Agenda

(4) → Threads

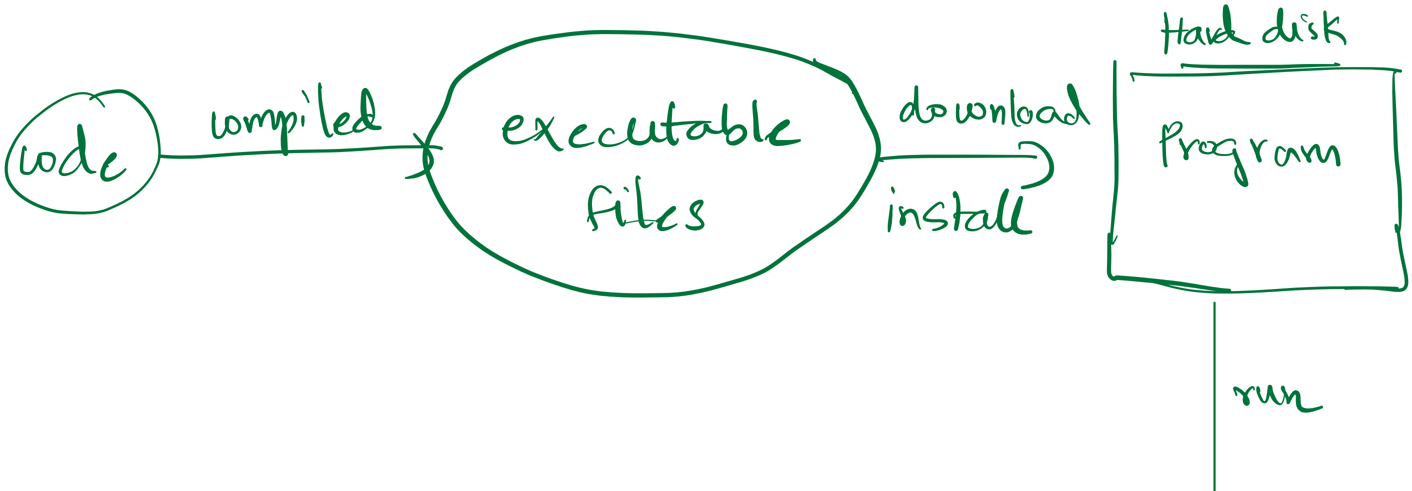
1. Processes & threads
2. Single core vs multicore systems.
3. Context Switching
4. concurrency vs parallelism
5. Execution of thread. — Hello World.

- OS
- COA

Computer architecture

.dmg .exe .apk

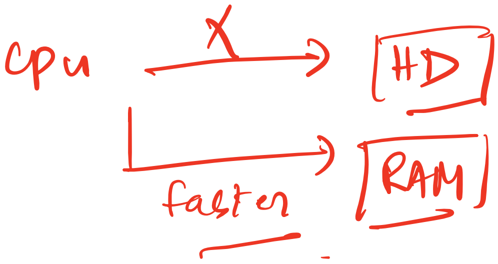
executable files.



↓ RAM

* Process is a ← Process.

program in Execution



CPU:

2.2 GHz

↓ 2.2×10^9

No. of instructions per second.

RAM	SSD
↓	↓
8GB	1TB
5K	5K

Word Processor :

1. Spell check. → P1
2. Grammer check → P2
3. style change → P3
4. suggestions → P4
5. Auto complete. → P5

Process :

Process Control Block (PCB) .

