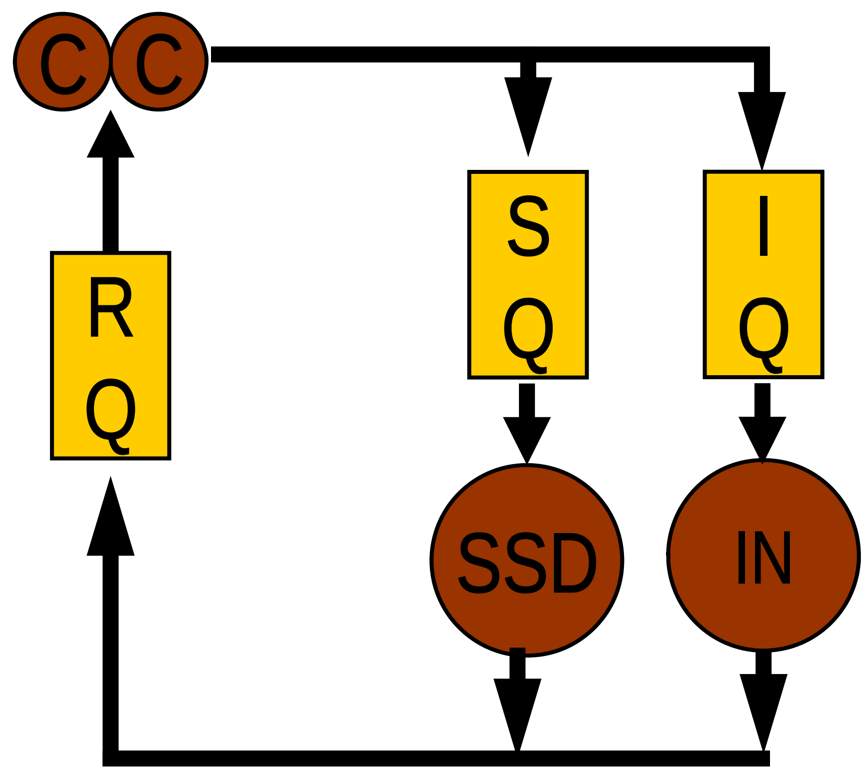
- One multi-core CPU
- One SSD
- One input device
- Three queues
  - CPU queue
    "ready queue"
  - SSD queue
  - Input queue



FIRST EXAMPLE

Start: Process 0 at t = 5ms

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20



Your program will display

- Process 0 starts
  Time = 5 ms

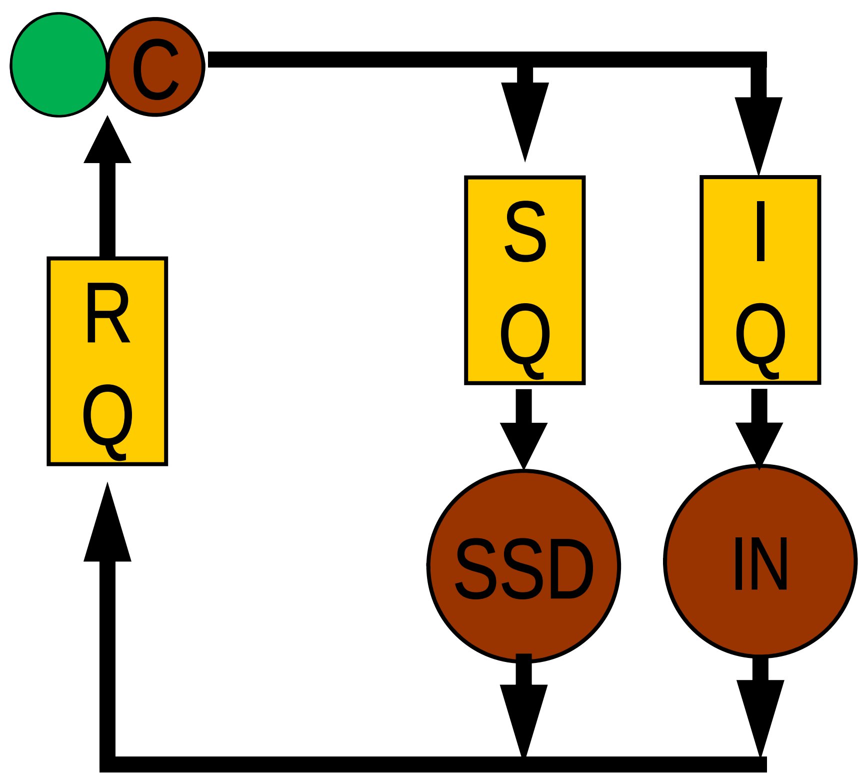
or

- Process 0 starts at t = 5 ms

Process 0 gets a core at t = 5ms

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

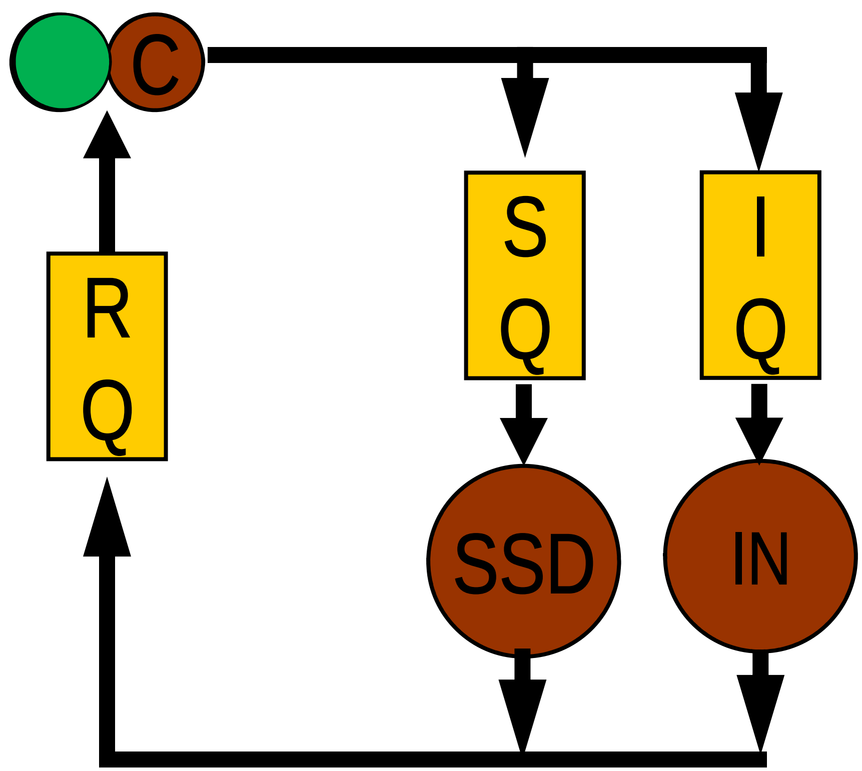
First core busy until t = 105ms



What happens next?

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

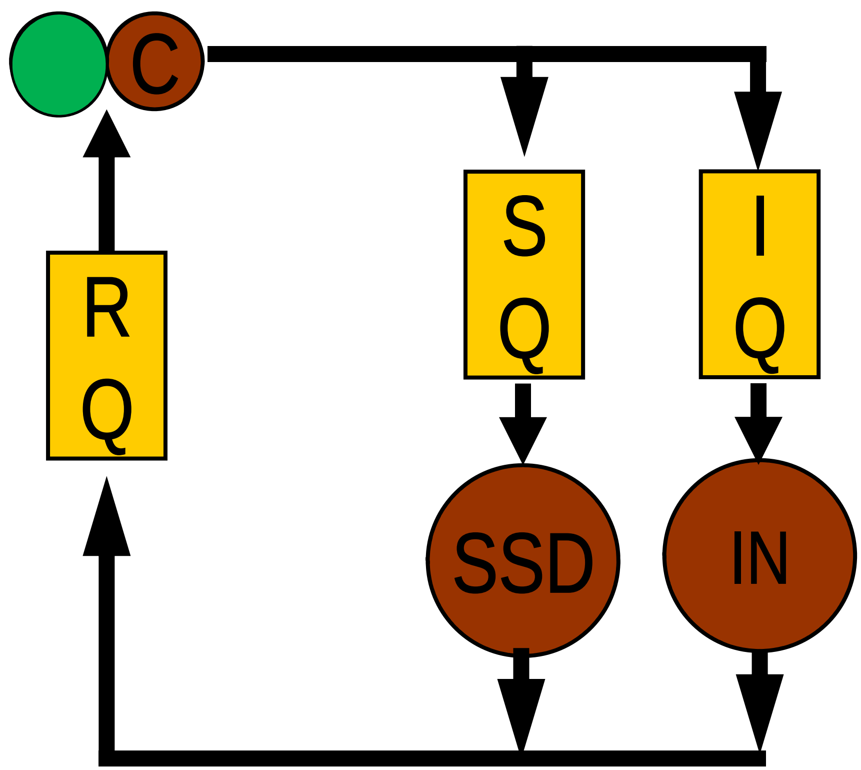
First core busy until t = 105 ms



Process 1 starts at t = 100ms

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

First core busy until t = 105ms



Your program will display

- Process 1 starts
  Time = 100 ms
  Process 0 is RUNNING

or

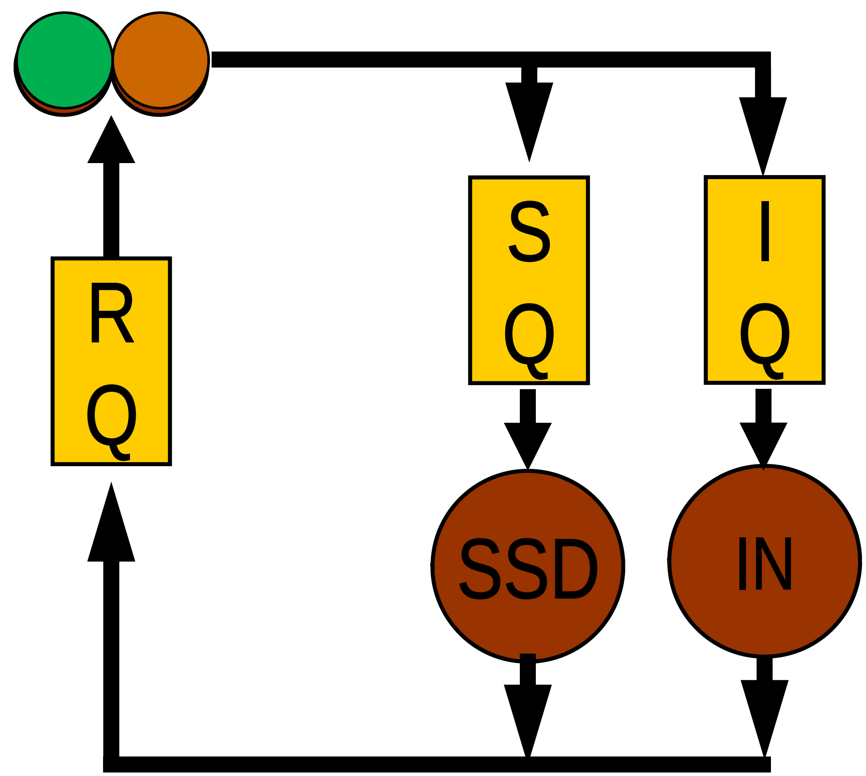
- Process 1 starts at t = 100 ms
  Process 0 is RUNNING

Process 1 gets core at t = 100ms

NCORES 2
NEW 5
<u>CORE 100</u>
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
<u>CORE 20</u>
SSD 0
CORE 20

First core busy until t = 105 ms

Second core busy until t = 120ms

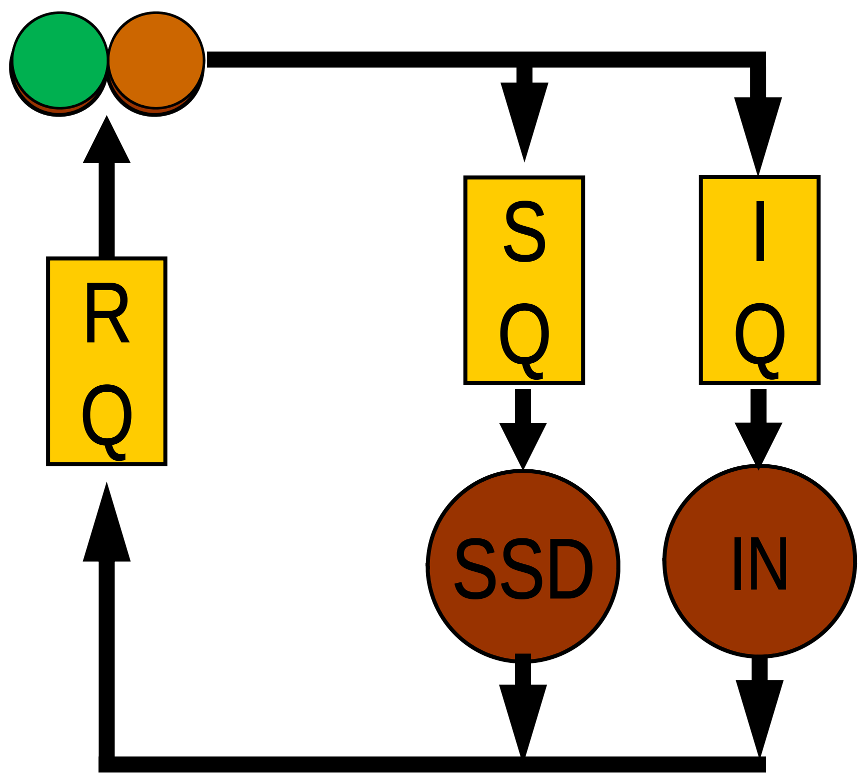


What happens next?

