1. Good Evening
2. We will begin at 9:10 pm.
3. Introduction to Threads.

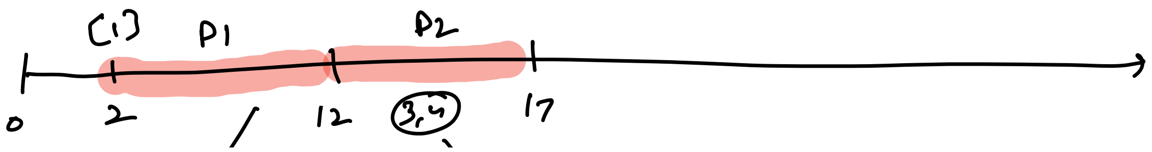
Agenda

1. Starvation in FCFS & SRTF
2. Round Robin
3. Threads
   - Intrtotudction
   - Single core vs Multi core
   - Concurrency vs Parallelism
   - Code [using Java]
     - Introduce Thread
     - Hello ward via threads Ex1
     - Print 1 to 100 via threads Ex2

Starvation (FCFS)

SRTF

	Id	AT	BT
→	1	2	10
✓	2	4	5
✓	3	6	4
✓	4	8	6



[2,3,4]

2,3,4 was starving

are starving [i.e. not getting CPU's attention & not making progress]

