Agenda :

1. Process vs Threads
2. Concurrency vs Parallelism
3. Single core vs multicore
4. Multi thread program - simple printer.

CWA

Process : A program in execution

code → (compiled) → executable file (.exe, .dmg, .apk) → (install) → HD program

Run this program → RAM process

CPU → X → HD (slow)

↳ RAM (fast)

cost

1TB HD	1TB RAM
5-6K	2 lakhs

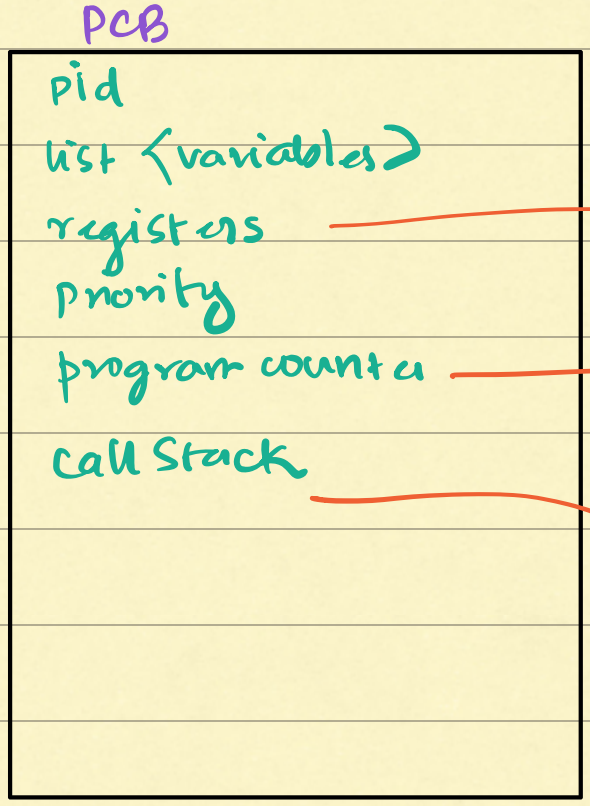
cpu

↳ 2.2 GHz

2.2×10^9 instruction/second

```
for(i=0; i<10; i++){
    cout<<"Hello";
}
```

Process : PCB → process control Block

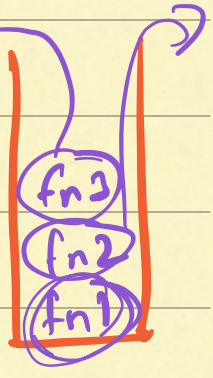


registers → smallest unit of memory

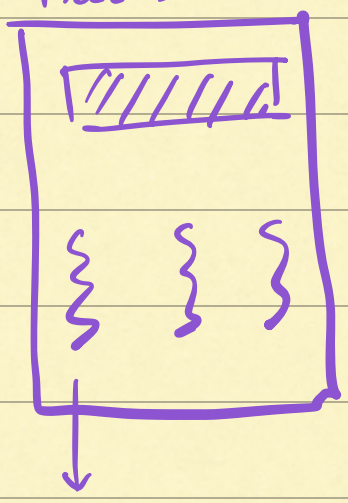
program counter → A pointer to the current line of execution.

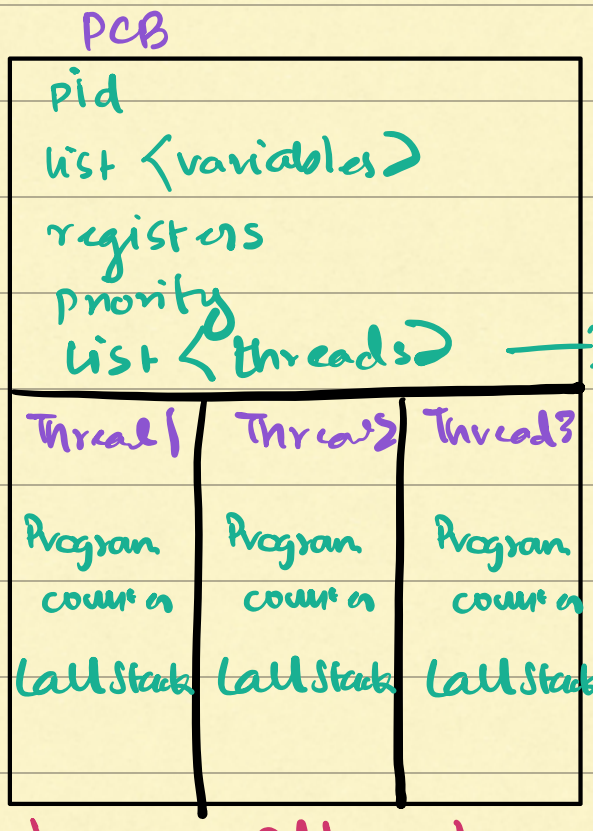
call stack → Stack to track the functions being called.

```
fn2(){
}

fn3(){
}

fn1(){
    @1000 → fn2()
}

fn1(){
    → fn(2){
    fn(3)
}
```