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

First core busy until t = 105 ms

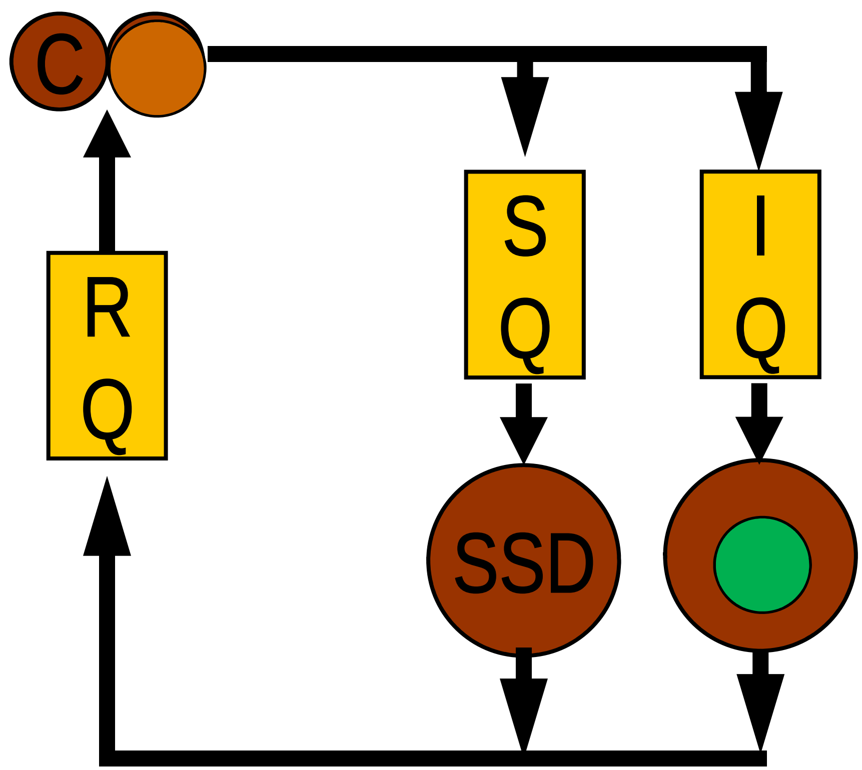
Second core busy until t = 120ms



Process 0 starts I/O at t = 105ms

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

Second core busy until t = 120ms
User busy until t = 5105ms

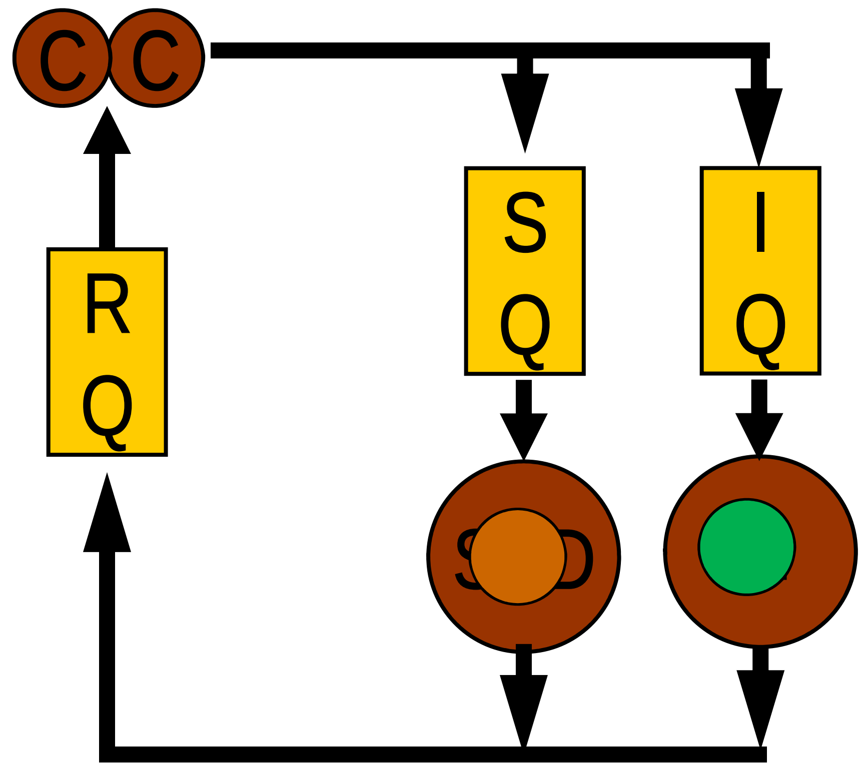


What happens next?

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

Second core busy until t = 120ms

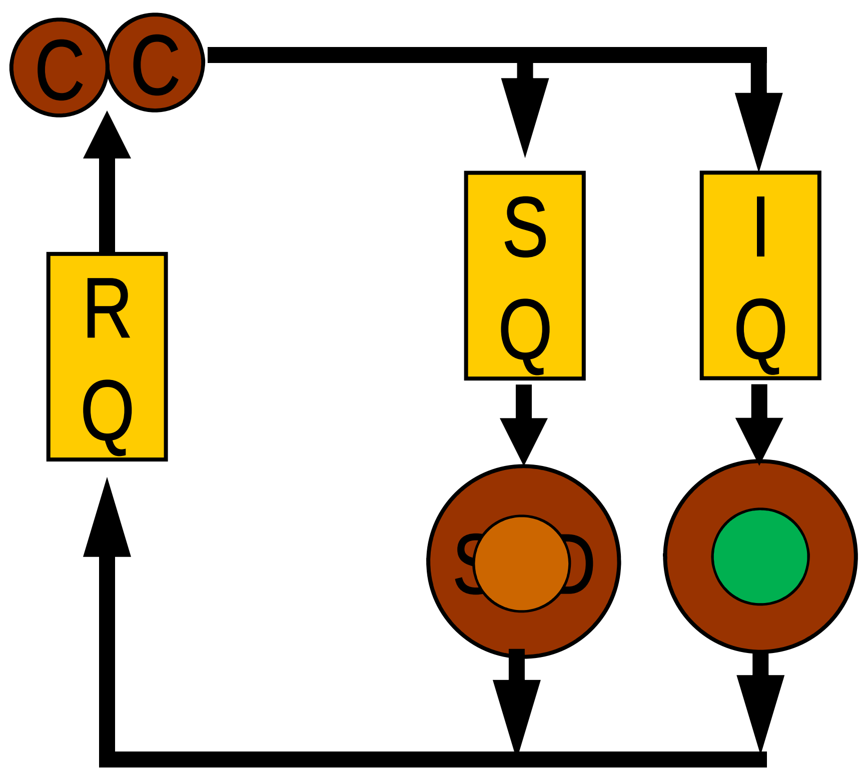
User busy until t = 5105ms



Process 1 gets SSD at t = 120ms

NCORES 2
NEW 5
CORE 100
<u>INPUT 5000</u>
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
<u>SSD 0</u>
CORE 20

SSD busy until t = 120 ms
User busy until t = 5105 ms

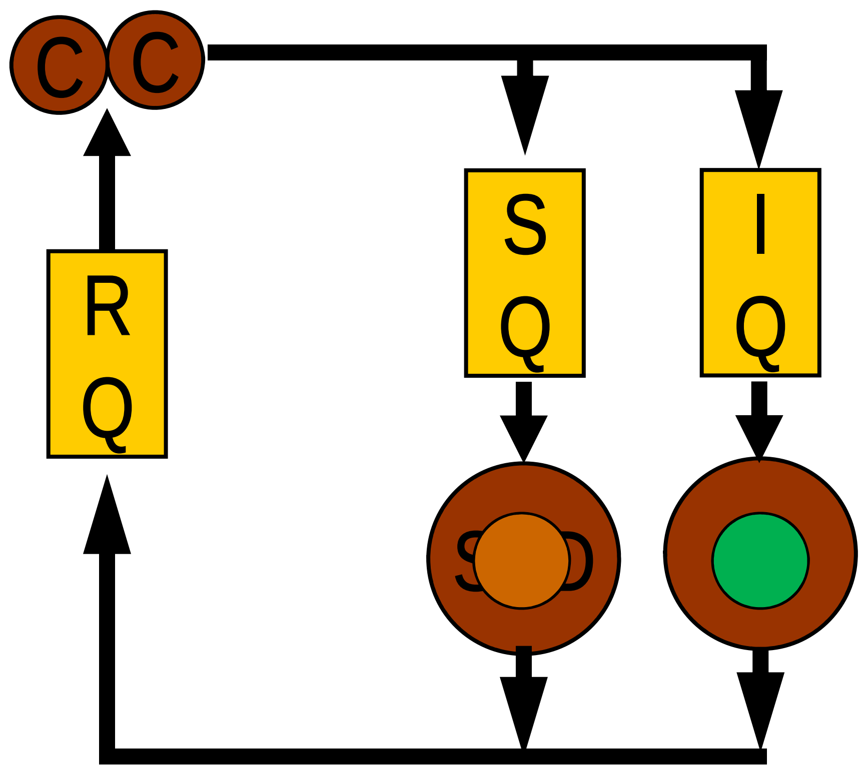


What is next?

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

SSD busy until t = 120 ms

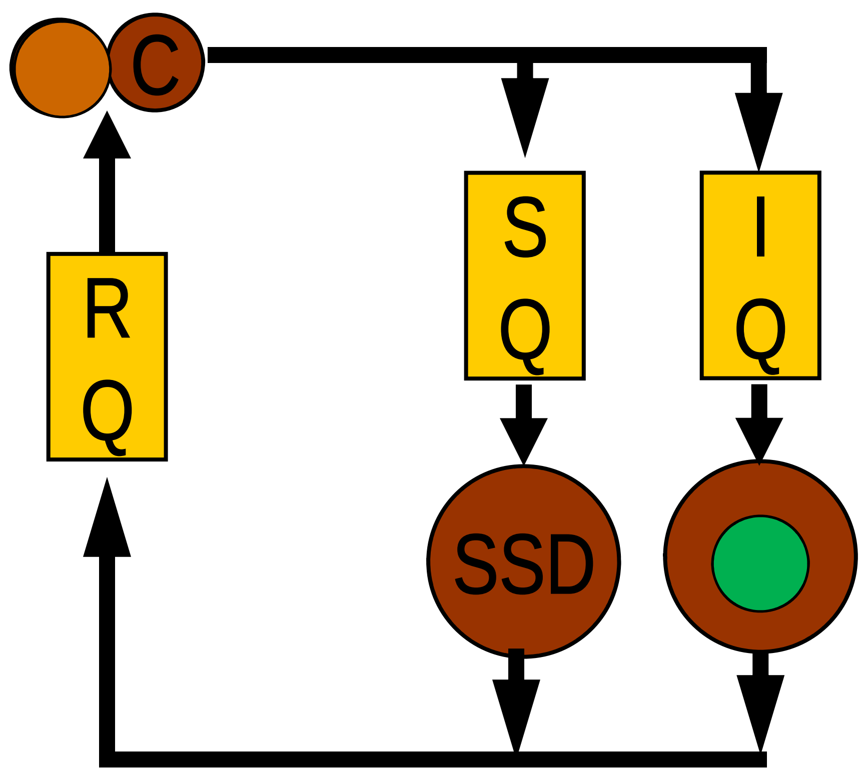
User busy until t = 5105 ms



Process 1 gets core at t = 120ms

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

First core busy until t = 140ms
User busy until t = 5105ms

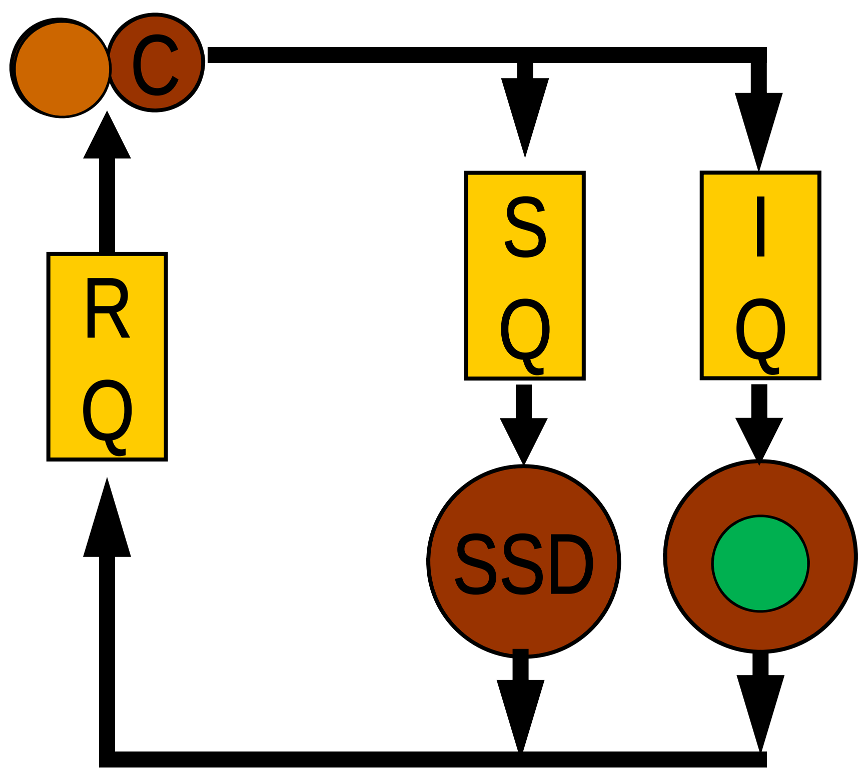


What happens next?

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

First core busy until t = 140 ms

Process 0 busy until t = 5105 ms

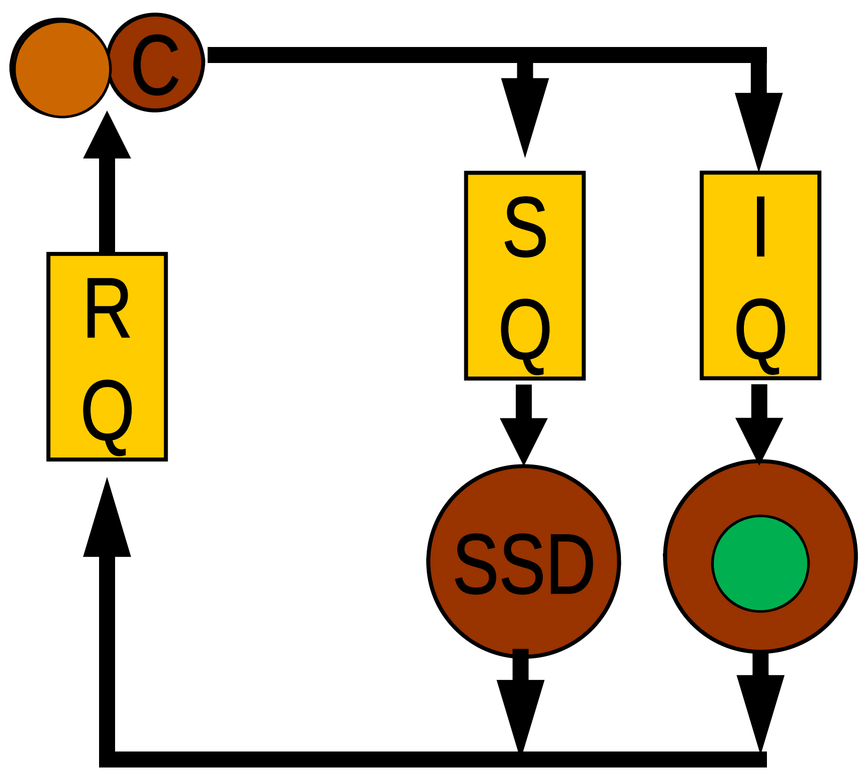


What happens next?