Round Robin

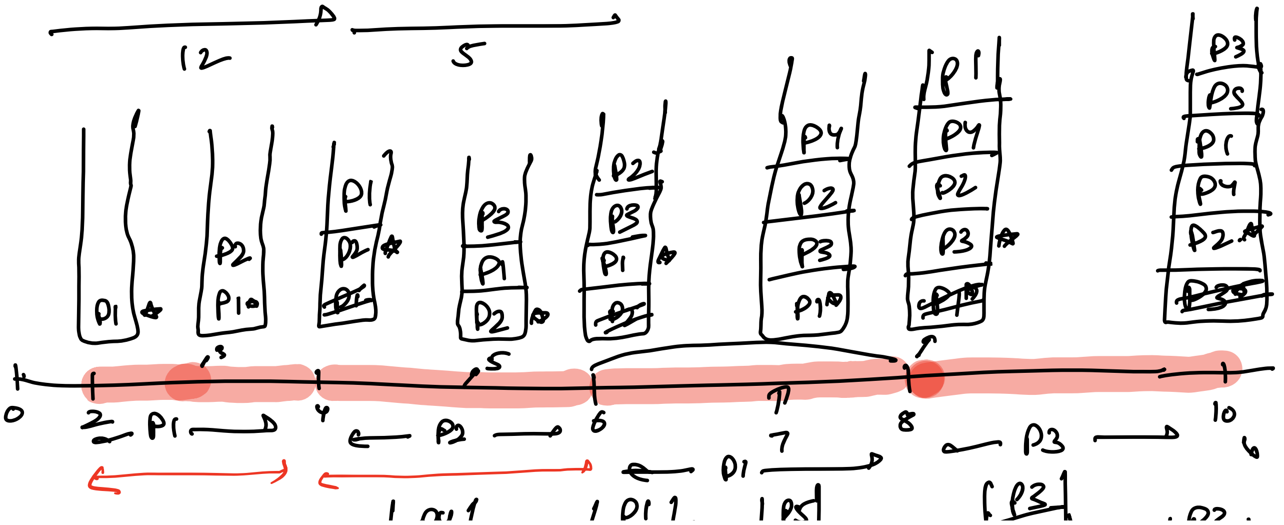
└ Quantum → q sec

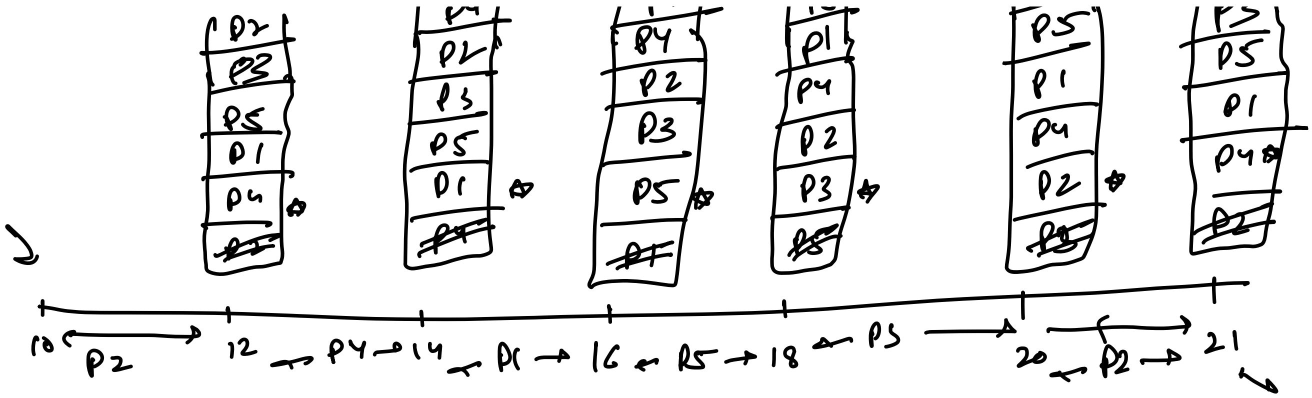
Run

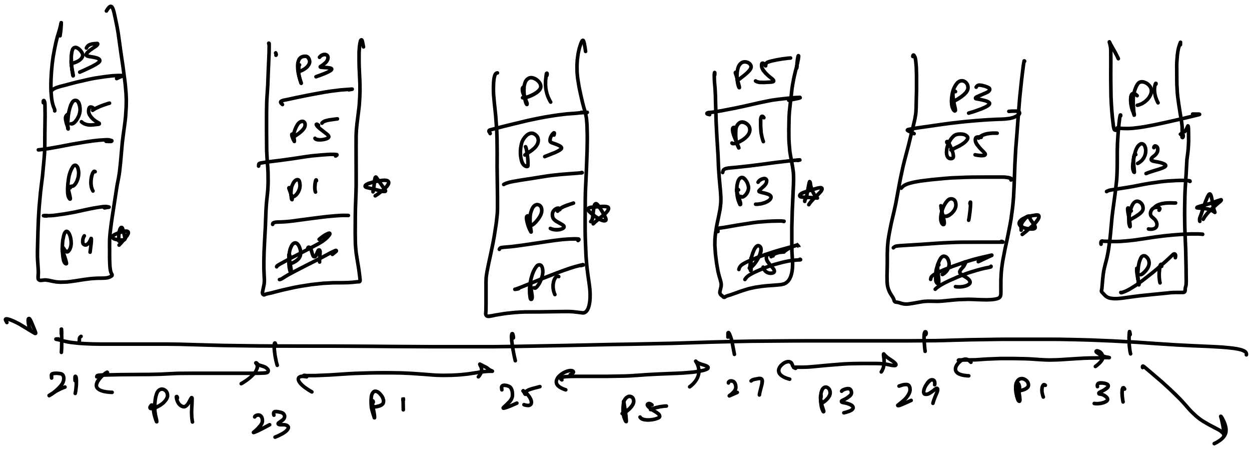
* Of a process finishes or quantum of time with the current process elapses, CPU will pick a new process.