Threads : light weigh process X
sub process x
smallest unit of CPU execution ✓
thread is what a CPU executes





IPC

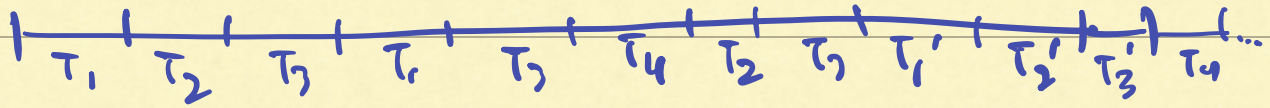
- A process will have atleast one thread.
- Data Sharing is easier in between threads
- Creation of process is expensive compared to threads

MS-Word :

- Auto Save → P1
- Auto spell check / correct → P2
- takes input → P3
- visually renders it. → P4
- search. → P5

Context Switching :

5 Threads

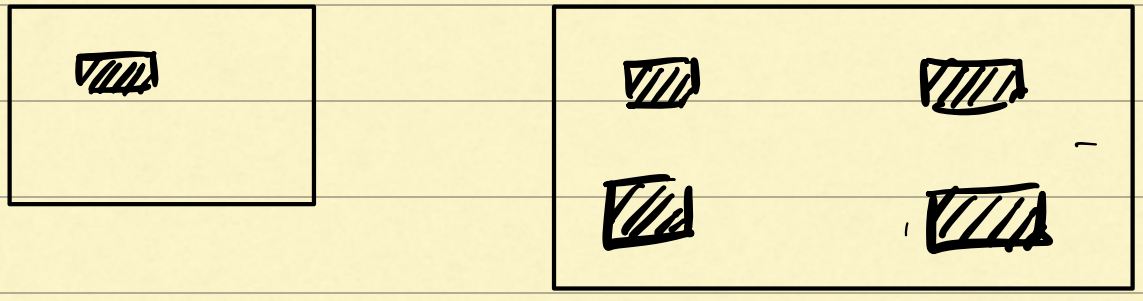


CPU Scheduler → schedules which thread to run

↓

Round Robin.

Single Core vs Multi Core



↳ 4 : quad core.

1 core = 1 thread at a moment.

4 cores ≃ 4 threads at any moment.

i3 : dual core

i5 : quad core

i7 : quad core + Hyper threading.

Break till : 8:15

Concurrency vs Parallelism :

Case I : • When there is a single core
• There is no context switching, ie each thread completes then moves on to another thread.

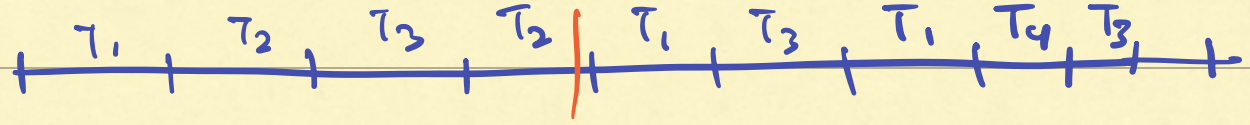
How many threads are partially complete at this moment ?

(1) → no concurrency

How many threads are currently making progress ?

(1) → no parallelism

Case II : Single Core
Context switching allowed.



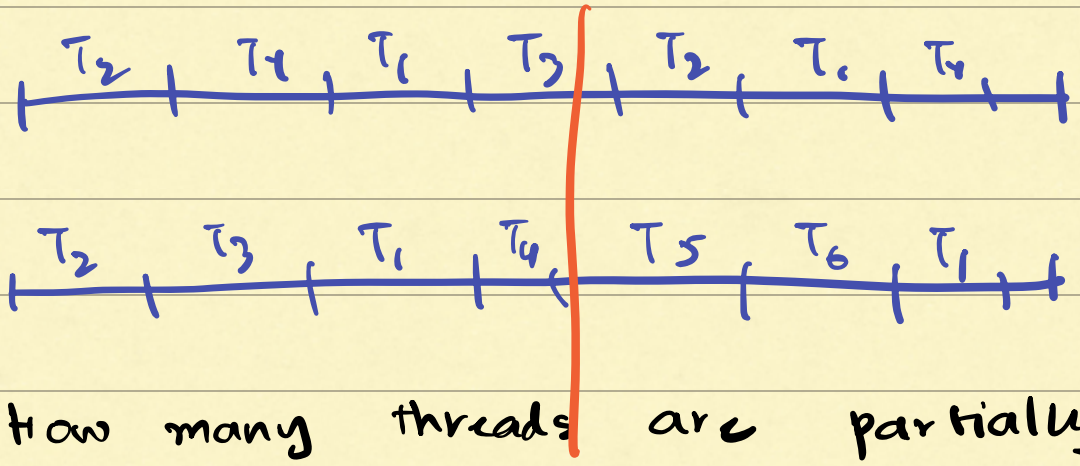
How many threads are partially complete at this moment ?

(N) → concurrency

How many threads are currently making progress ?