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

First core busy until t = 140 ms

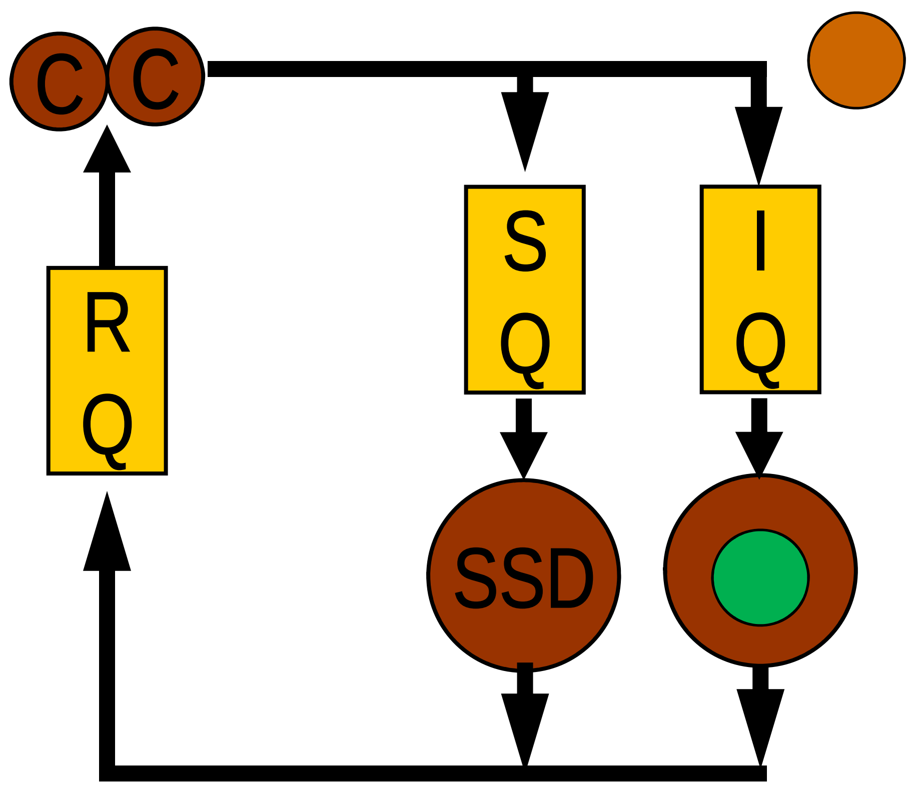
Process 0 busy until t = 5105 ms



Process 1 ends at t = 140ms

User busy until t = 5105 ms

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20



Your program will display

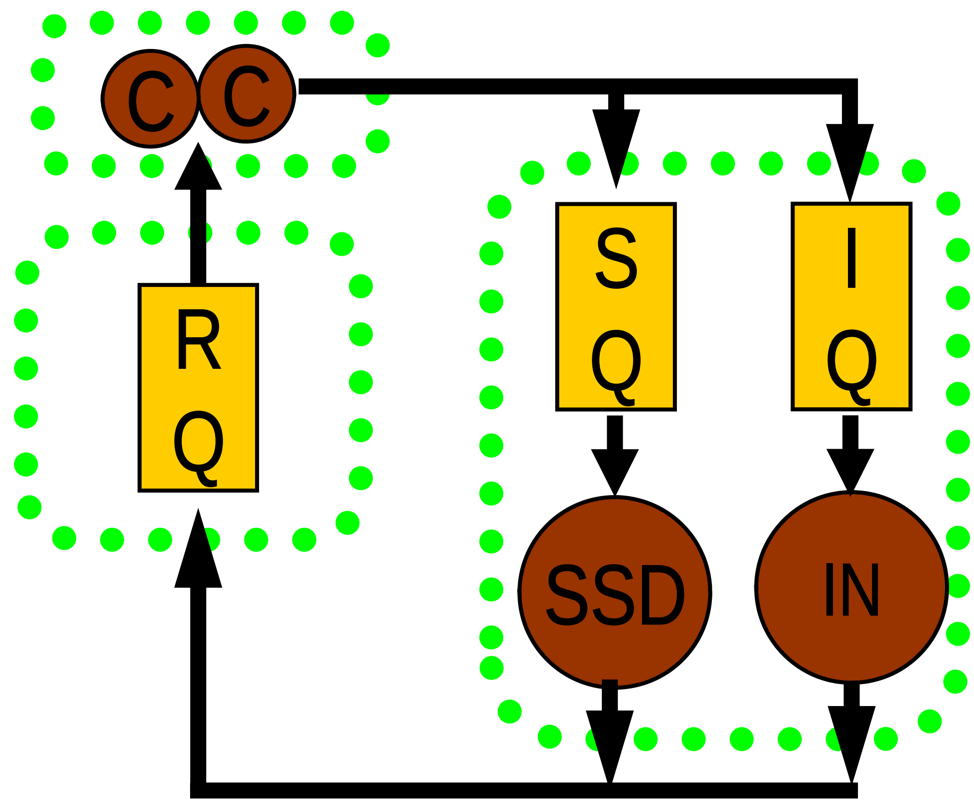
- Time = 140 ms
  Process 1 terminates
  Process 0 is BLOCKED

or

- Process 1 terminates at t = 140 ms
  Process 0 is BLOCKED

The three process states

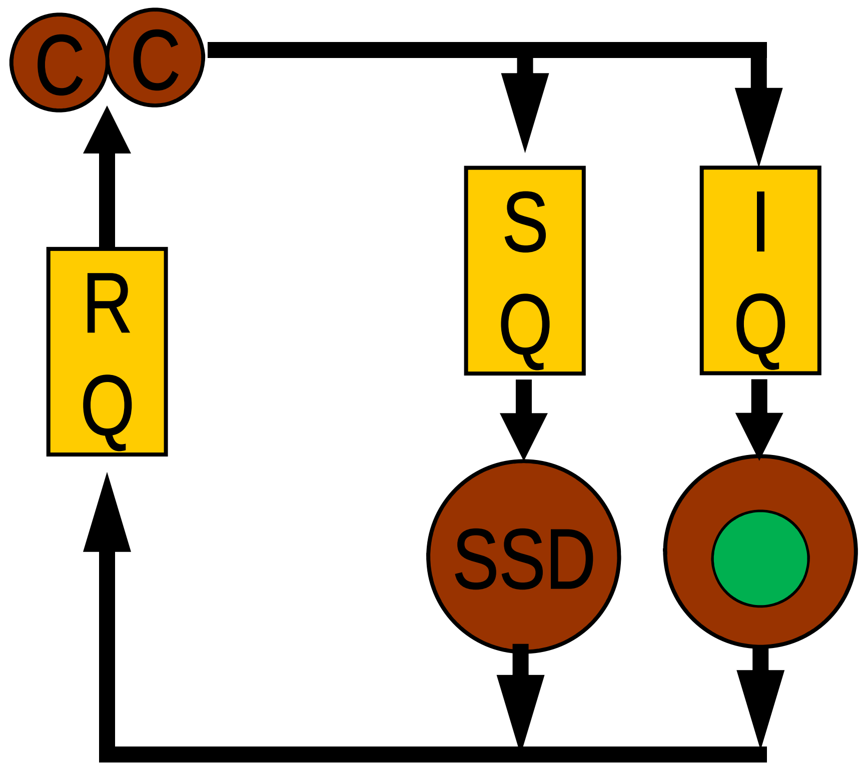
- **RUNNING** means executing on a core
- **READY** means waiting for a core
- **BLOCKED** means being neither RUNNING or READY



What happens next?

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

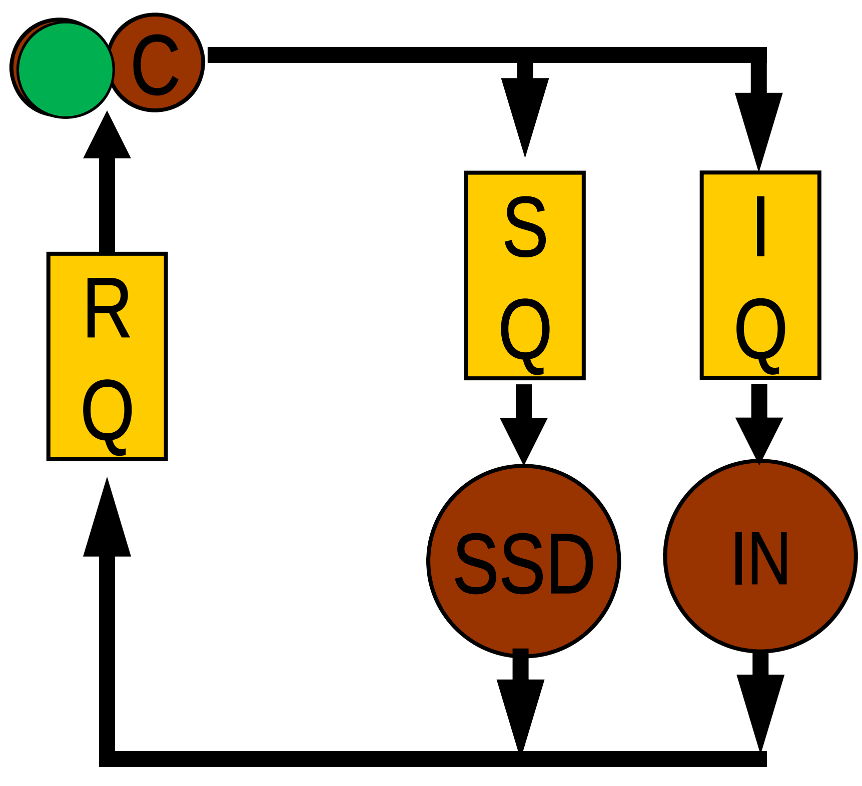
User is busy until t = 5105 ms



Process 0 gets core at t=5105ms

NCORES 2
NEW 5
CORE 100
INPUT 5000
<u>CORE 80</u>
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

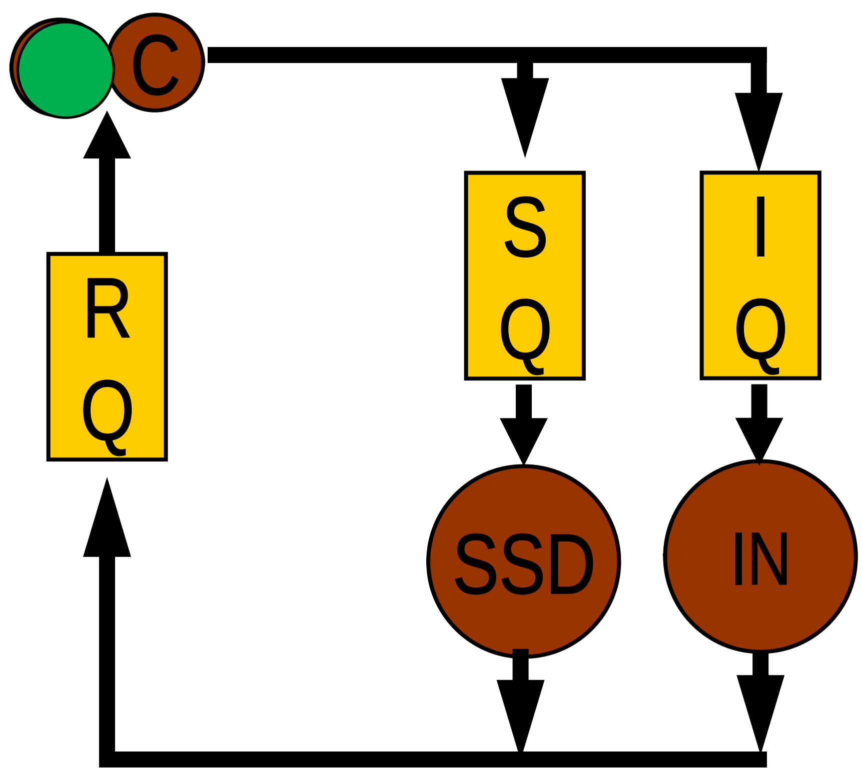
Core busy until t = 5185 ms



What happens next?

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

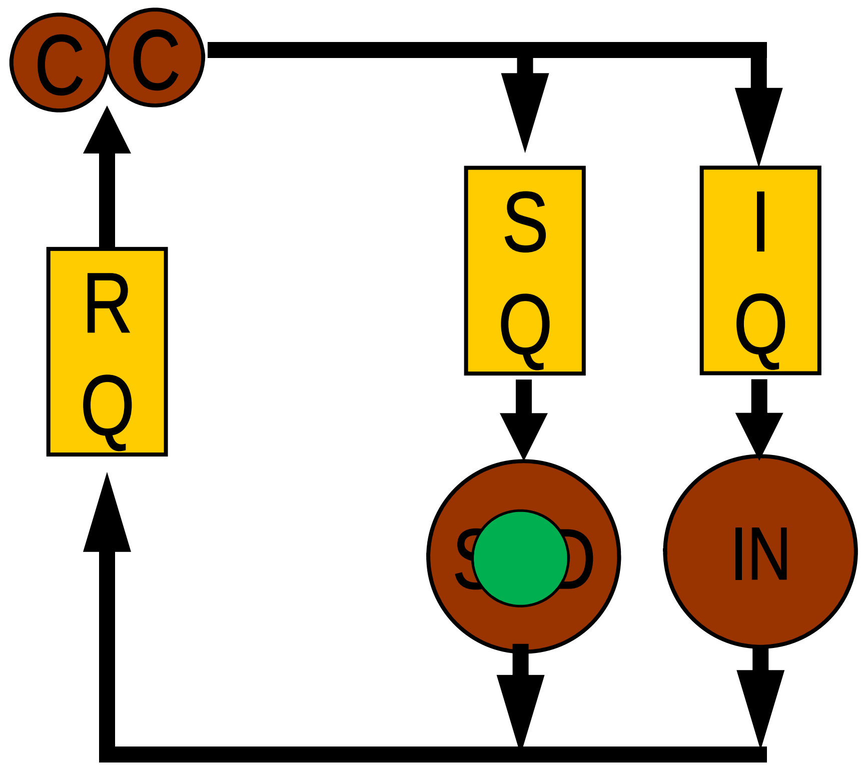
Core busy until t = 5185 ms



Process 0 gets SSD at t=5185ms

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

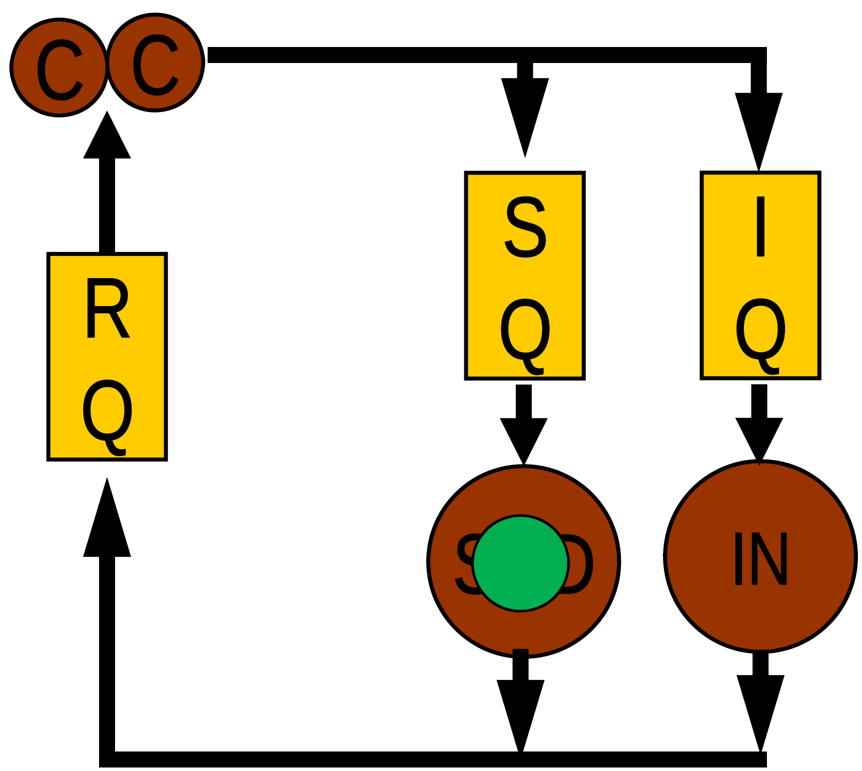
SSD is busy until t = 5186 ms



What happens next?