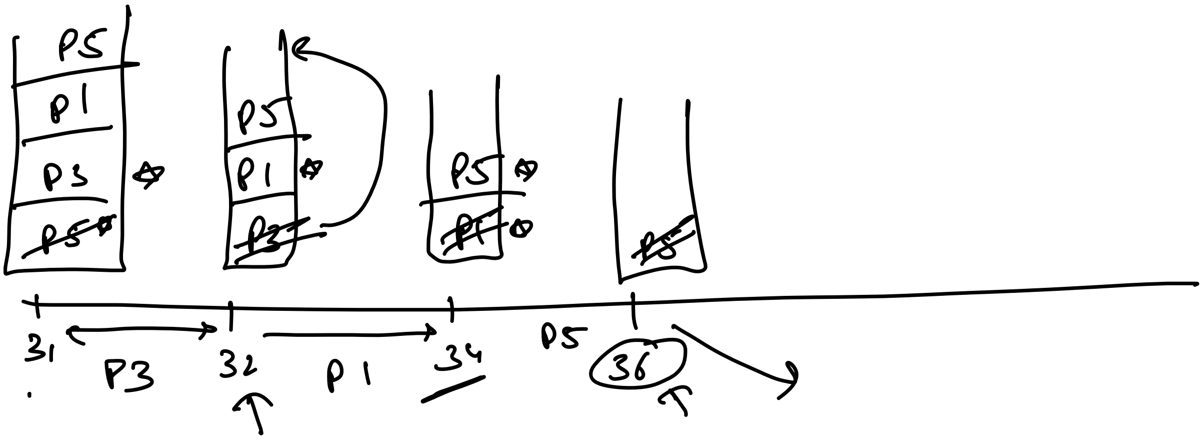
q = 2 sec

id	AT	BT	RT
1	2	12	~~10~~ ~~8~~ ~~6~~ ~~4~~ ~~2~~
2	3	5	~~3~~ ~~1~~
3	5	7	~~5~~ ~~3~~ ~~1~~
4	7	4	~~4~~ ~~2~~
5	10	8	~~8~~ ~~6~~ ~~4~~ 2









q = 2 sec

3 × 10^9 ~~GHz~~ cycles per second

Round Robin

- Partial ?
- Starving ?
- Simple

q = 30 seconds → FCFS [Starvation]

q = 0.0002 seconds → More Context Switching

PCB

Program Counter =

$q = 2 \times 10^{-9}$ sec

Appropriate quantum is required which can be found by experimentation

[illegible]

~ 10^{-9} seconds

Throughput → RR algorithm

→ # of processes completed per unit of time

$5/36$ = 0.14 process/second

Latency → Time taken by a process from arrival till finish

SRTF & RR

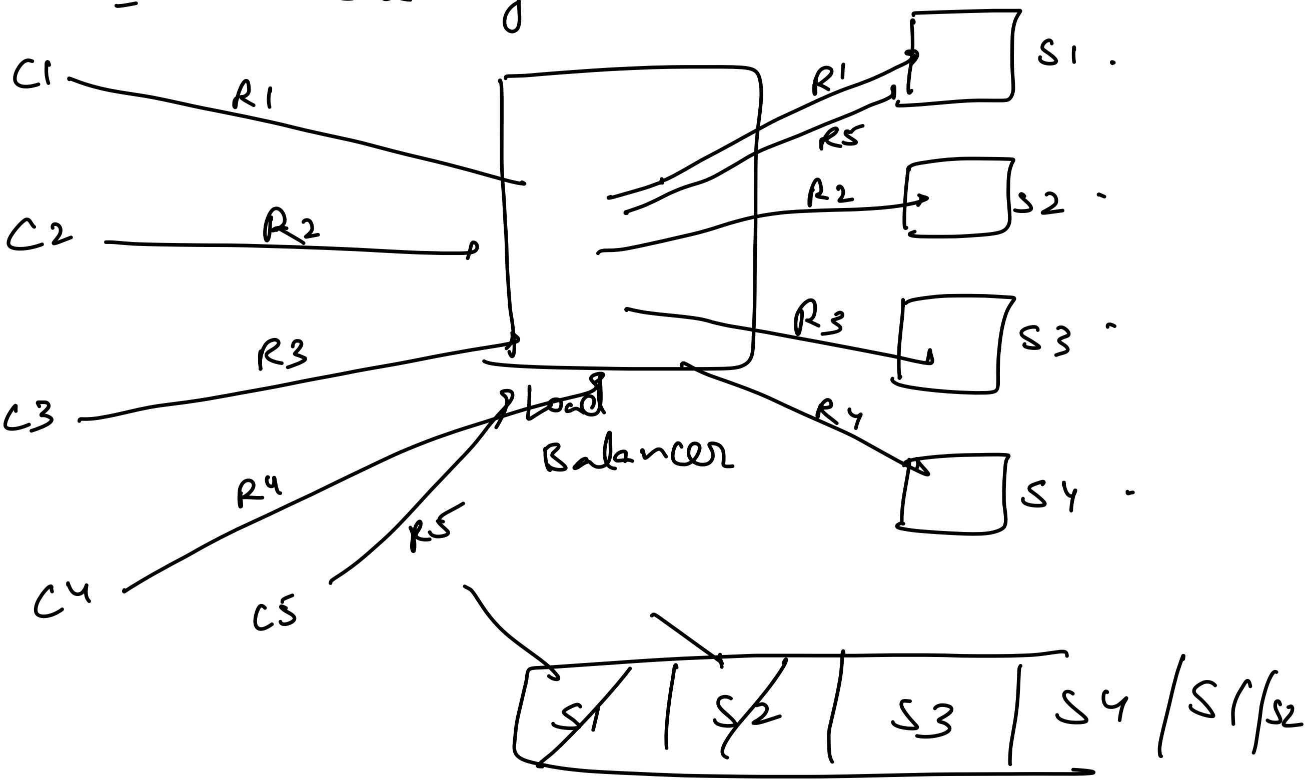
which one has higher latency?