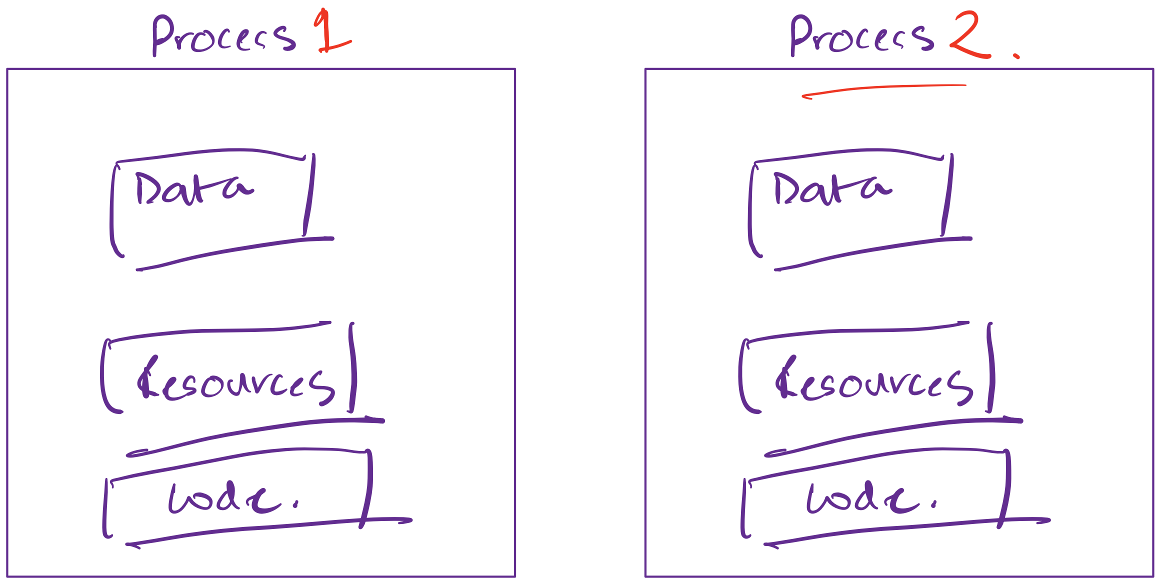
↳ A data structure which stores information about a process.

```
class PCB {
    string code;
    int pid; → process id.
    priority;
    Location
    Memory details // Data.
    register
    List <variables>
```

address of the

program counter → next line of code

}



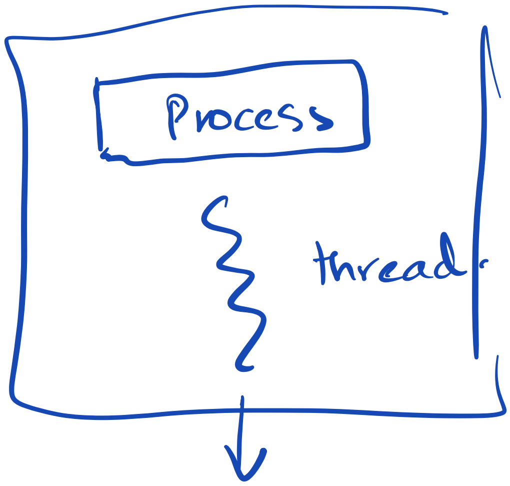
Word Processor :