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

SSD busy until t = 5186 ms



Process 0 gets core at t=5186ms

NCORES 2

NEW 5

CORE 100

INPUT 5000

CORE 80

SSD 1

CORE 30

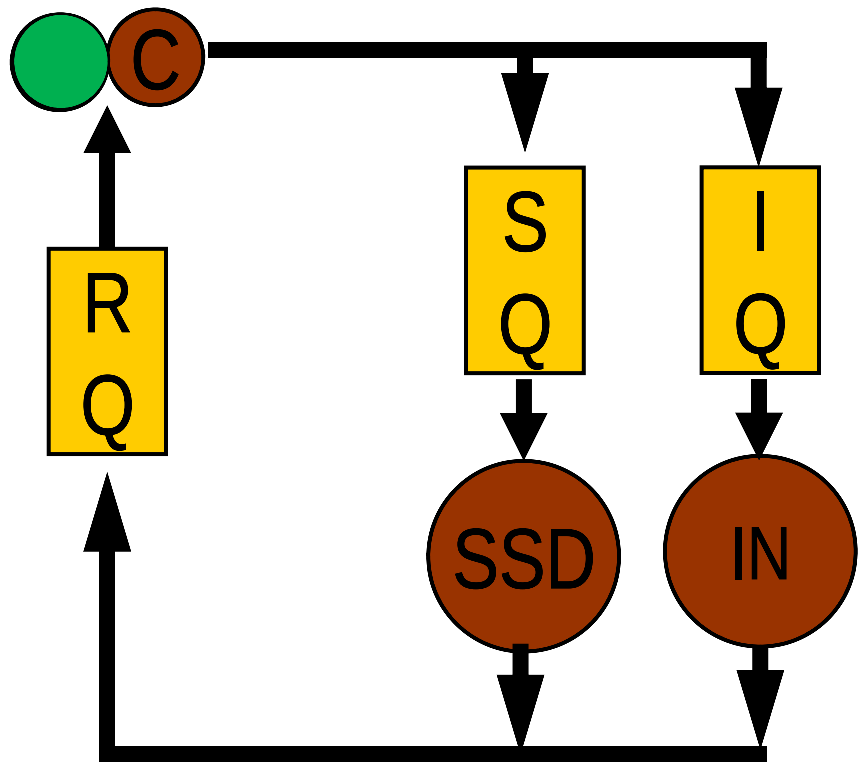
NEW 100

CORE 20

SSD 0

CORE 20

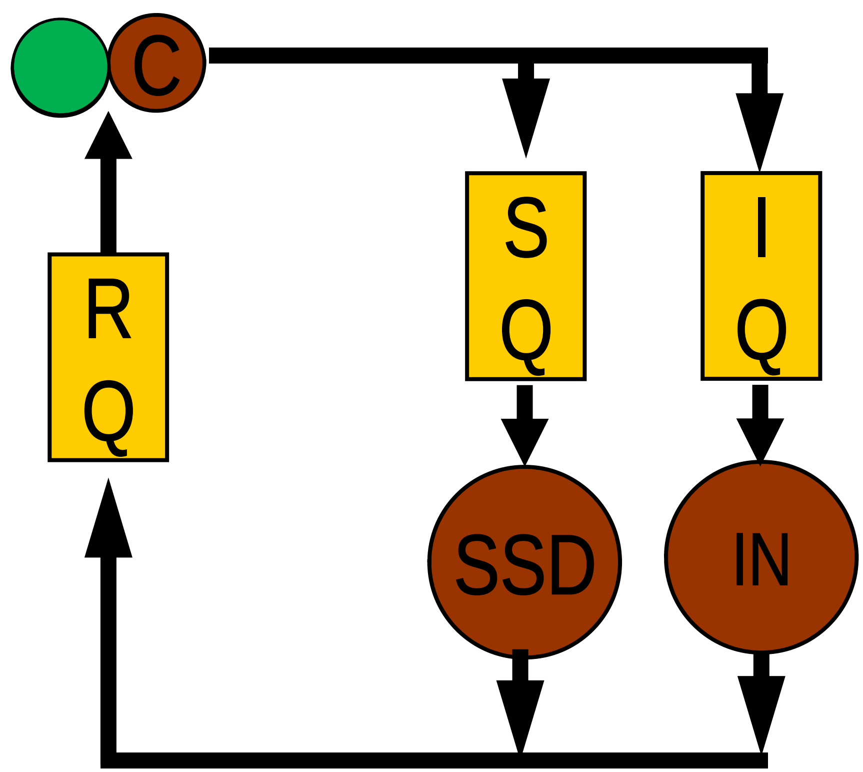
A core is busy until t = 5216ms



What happens next?

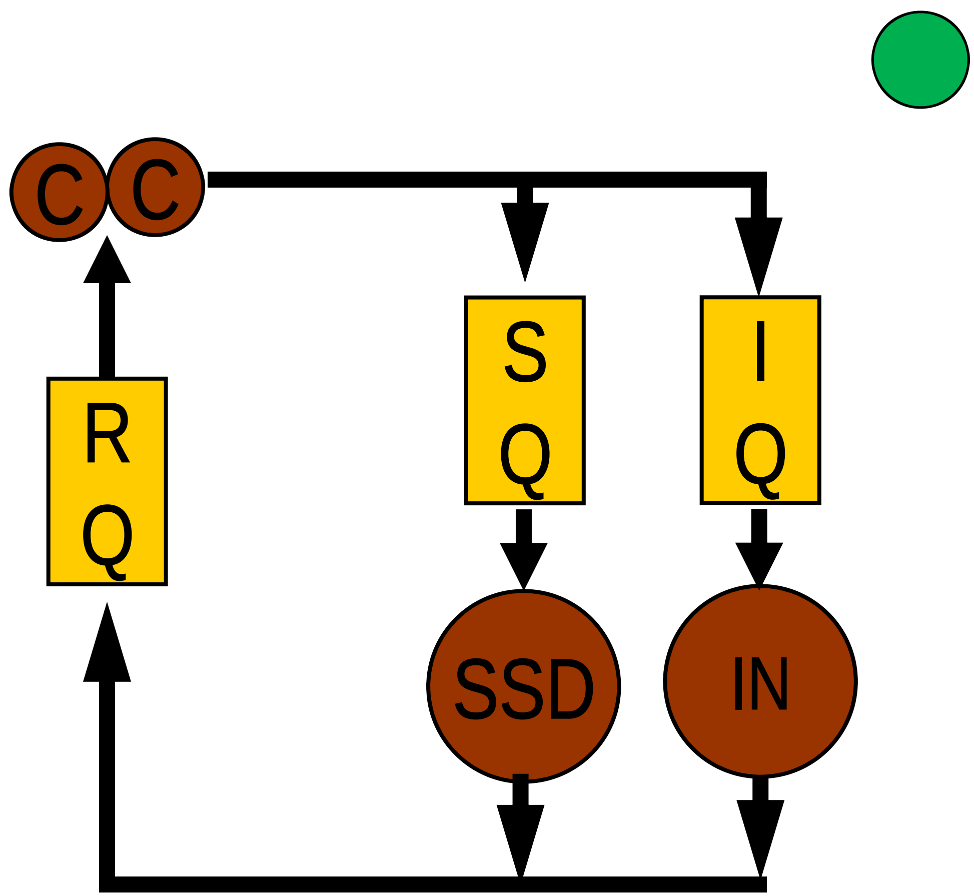
NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
CORE 30
NEW 100
CORE 20
SSD 0
CORE 20

A core is busy until t = 5216ms



Process 0 terminates

NCORES 2
NEW 5
CORE 100
INPUT 5000
CORE 80
SSD 1
<u>CORE 30</u>
NEW 100
CORE 20
SSD 0
CORE 20



Your program will display (I)

- Process 0 terminates
  Time = 5216 ms

or

- Process 0 terminates at t = 5216 ms

Your program will display (II)

- **SUMMARY:**
  Number of processes that completed: 2
  Total number of SSD accesses: 2
  Average SSD access time: 0.5 ms
  Total elapsed time: 5216 ms
  Core utilization: 4.79 percent
  SSD utilization: 0.02 percent

How to compute core utilization

- Keep track of total time for all RUN requests:
  - 100 + 80 + 30 + 20 + 20 = 250 ms
- Divide by elapsed time:
  - 250/5216 = **0.0479** (rounded)

- Since NCORES = 2 , the maximum CORE utilization is 2.0

How to compute SSD utilization

- Keep track of total time for all SS requests:
  - 0 + 1 = 1 ms
- Divide by elapsed time:
  - 1/5216 = **0.0002** (rounded)

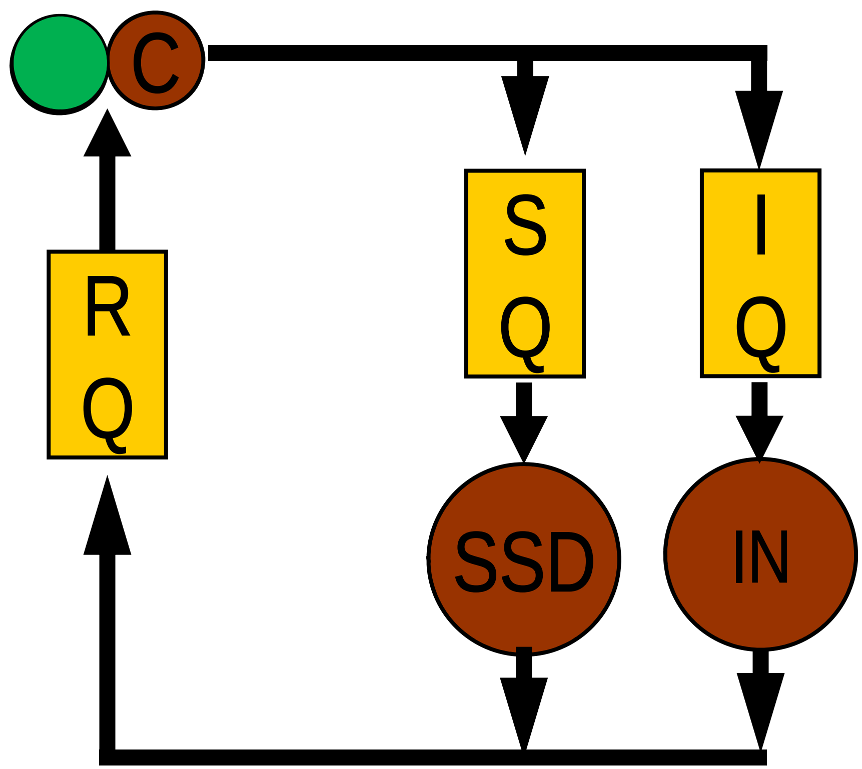
- SSD utilization is normally computed for each storage device, so it will never exceed 1.0

INTRODUCING CONTENTION

P0 gets a core at t = 5ms

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

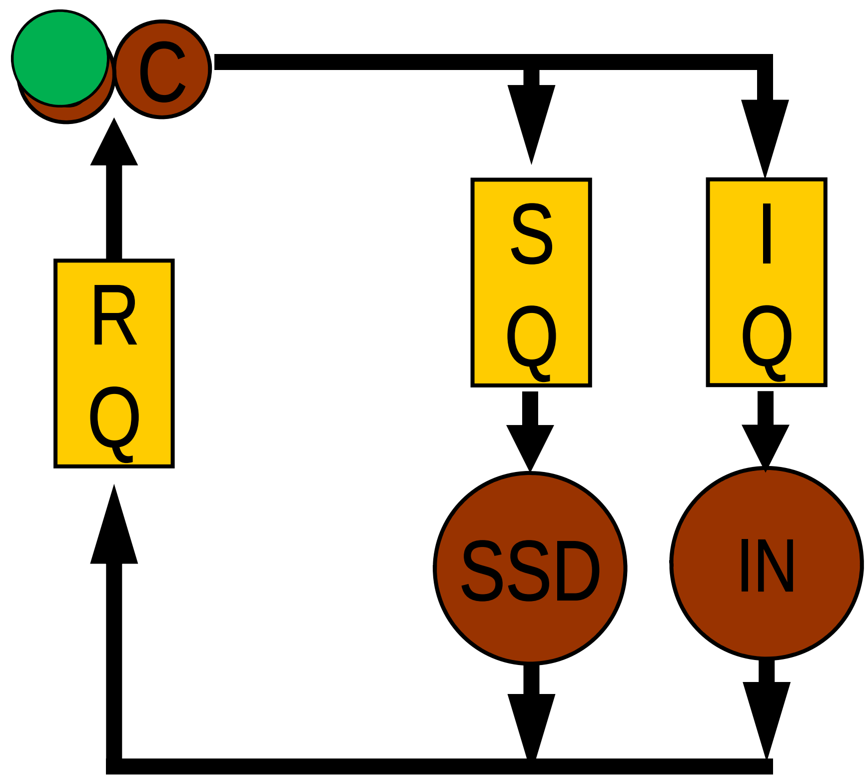
P0 holds a core until t = 105ms



What happens next?

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

P0 holds a core until t = 105 ms

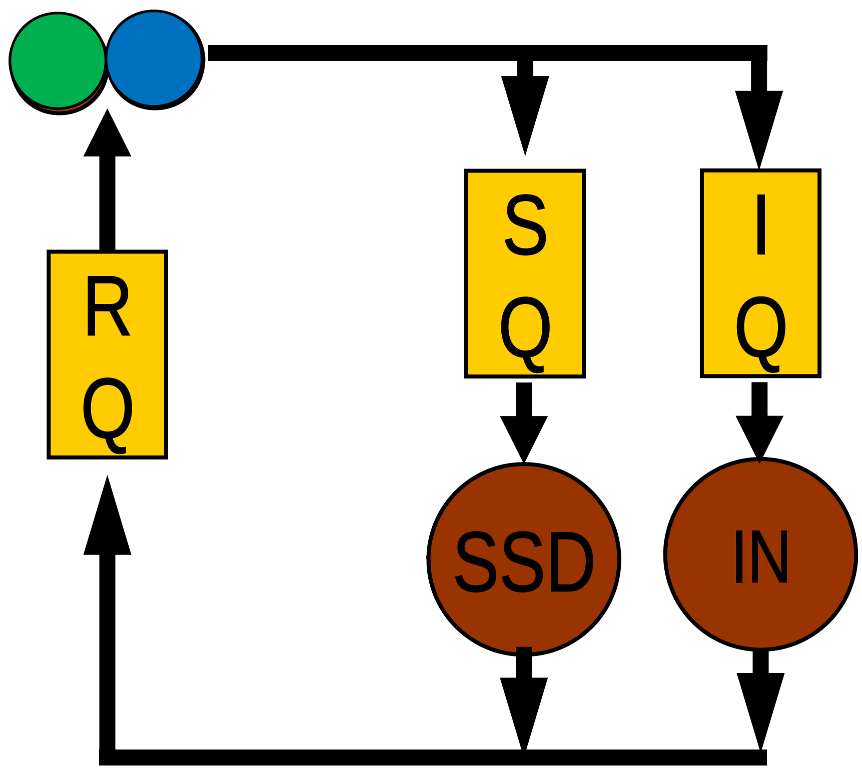


P1 gets a core at t = 20ms

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

P0 holds a core until t = 105ms

P1 holds a core until t = 70ms

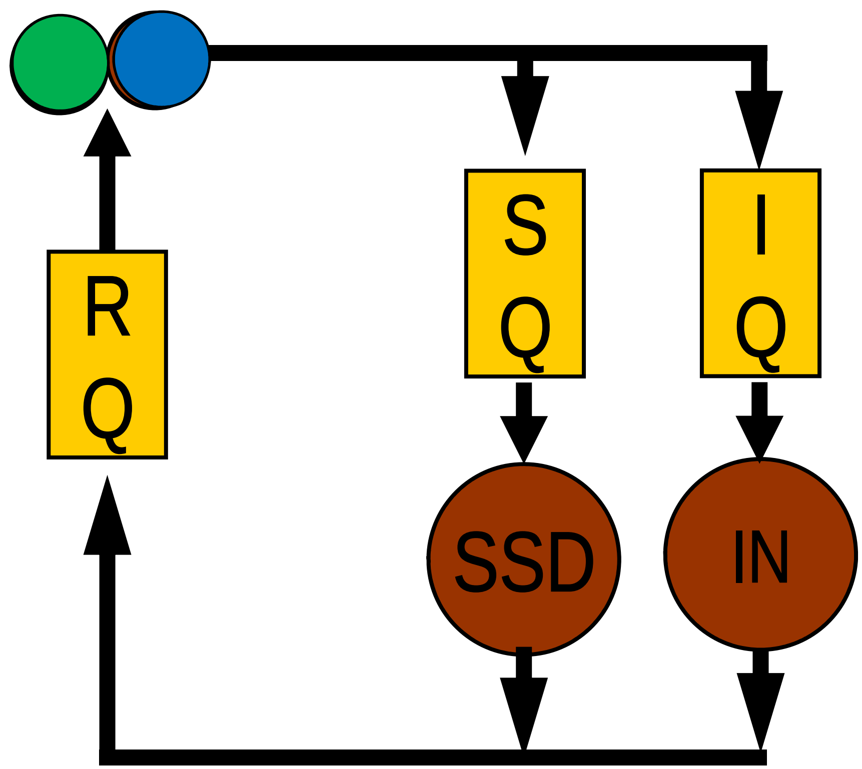


What happens next?