Latency vs Starvation

[illegible]

Another Scenario where Round Robin is used

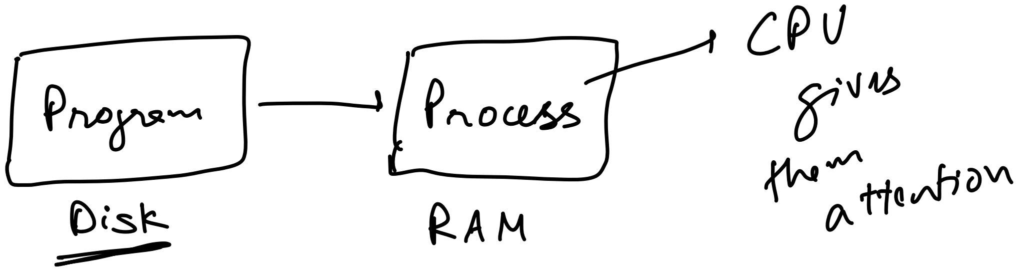
= Load Balancing



Introduction to Threads

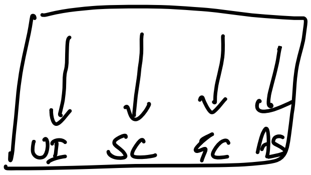
Break

10:19 — 10:29



Word Processor

- → Shows you your input → UI Display
- → Spell Check
- → Grammar check
- → Auto Suggest



```
main() {
    L1 ✓
    L2 —
    fun() ✓
    L3 ✓
    L4 ✓
}
```

```
fun() {
    L11 ✓
    L21 ✓
    L31 ✓
    L41 ✓
}
```

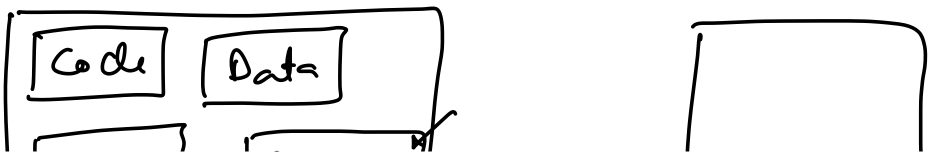
Serial or Sequential → Concurrent

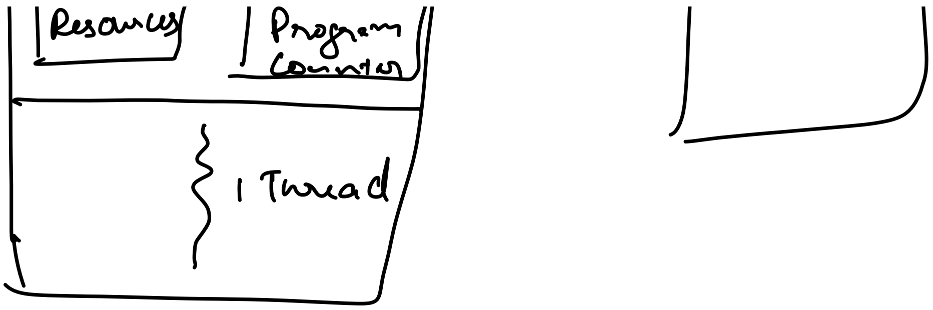
Parallel

To explain the difference.