1. Spell check. → P1
2. Grammer check → P2
3. style change → P3
4. suggestions → P4
5. Auto complete. → P5

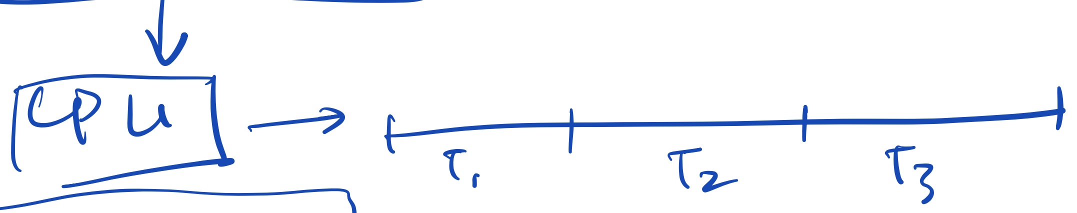
1 Process
↓
5 threads

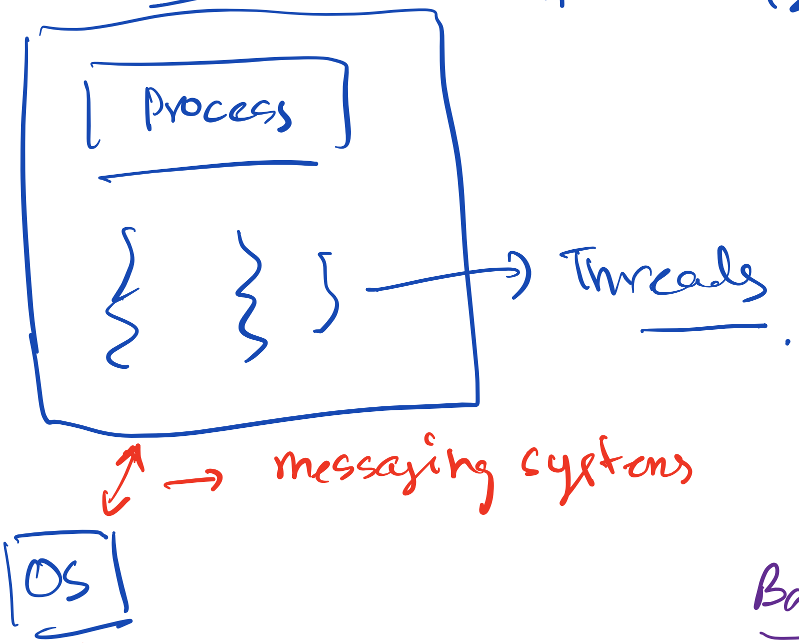
Thread : basic unit of process
light weight process
part of process ✗

→ Single unit of Execution ✓



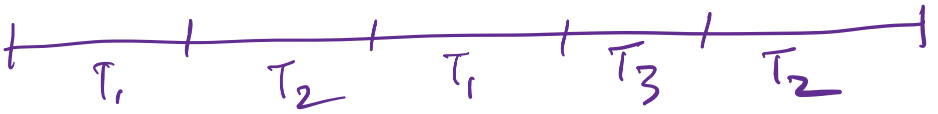
Every process must have atleast one thread.