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

P0 holds a core until t = 105ms

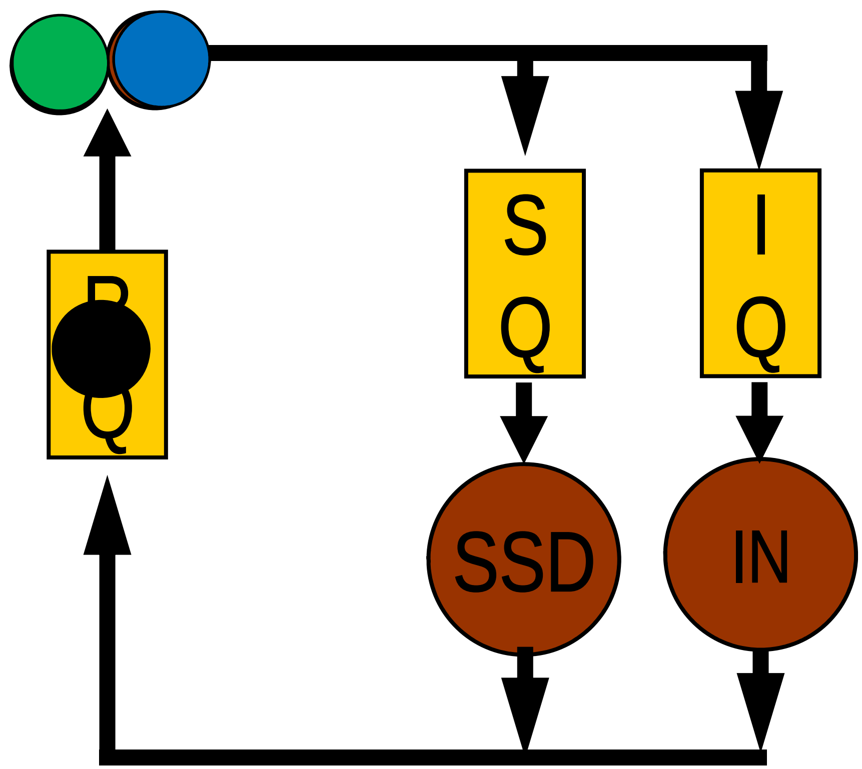
P1 holds a core until t = 70ms



P2 waits for a core

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

First core busy until t = 105ms
Second core busy until t = 70ms

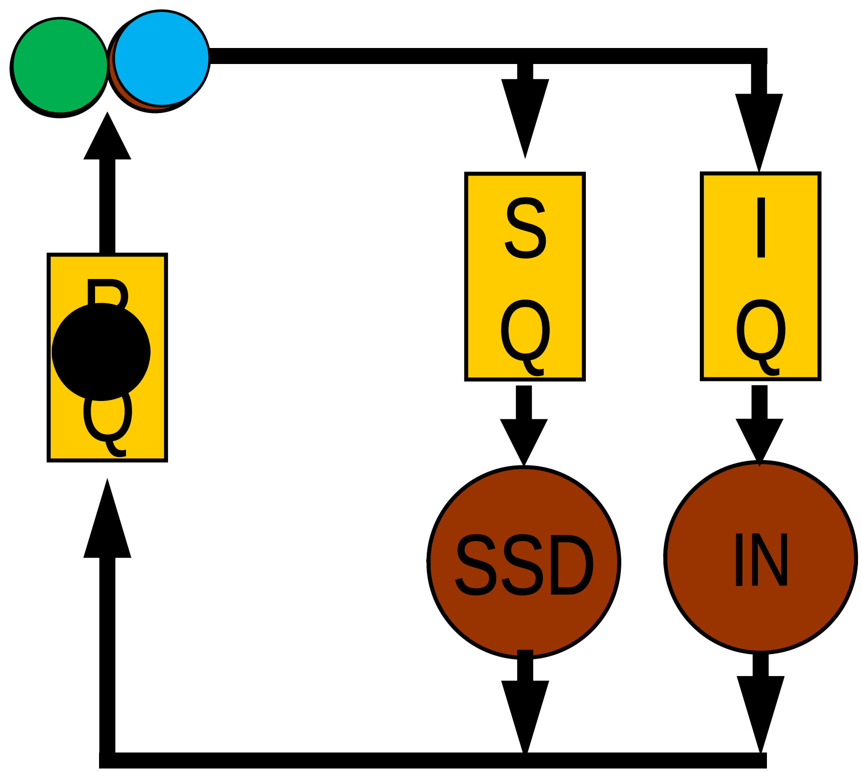


What happens next?

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

P0 holds a core until t = 105ms

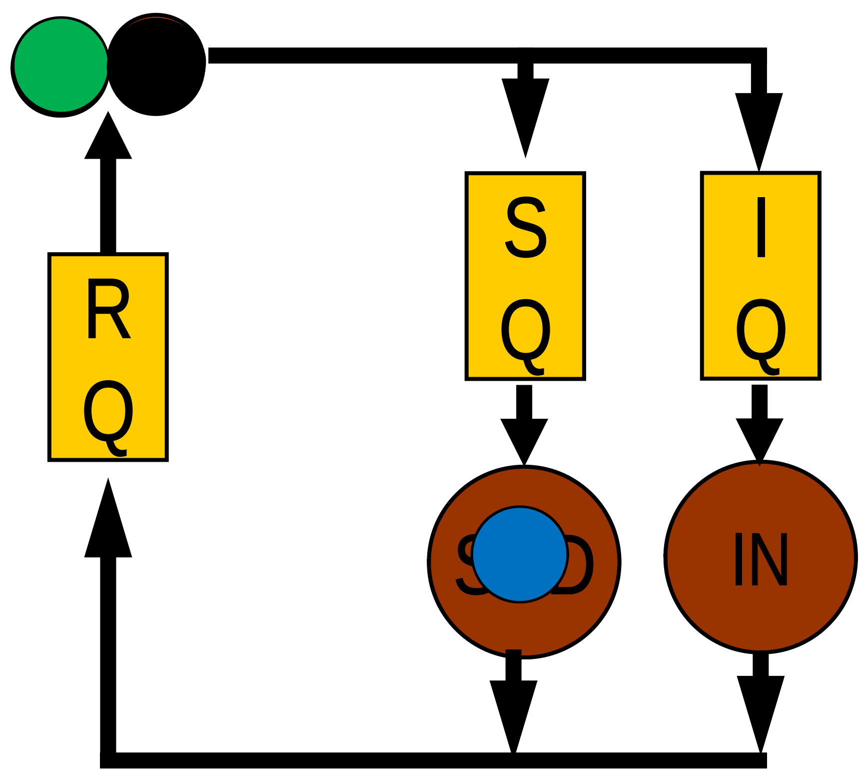
P1 holds a core until t = 70ms



P1 gets SSD at t = 70ms
P2 get core

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

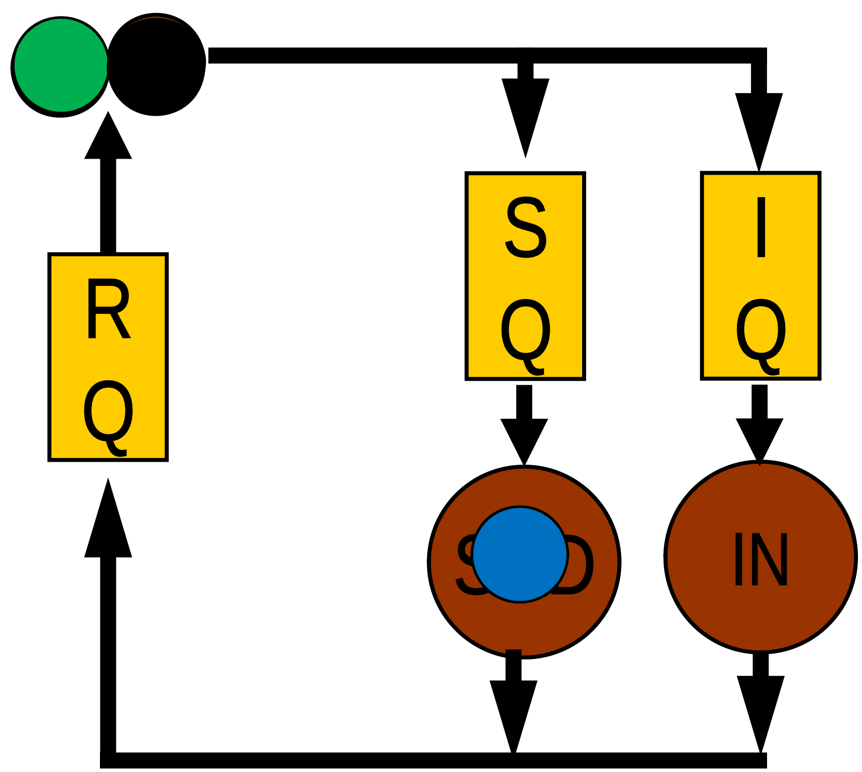
P0 holds a core until t = 105ms
P2 holds a core until t = 90ms
P1 holds SSD until t =70ms



What happens next?

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

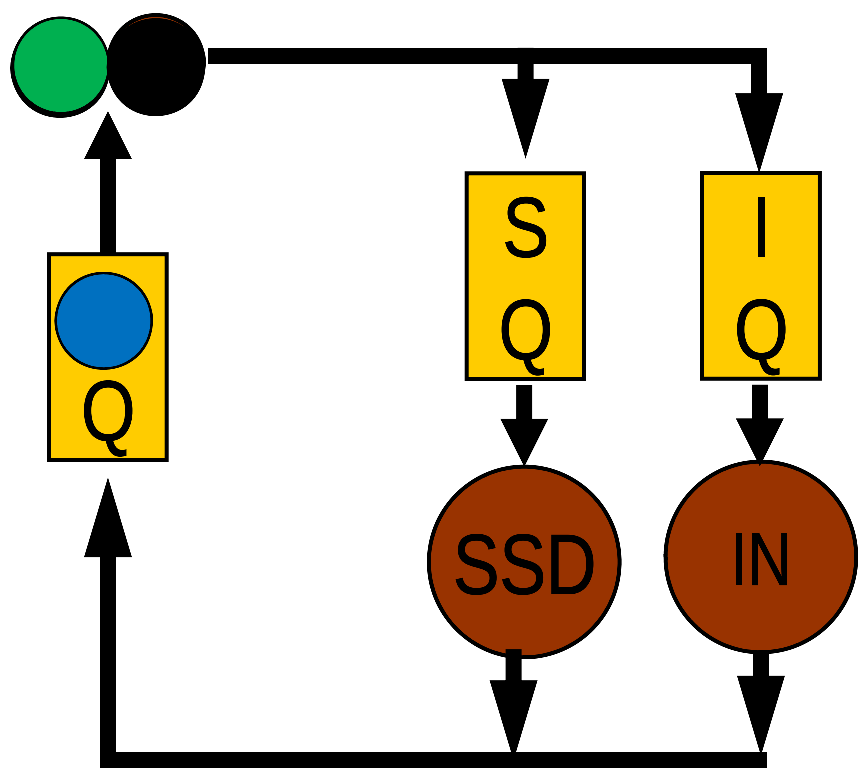
P0 holds a core until t = 105ms
P2 holds a core until t = 90ms
P1 holds SSD until t =70ms



Process 1 waits for a core

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

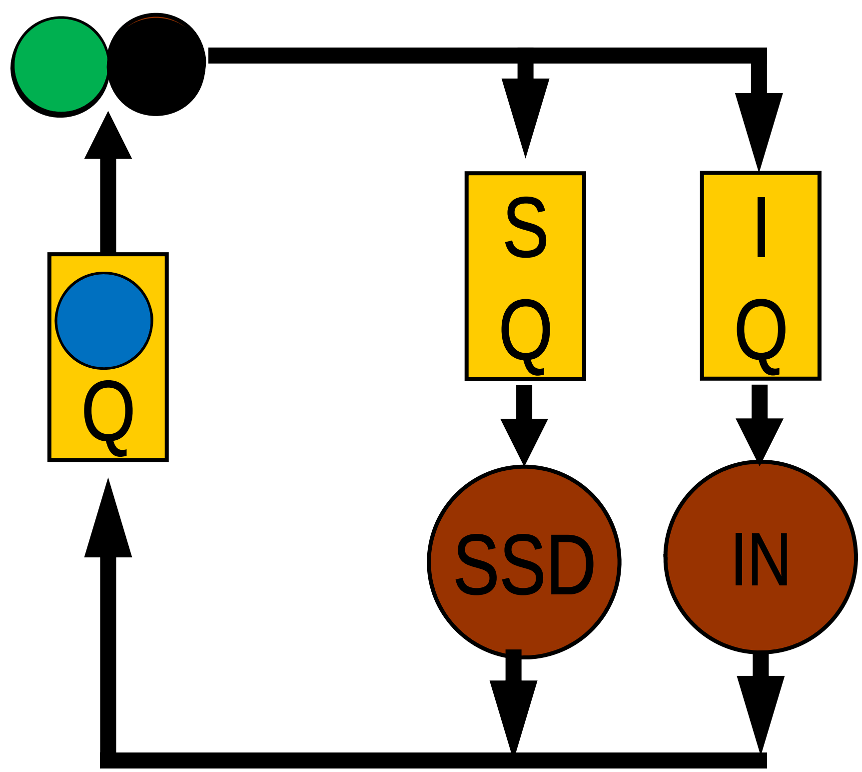
P0 holds a core until t = 105ms
P2 holds a core until t = 90ms



What happens next?