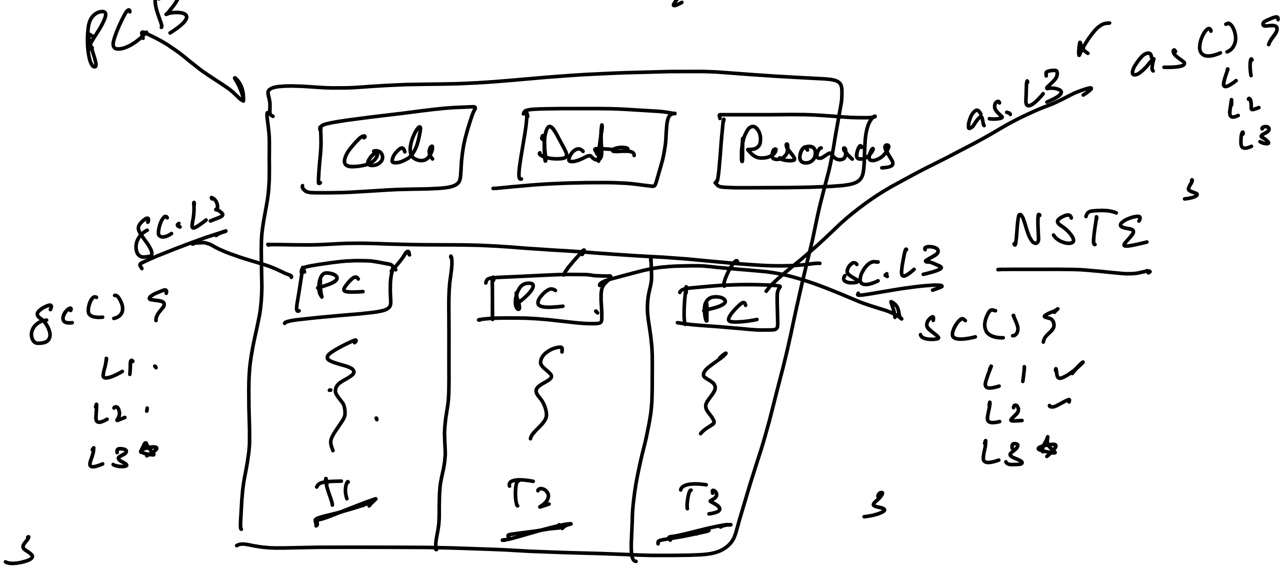
Thread

→ Unit of CPU execution

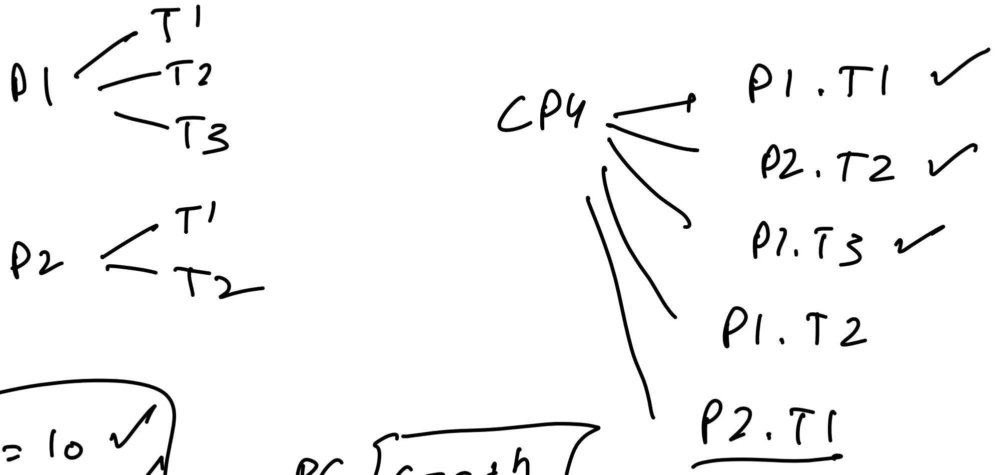




Process → Single Threaded Process
or
Sequential Process

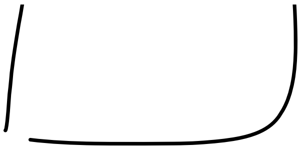


Process { Multi-threaded process
or
Concurrent Process }

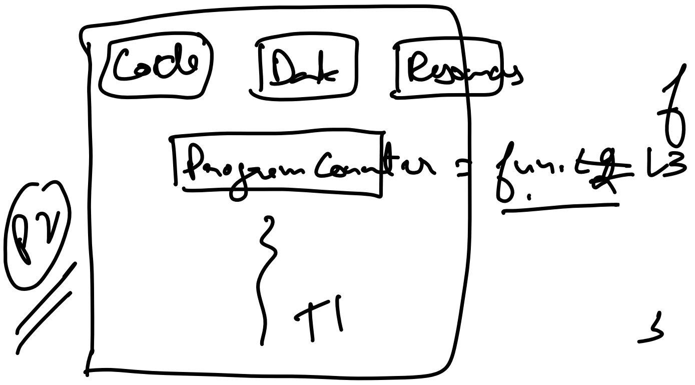


a = 10 ✓
b = 20 ✓
c = a+b *

PC [c=a+b]