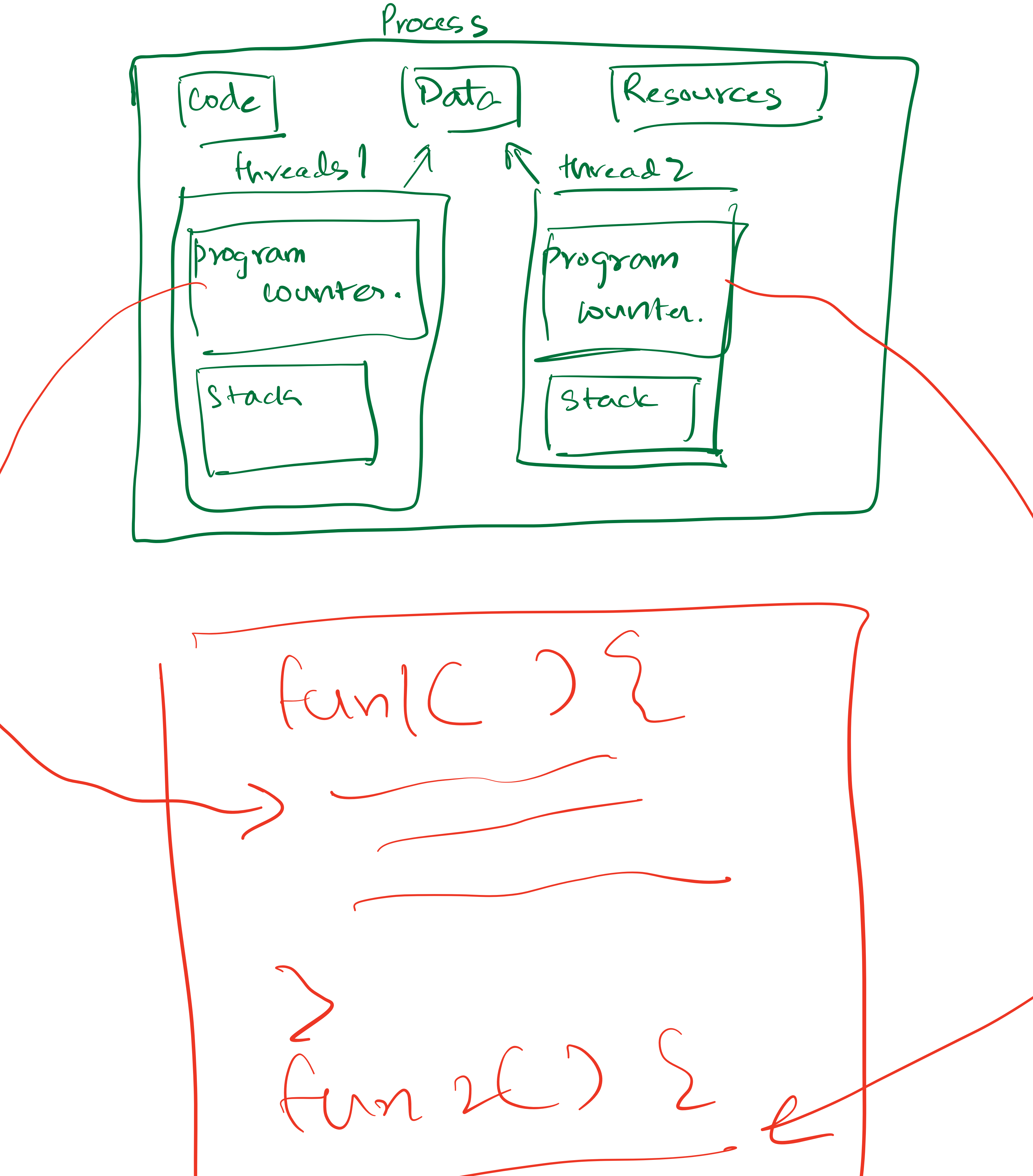


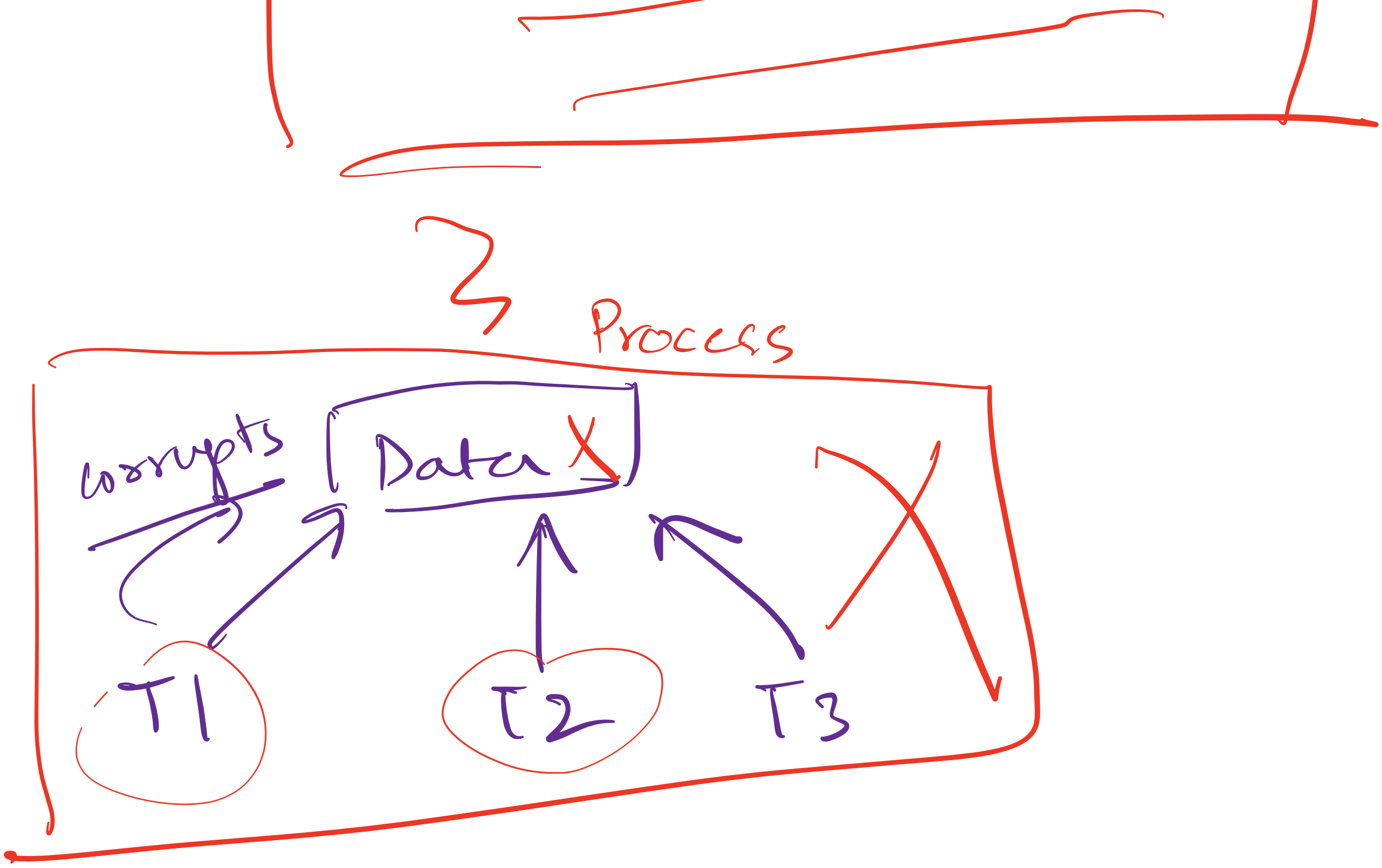


→ CPU scheduler

Based on following :
Resources
time
Priority.


T1
T2
T1
T3
T2


Process
Code
Data
Resources
threads 1
thread 2
program counter.
Stack
program counter.
stack
fun1( ) {
}
fun2( ) {



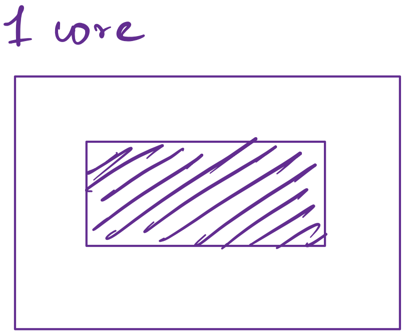
The difference Between threads & Process:

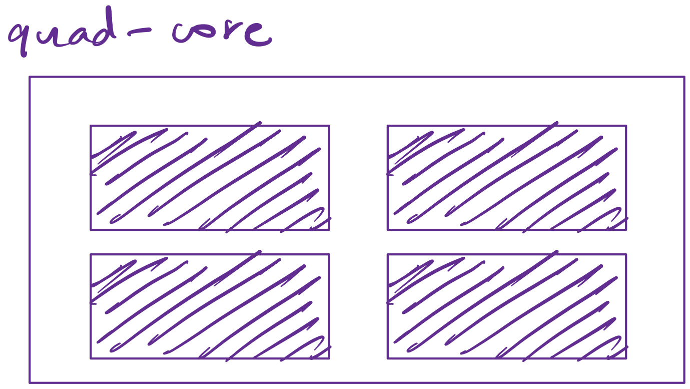