NCORES 2
NEW 5
<u>CORE 100</u>
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
<u>CORE 50</u>
NEW 30
<u>CORE 20</u>
SSD 0
CORE 20

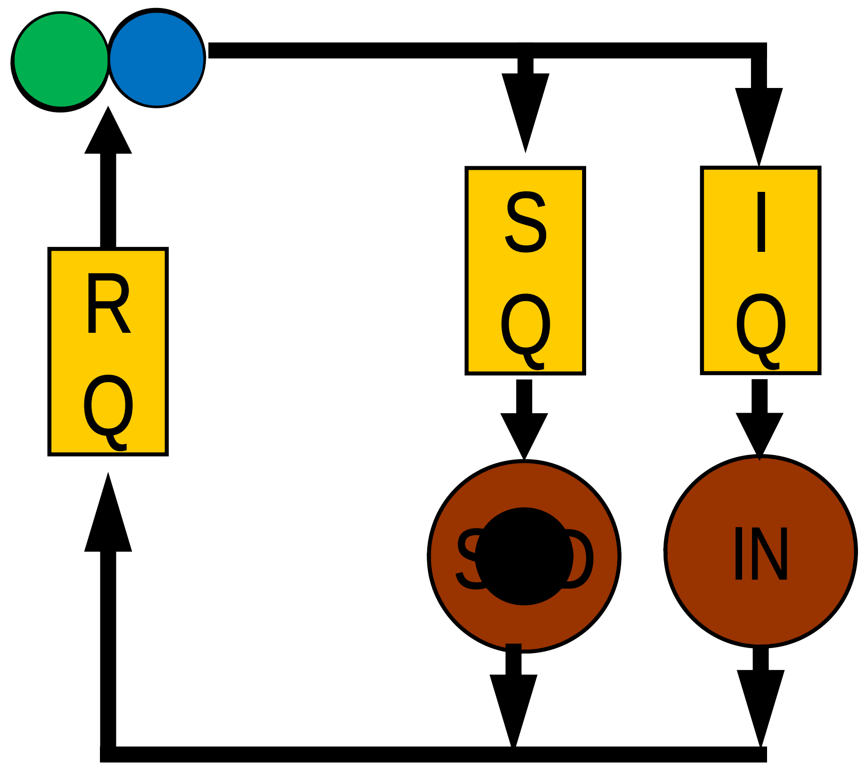
P0 holds a core until t = 105ms
P2 holds a core until t = 90ms



P2 gets SSD at time t = 90ms P1 gets a core

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

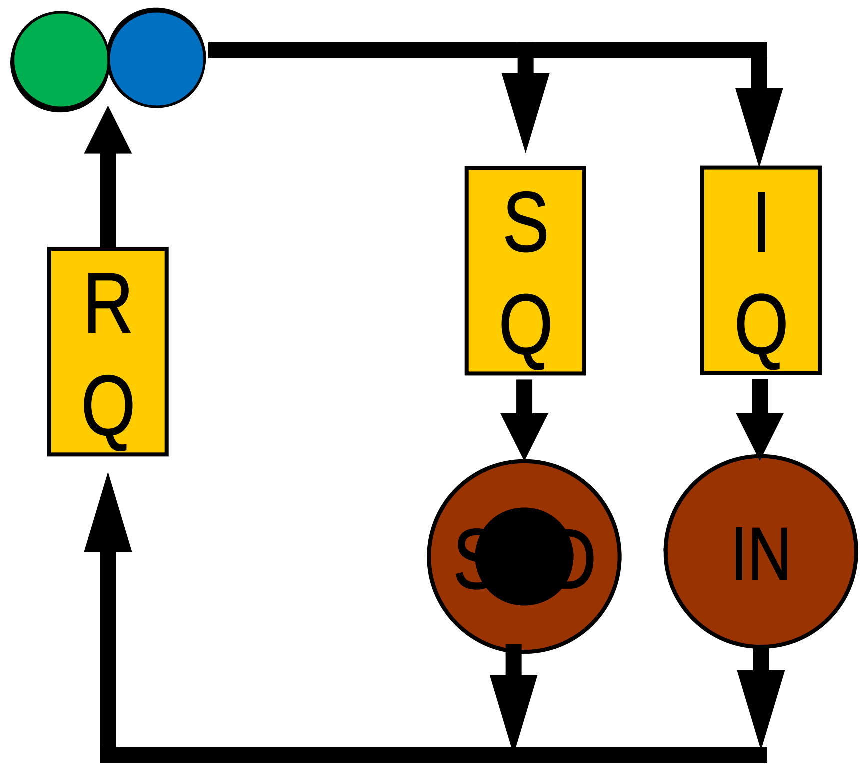
P0 holds a core until t = 105ms
P1 holds a core until t = 140ms
P2 holds SSD until t = 90ms



What happens next?

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

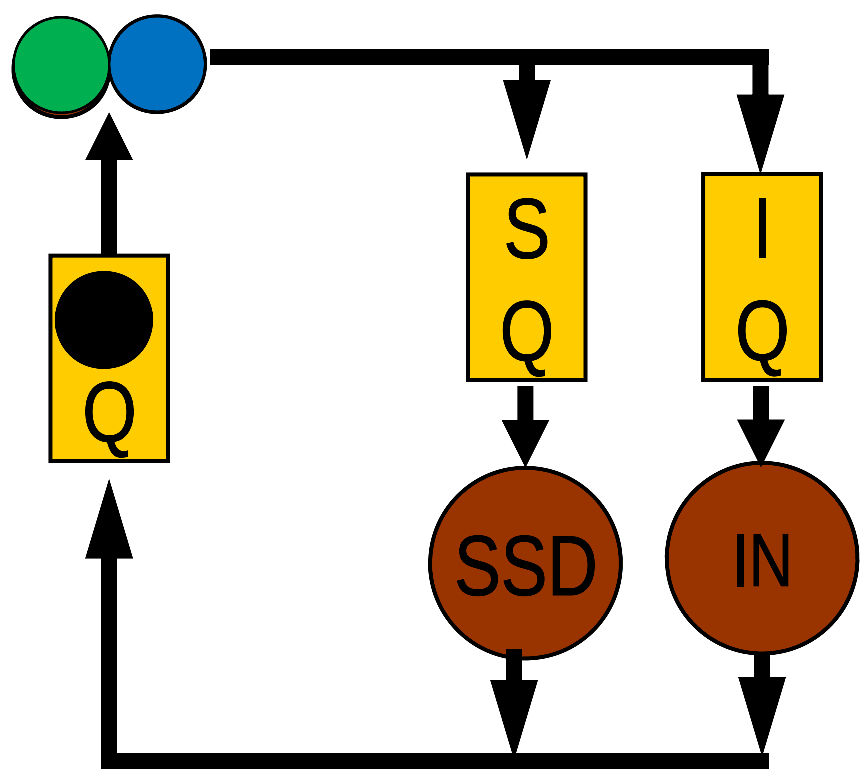
P0 holds a core until t = 105ms
P1 holds a core until t = 140ms
P2 holds SSD until t = 90ms



P2 waits for a core

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

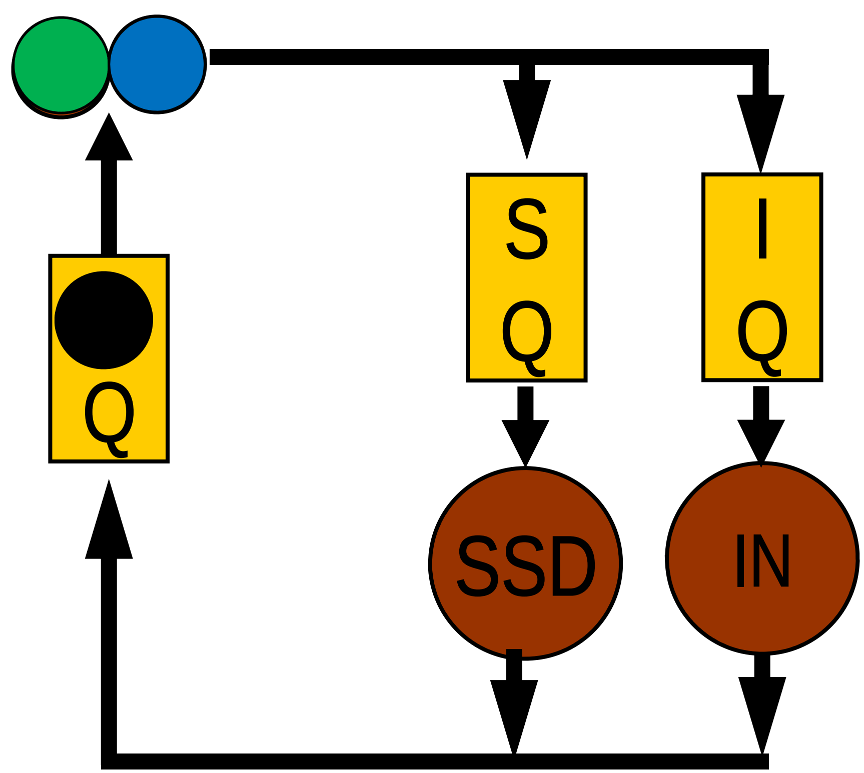
P0 holds a core until t = 105ms
P1 holds a core until t = 140ms



What happens next?

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

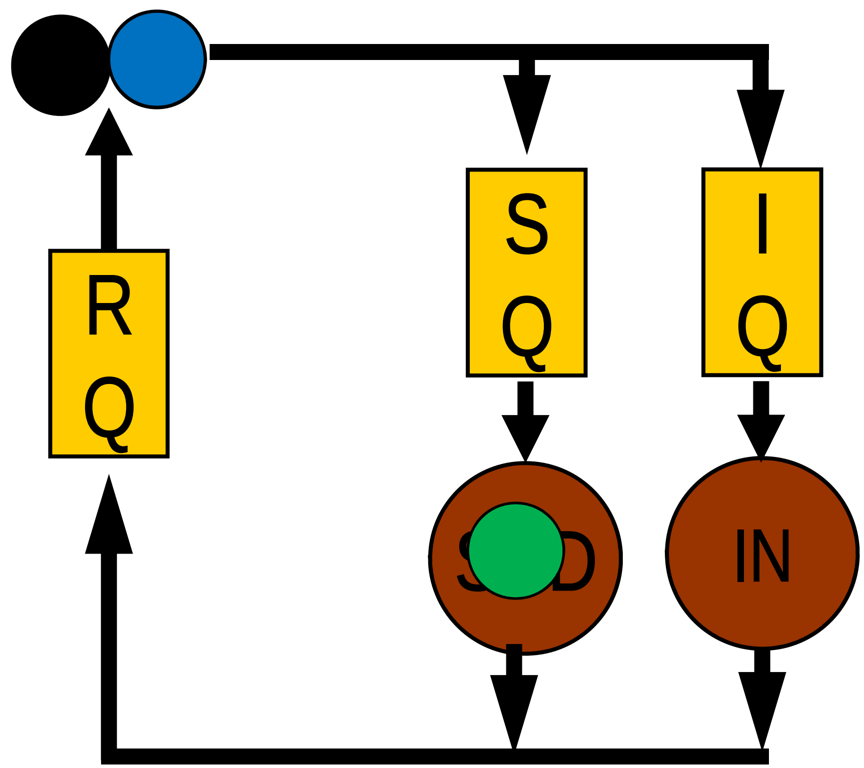
P0 holds a core until t = 105ms
P1 holds a core until t = 140ms



P0 gets SSD at t = 105ms
P2 gets a core

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

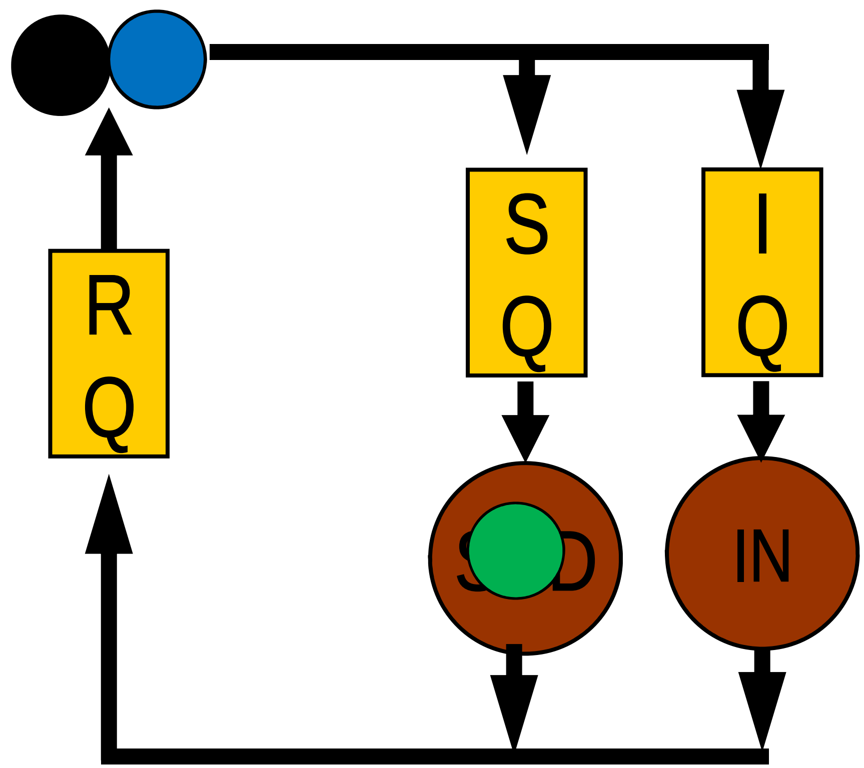
P1 holds a core until t = 140ms
P2 holds a core until t = 125ms
P0 holds SSD until t = 105ms



What happens next?