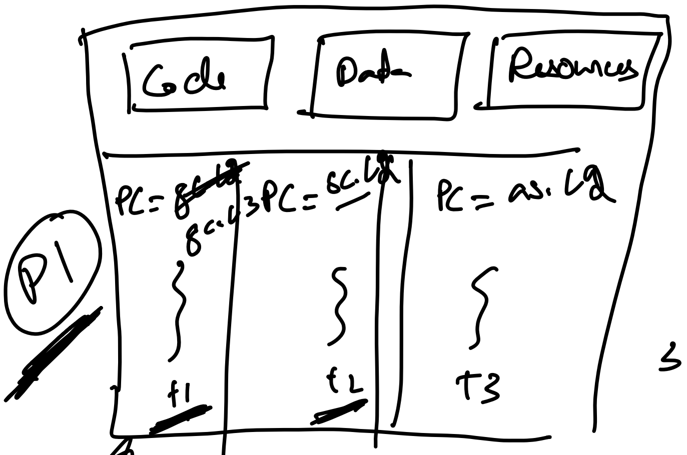


5 minutes



```
fun() {
    L1 ✓
    L2 ✓
    L3.
    L4
}
```

Process, Single Threaded Process, Sequential Programs



t1 = gc

```
gc() {
    L1 ✓
    L2 ✓
    L3
    L4.
}
```

t2 = sc

```
sc() {
    L1 ✓
    L2
    L3
    L4
}
```

t3 = as

```
as() {
    L1 ✓
    L2
    L3
    L4
}
```

Process, Multi-threaded Process, Sequential as well as concurrent

RR

CPU	P1.T1	P2.T1	P1.T2	P2.T1	P1.T3	P1.T1
	gc.L1	fun.L1	sc.L1	fun.L2	as.L1	gc.L2

P1 = gc.L1 ✓

t1 t2 t3

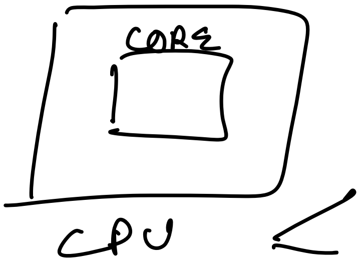
t1 SC. L1 t2 ✓
SS. L1 t3 ✓
SC. L2 ✓ ✓

Concurrent vs Parallel

↓

Single Core vs Multi Core

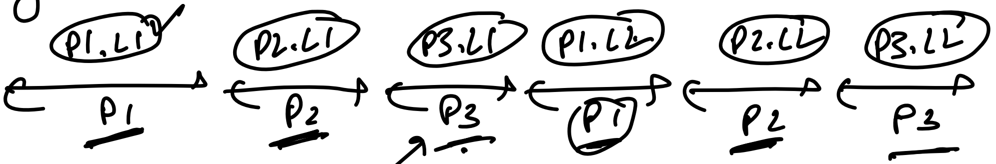
CORE → Executing unit within CPU



Single core CPUs can give concurrency i.e. multiple processes are at different stages of execution but none of them is running at same time

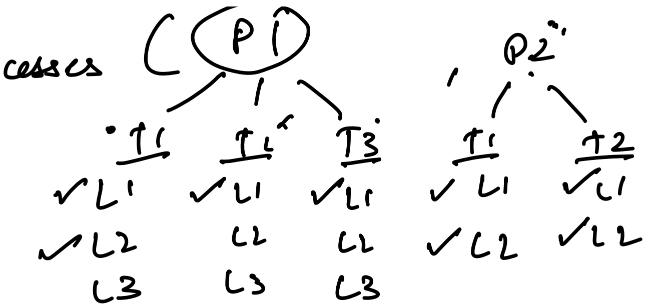
Single Core ↓

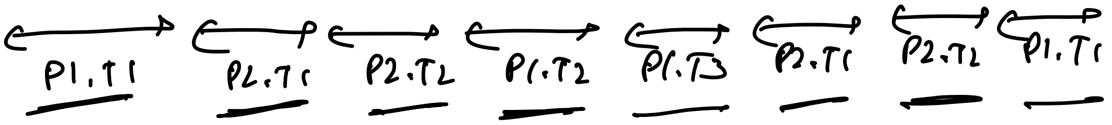
Single Threaded Processes (P1, P2, P3)



P1	P2	P3
✓L1✓.	✓L1✓.	✓L1✓.
✓L2✓.	✓ L2✓.	✓L2✓.
L3 .	L3 .	L3 .