1. Data Sharing : All threads share common data but process doesn't share data.

2. Memory : A process takes more memory than thread, so creating new process is memory consumig

Break Till 8:19.

Single core Vs Multi core





Each core means it can run 1 thread at a time.

Each core = Each Thread.

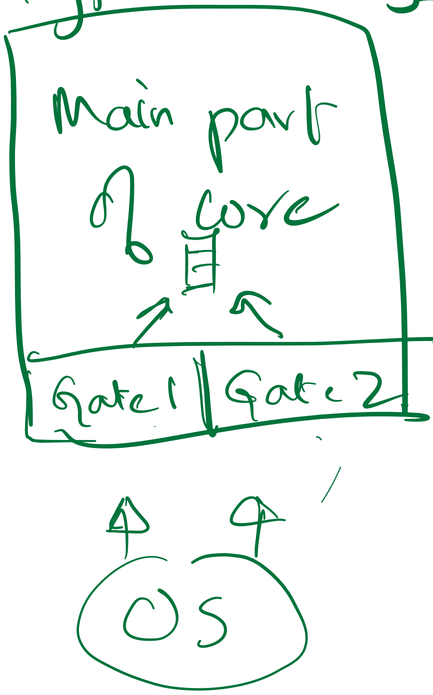
4 core cpu → It can run 4 threads at most at a moment.