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

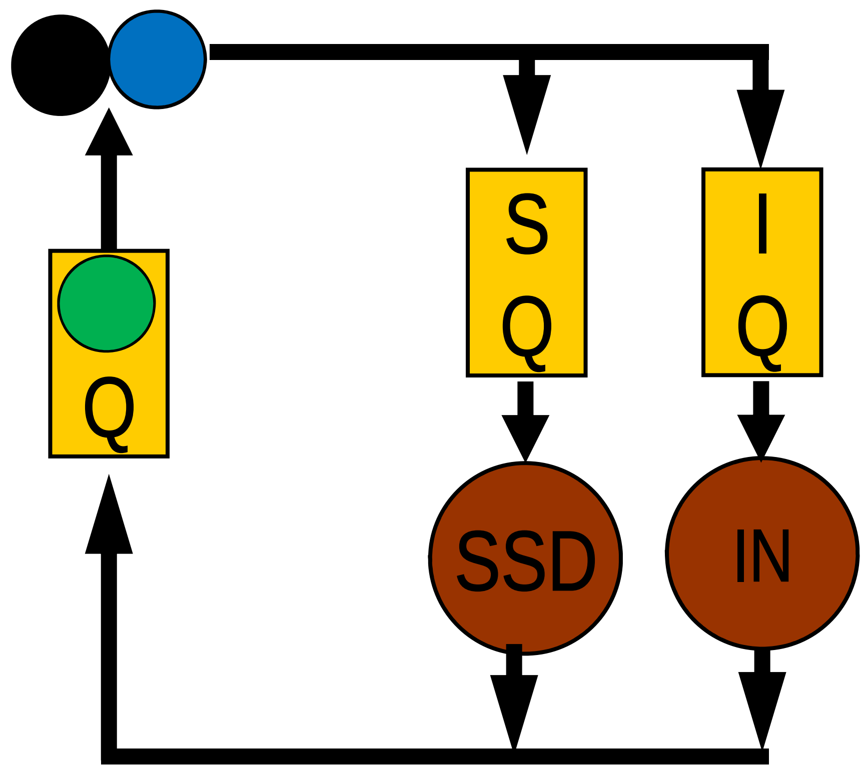
P1 holds a core until t = 140ms
P2 holds a core until t = 125ms
P0 holds SSD until t = 105ms



P0 waits for a core

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

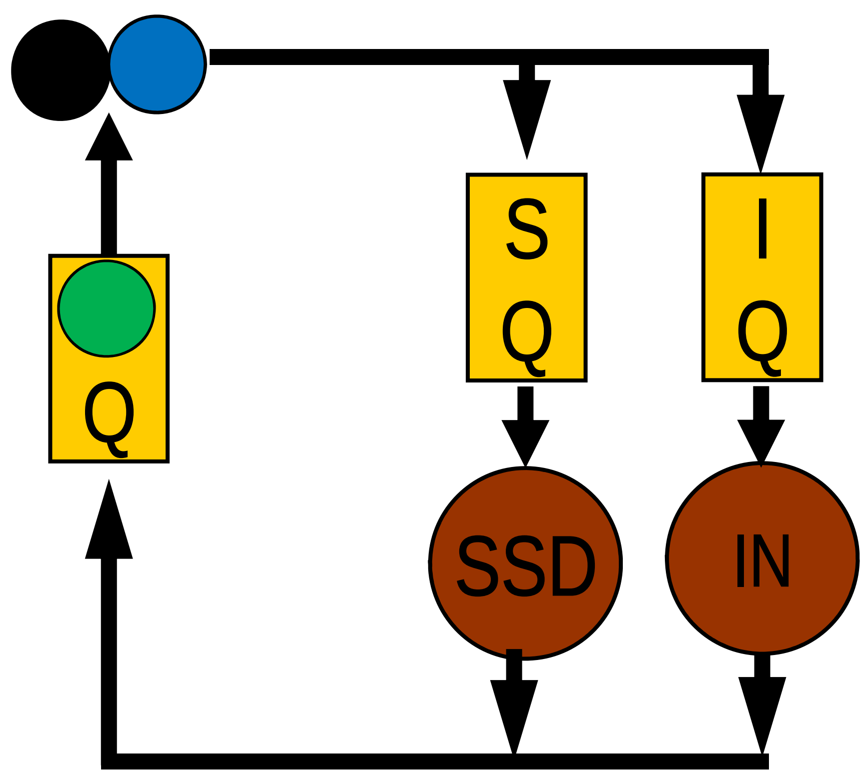
P1 holds a core until t = 140ms
P2 holds a core until t = 125ms



What happens next?

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

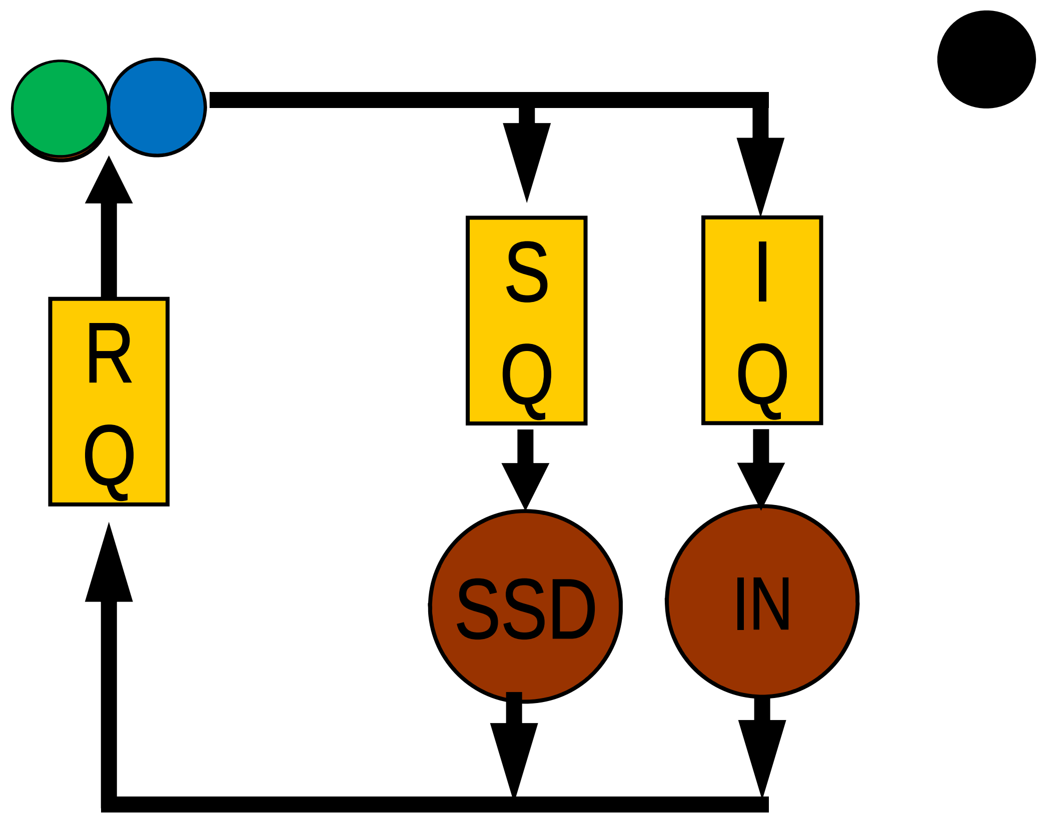
P1 holds a core until t = 140ms
P2 holds a core until t = 125ms



P2 terminates
P0 gets a core at time t = 125ms

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

P1 holds a core until t = 140ms
P0 holds a core until t = 155ms



Your program will display

- Time = 125 ms
  Process 2 terminates
  Process 0 is RUNNING
  Process 1 is RUNNING

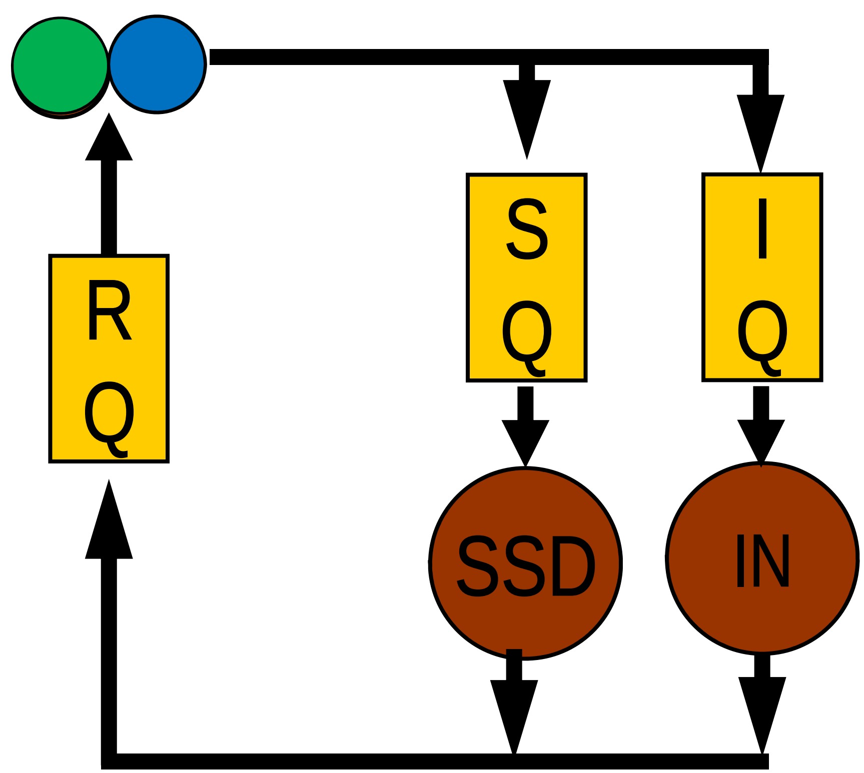
or

- Process 2 terminates at t = 125 ms
  Process 0 is RUNNING
  Process 1 is RUNNING

What happens next?

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

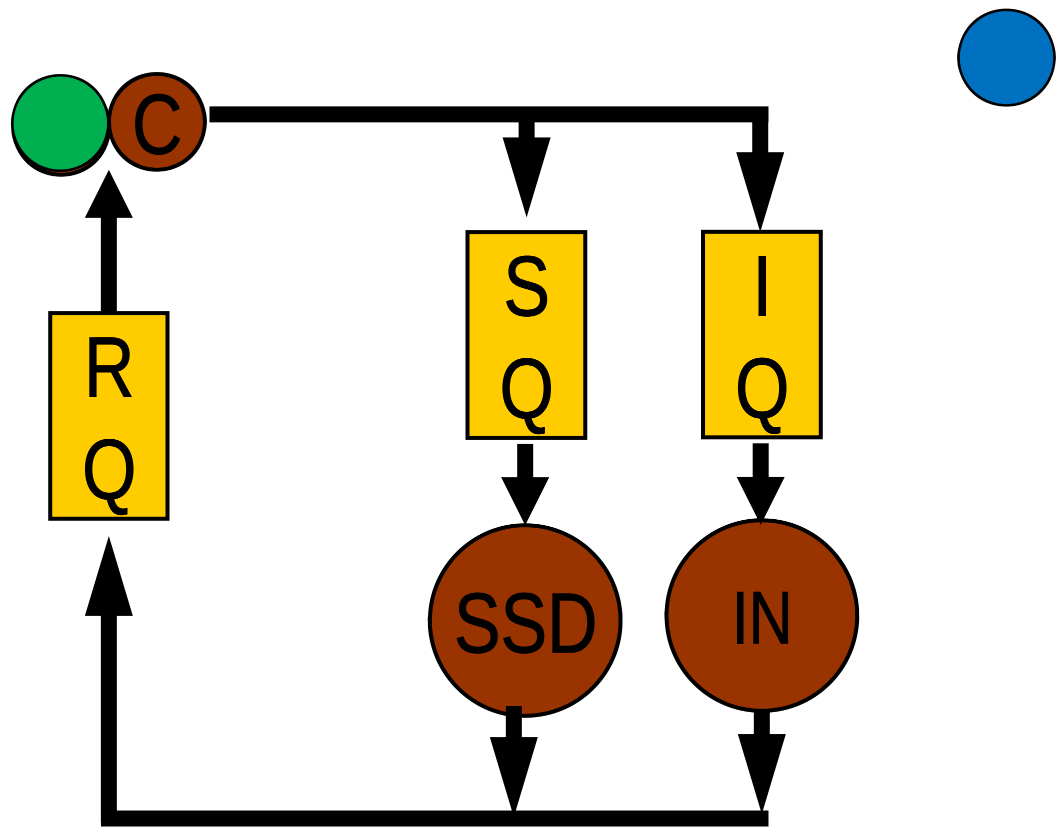
P1 holds a core until t = 140ms
P0 holds a core until t = 155ms



P1 terminates

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

P0 holds a core until t = 155ms



Your program will display

- Time = 140ms
  Process 1 terminates
  Process 0 is RUNNING

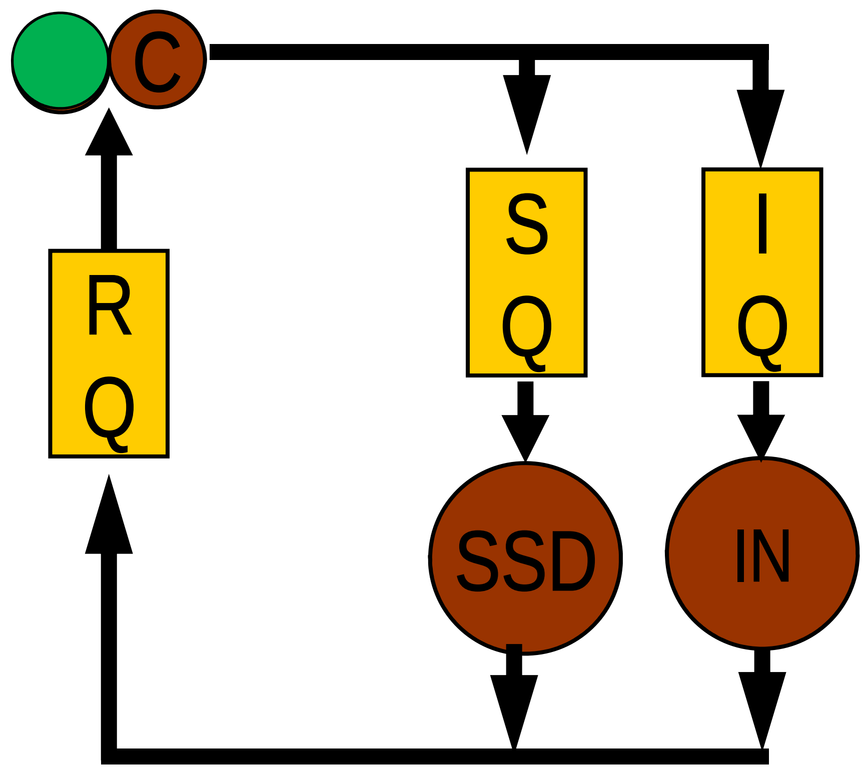
or

- Process 1 terminates at t = 140 ms
  Process 0 is RUNNING

What happens next?