Multi-threaded Processes



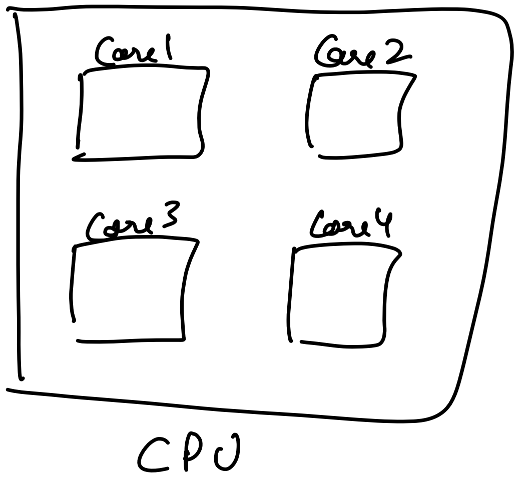


Q1. [Single Core CPU can run multiple single threaded processes ?]

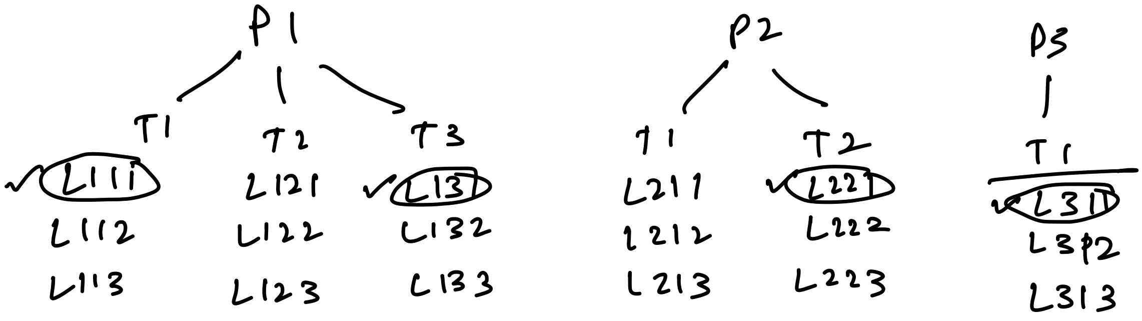
Q2. [Single Core CPU can run multiple multi-threaded processes]

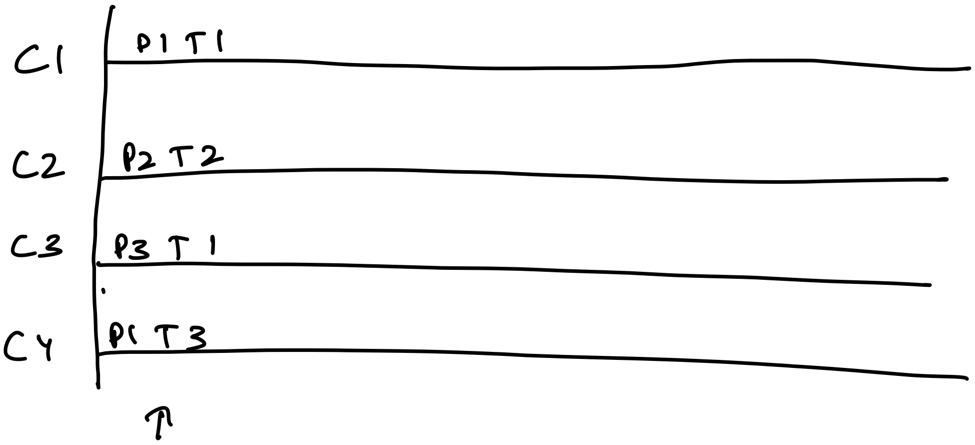
→ But concurrently not parallely ?