i3 → dual core

i5 → quad core

i7 → quad + hyperthreading -

Hyperthreading



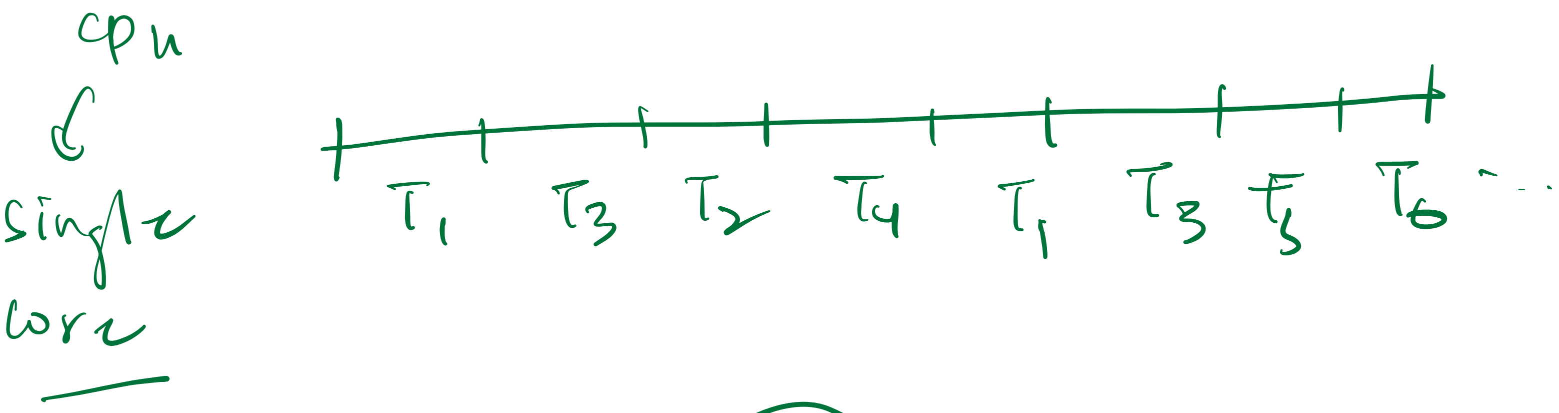
Threads - 2047 ← 2000 cores)

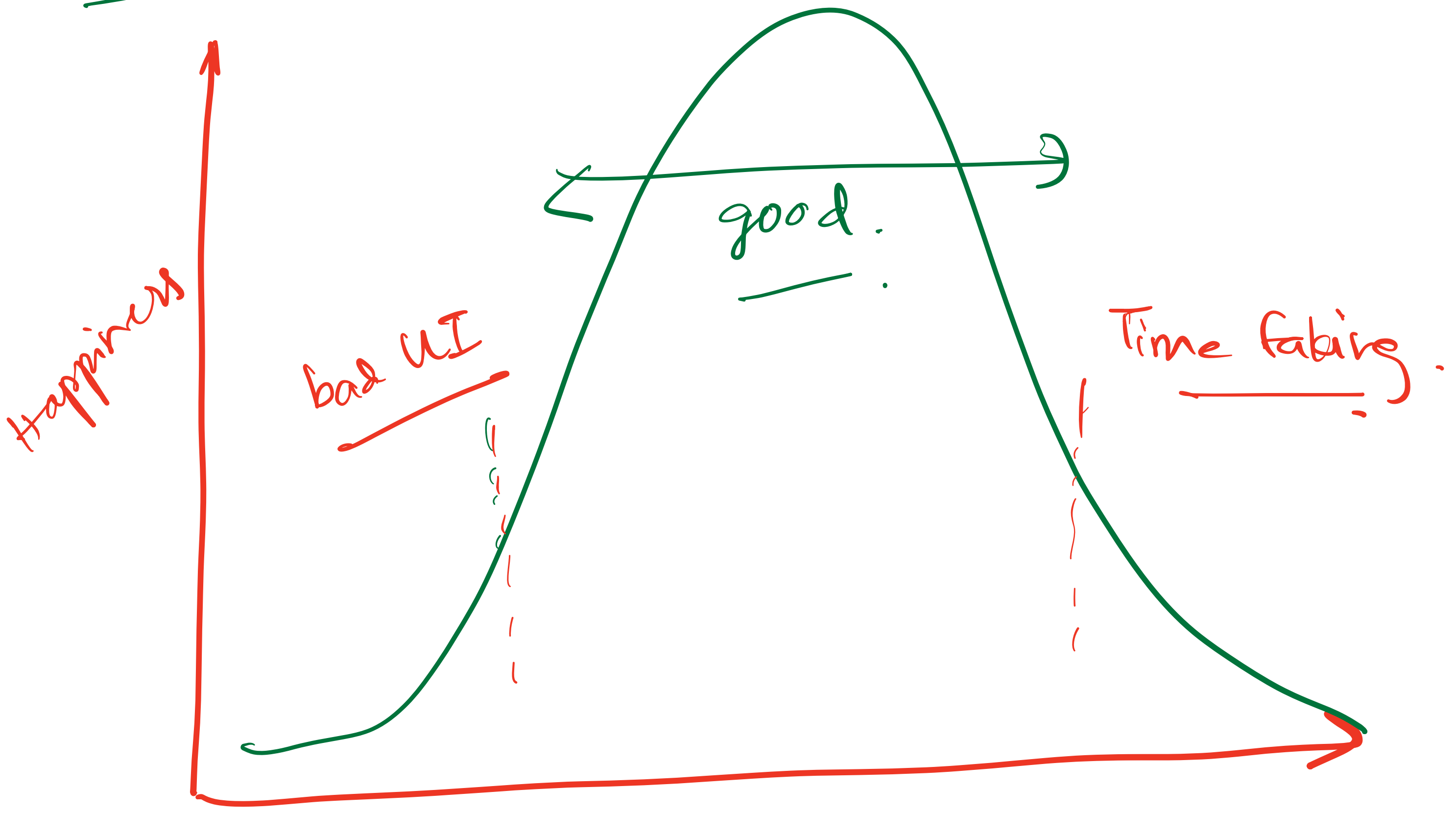
Processes - 399.