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

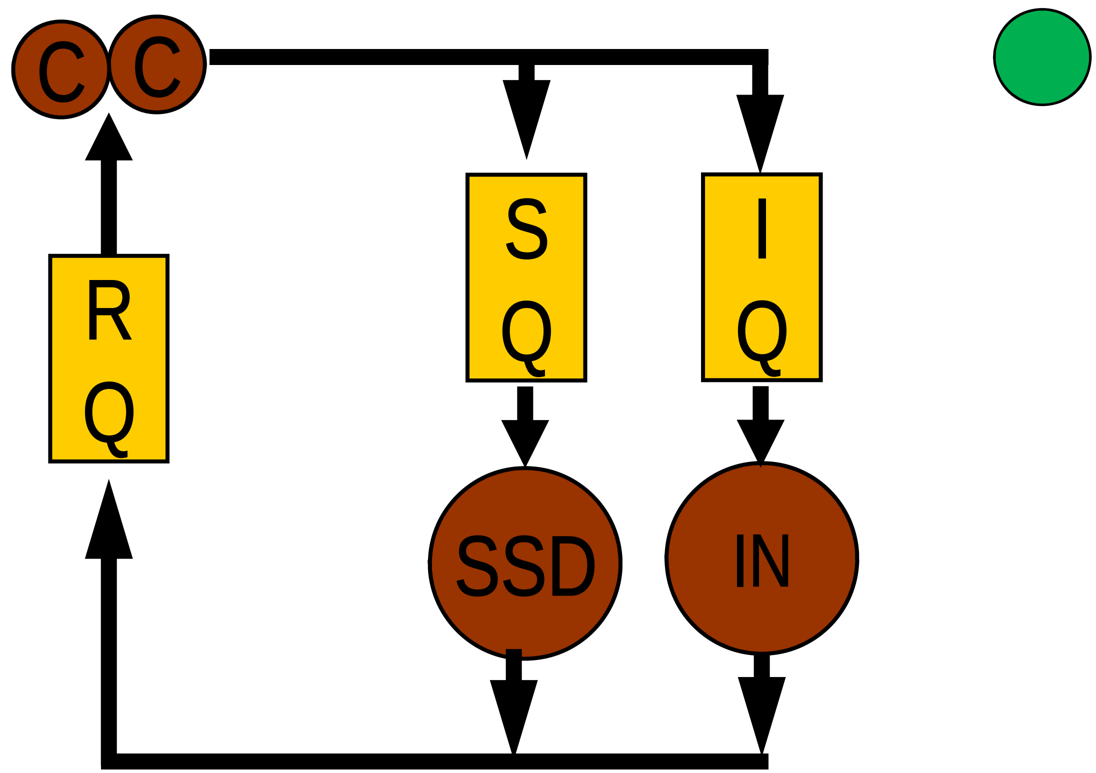
P0 holds a core until t = 155ms



P0 terminates

NCORES 2
NEW 5
CORE 100
SSD 0
CORE 30
NEW 20
CORE 50
SSD 0
CORE 50
NEW 30
CORE 20
SSD 0
CORE 20

P0 holds a core until t = 155ms



Your program will display

- Time = 155ms
  Process 0 terminates

or

- Process 0 terminates at t = 155ms

Your program will display (II)

- **SUMMARY:**
  Number of processes that completed: 3
  Total number of SSD accesses: 3
  Average SSD access time: 0 ms
  Total elapsed time: 155 ms
  Core utilization: 174.19 percent
  SSD utilization: 0.00 percent

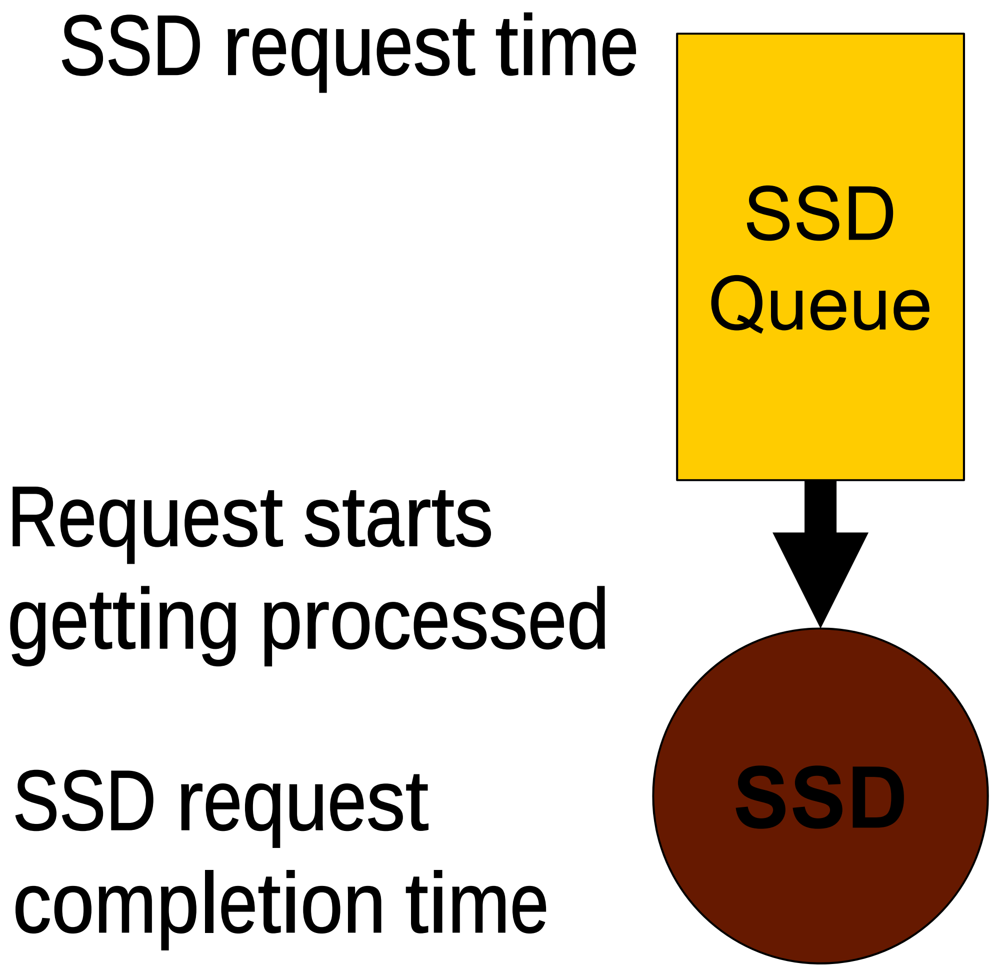
Core Utilization = (100 + 30 + 50 + 50 + 20 + 20)/155
= 270/155 =1.7419

SSD average access time (I)

- Must consider SSD contention
  - Unlikely in most cases
- Must compute difference between
  - SSD request time
  - SSD request completion time

for each SSD request

SSD average access time (II)



The best way to implement it

- Have two counters both initialized at zero
  - **SSDcount**
  - **SSDtimes**
- When a SSD request is issued by a process
  - Add 1 to **SSDcount**
  - **Subtract** current time from **SSDtimes**
- When the SSD request is completed
  - **Add** current time to **SSDtimes**

Example

- Assume a request processing time of 5 ms
- First request arrives at time t = **10 ms**
- Immediately processed
- Second request arrives at time t = **12 ms**
- First request is completed at time t = **15 ms**
- Second request can be processed
- Second request is completed at time t = **20 ms**
- Average completion time is
  (– 10 – 12 + 15 + 20)/2 =13/2 = 6.5 ms

ENGINEERING THE SIMULATION

Simulating time

- ***Absolutely nothing happens to our model between two successive "<u>events</u>"***
- ***Events are***
  - Arrival of a new process
  - Start of a computing step
  - Completion of a computing step
- We associate an ***event routine*** with each event

Arrival event routine

- Process first request of process
  - It will always be a CPU request

Core request routine

- **current time is clock**
  request time is crt
- **if a core is free :**
  mark core busy until clock + crt
  else :
  enter process in ready queue

Core request completion routine

- if ready queue is empty :
  mark core idle

  else:
  pick first process P' in ready queue
  crt' is request time for P'
  mark core busy until clock + crt‘
- proceed with next request for the process that completed the CPU request

SSD request routine

- **current time is clock**
  request time is srt
- **increment SSDcount**
- **subtract current time from SSDtimes**
- **if SSD is free :**
  mark SSD busy until clock + srt
  else :
  enter process request in SSD queue

SSD request completion routine

- **add current time to SSDtimes**
- **if SSD queue is empty :**
  mark SSD free
  else :
  pick first process request P' in SSD queue
  drt' is request time for P'
  mark SSD busy until clock + srt'
- **proceed with next request for process that completed its SSD request**

Input request routine

- **current time is clock**
  request time is irt
- **if user is free :**
  mark user busy until clock + irt
  else :
  enter process request in input queue

Input request completion routine

- if input queue is empty :
  mark user free
  else :
  pick first process request P'
  in input queue
  irt' is request time for P'
  mark user busy until clock + irt'
- proceed with next request for process that completed its input request

The simulation scheduler

1. Find next event by looking at
   a) Core completion times
   b) SSD termination times
   c) Completion times of input requests
   d) Arrival time of next process
2. Set current time to ***<u>time of next event</u>***
3. Process event routine
4. Repeat until all processes are done

Organizing our program (I)

- Most steps of simulation involve scheduling future completion events
- Associate with each completion event an event notice
  - **Time of event**
  - **Device**
  - **Process sequence number**

177,"Core", 12

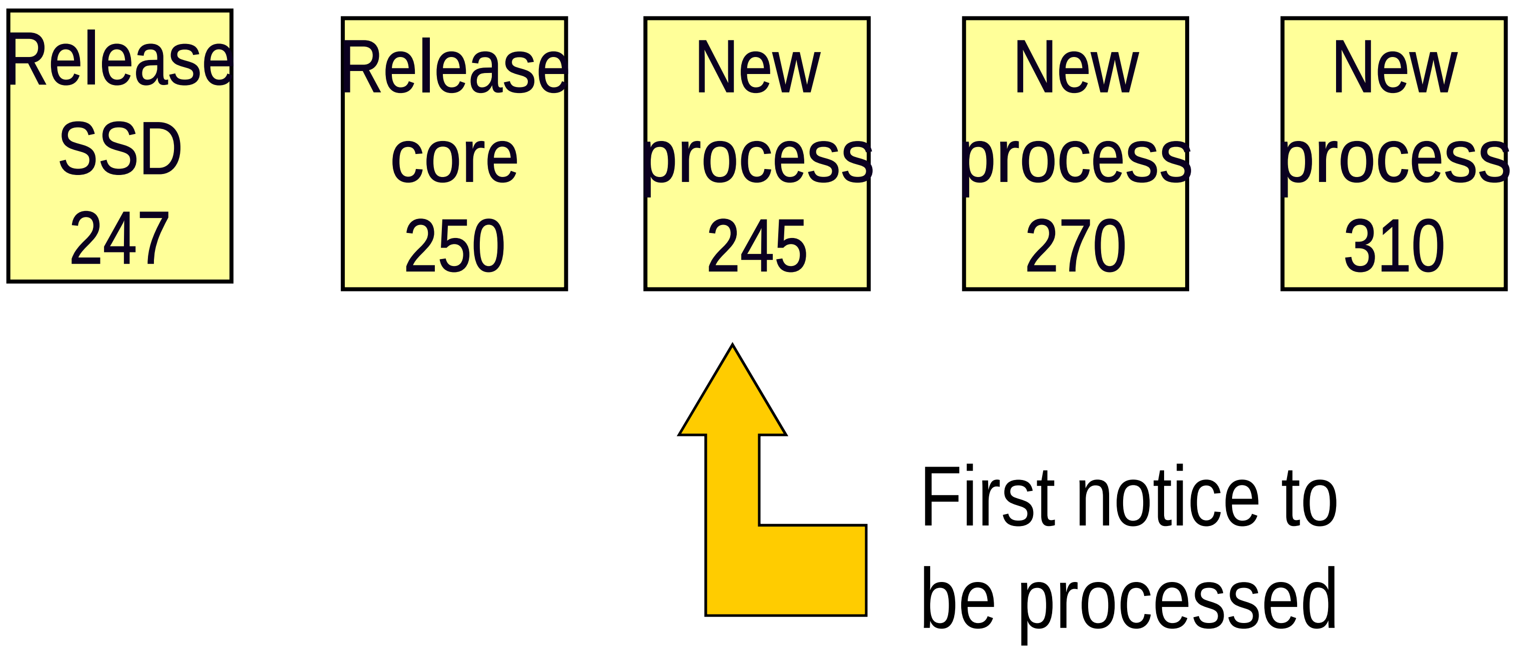
Organizing our program (II)

- Do the same with process starts
  - **Time of event**
  - **Process start**
  - **Process sequence number**

120,"Arrival", 19

Organizing our program (III)

- Process all event notices in chronological order



Organizing our program (IV)

- Keep the event list sorted (***priority queue***)

New process 245	Release SSD 247	Release core 250	New process 270	New process 310

First notice to be processed is head of the list

Organizing our program (V)

- Overall organization of main program

```
schedule first event    # will be a process
start
while event list is not empty :
        process next event  in list
print simulation results
```

Organizing our event list

- ***Priority queue***
- Two kinds of entries
  - ***Computational steps completion times:***
    - Created and inserted "on the fly"
  - ***Process arrivals:***
    - Created during input phase
    - Already sorted

AN IMPLEMENTATION

- My main data structures would include:
  - Data table
  - Process table

The data table

- Stores the input data
- Line indices are used in process table

Operation	Parameter
NEW	5
CPU	10
INPUT	0
CPU	20
NEW	20
CPU	50
…	…

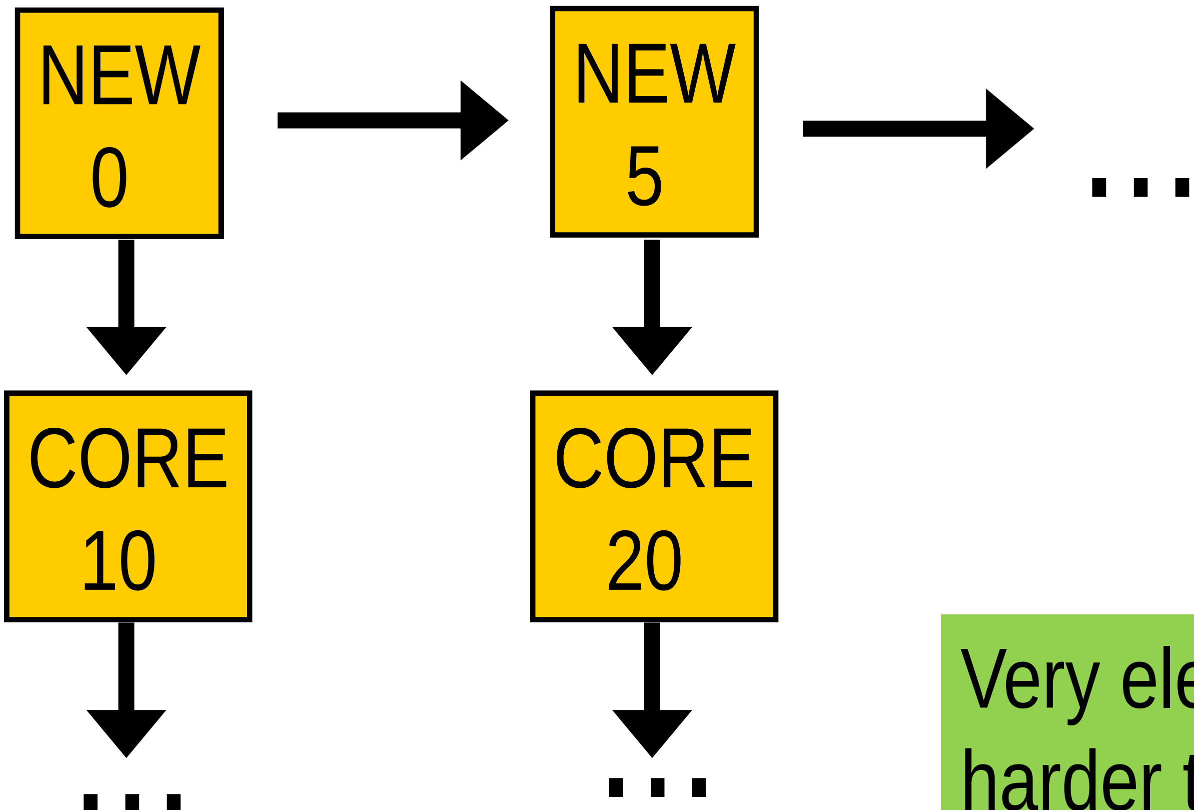
The process table (I)

Start Time	First Line	Last Line	Current Line	State
5	0	3	varies	varies
20	4	…	…	
…	…	…	…	

The process table (II)

- One line per process
  - ***Line index is process sequence number*!**
- First column has start time of process
- First line, last line and current line respectively identify first line, last line and current line of the process in the input table
- Last column is for the current state of the process

A full list implementation of the data table



Very elegant but harder to debug

Reading your input

- You *must* use I/O redirection
  - **`assign1 < input_file`**
- Advantages
  - Very flexible
  - Programmers write their code as if it was read from standard input
    - No need to mess with **`fopen()`**, **`argc`** and **`argcv`**