Multi-core systems



Multi-thread multiple processes





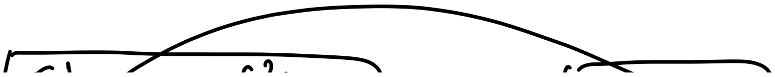
Concurrent Parallel

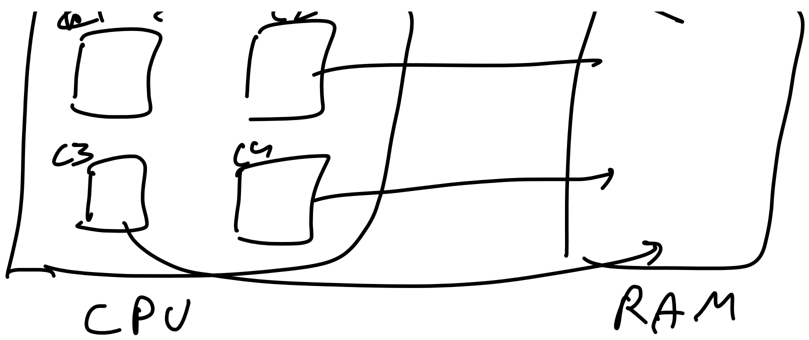
1. Are multiple processes in different states of execution in both of them? ✓

2. Can multiple processes make progress at same time in both of them? No only in parallel

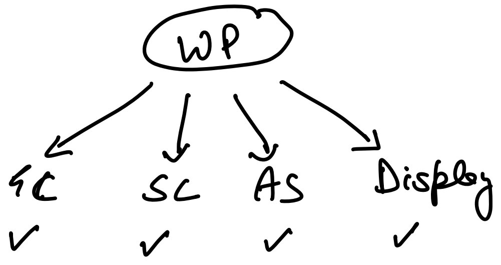
→ CPU runs processes

→ Core runs threads





Thread vs Process

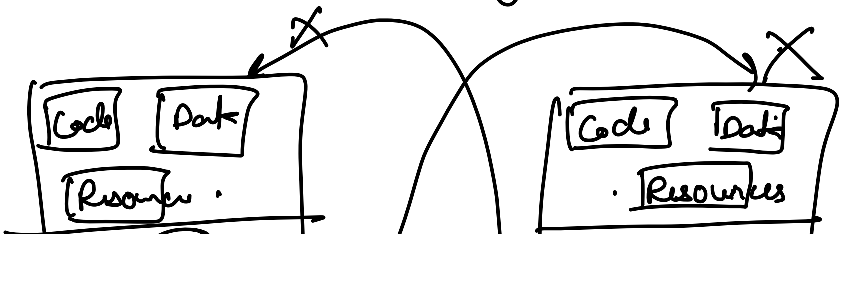


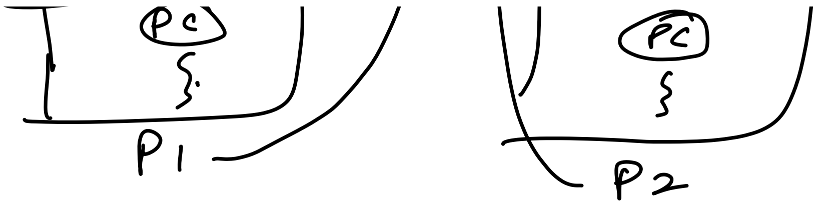
Work → Divide it for concurrent/parallel exeution

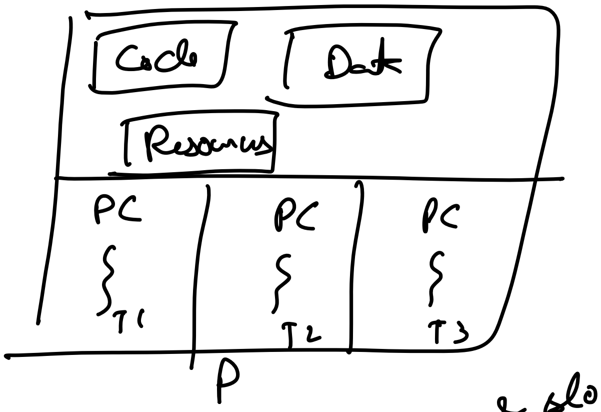
Should we create separate processes or separate threads?

Related tasks → likely to share data → Threads

Unrelated " → Not likely to share data → Processes







Data sharing is difficult & slow with processes (IPC)

" " is fun with threads.

Backlog → * Intro to thread via Code
* Hello World via threads
* Print 1 to 100 via "