Are in action.

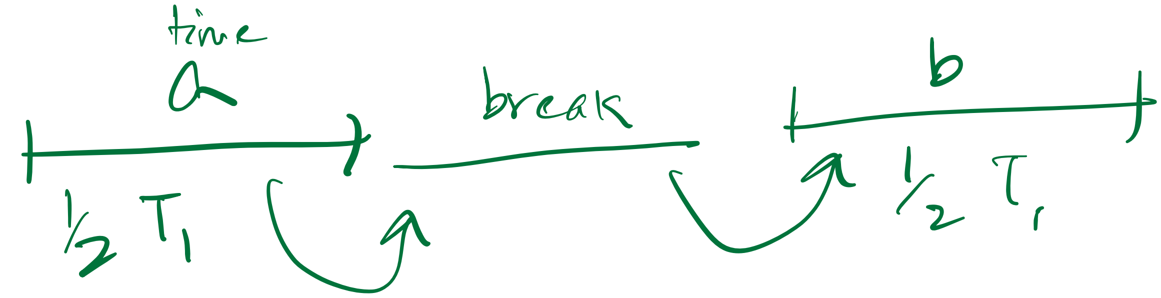
CONTEXT SWITCHING

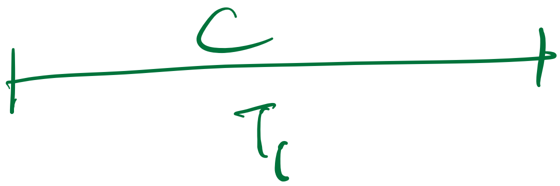
10 Threads.





no. of context switches.





① $a + b = c$

② $a + b < c$

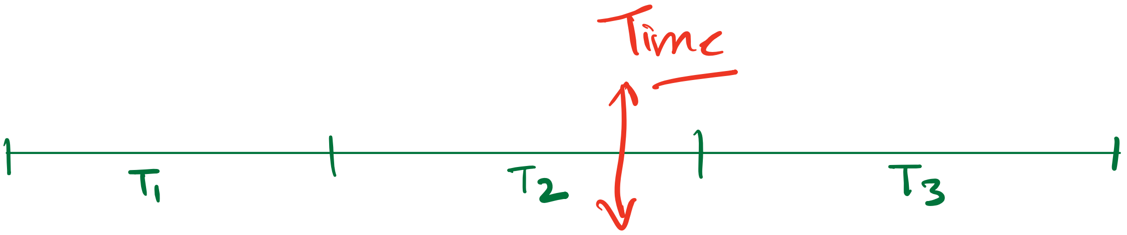
③ $a + b > c$.

cpu schedular takes care of context switching

Concurrency & Parallelism

Case - I

Single core system with no context switching. → 1 thread is completed before moving on to the next thread.



① How many threads are partially completed?

= 1 → concurrency

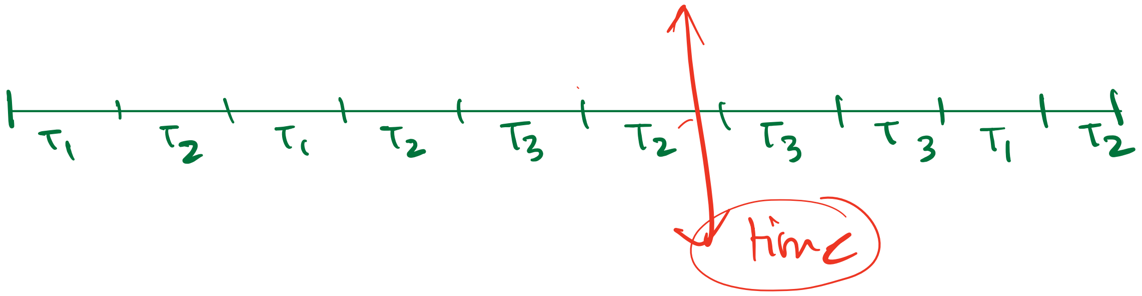
② How many threads are making progress?

⇒ 1 → parallelism

Neither Concurrent nor parallel.

Case - II

Single core system, but context switching is allowed.



① How many threads are partially completed?

$\boxed{>1}$

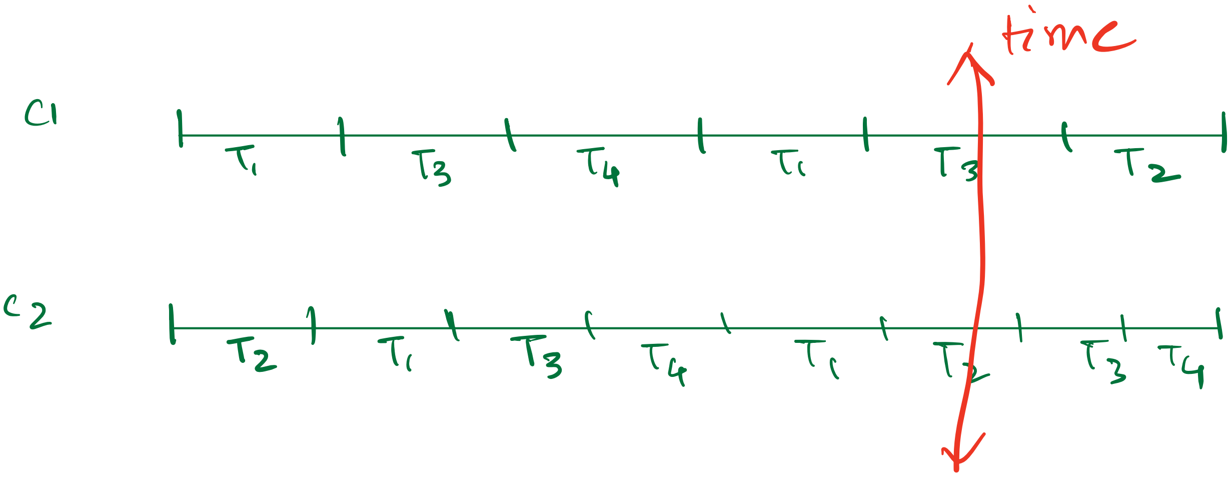
② How many threads are making progress?

$= 1$

Concurrent but not parallel.

Case - III

Dual Core + Context Switching.



① How many threads are partially completed?

71

② How many threads are making progress!

21

Both concurrent and parallel

Execution of thread :

Don't think about threads,
think about the task..

① **Define the task.:**

- Create class of the task

```
class HelloWorldPrinter{


 }
```

② Implement - Runnable Interface.

```
class HelloWorldPrinter implements Runnable
{
      void run () {



      }
}
```

} code you want to run on the thread.

③ Create & Start a thread.

Home Work

Print 1-100 in any order, but all of them using differents threads.

Printing 1 using <thread-Name>
" || "   "
"   2   using   "