(1) → no parallelism

Case III : Dual Core + Context Switching.



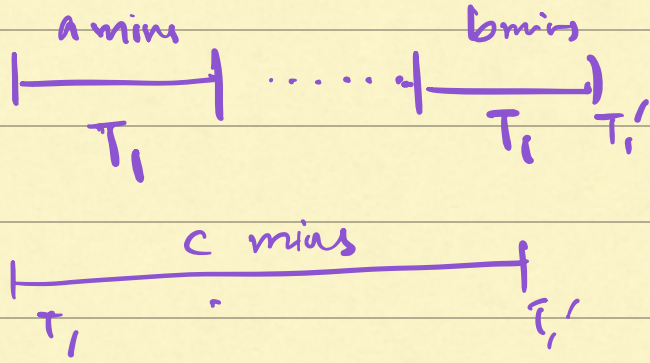
How many threads are partially complete at this moment ?

(N) — concurrency

How many threads are currently making progress ?

(N) - Parallelism

with context Switching.

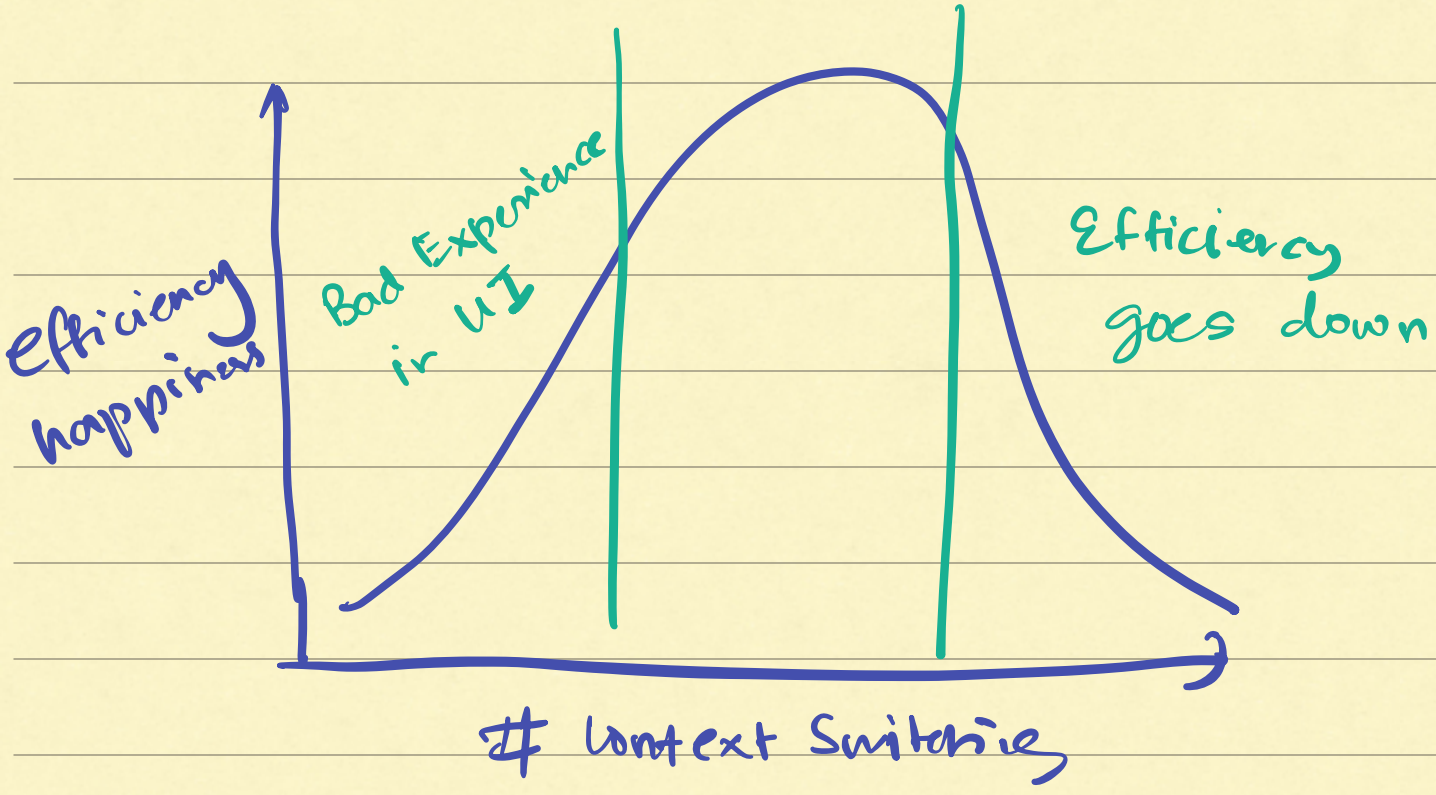


$a + b = c$

$\boxed{a + b > c}$ ✓

$a + b < c$

- Saving the context before moving from T_1
- Fetching the context when picking up T_1 again



How to write a multithread program

1. Define your task:

   Create a class : HelloWorld

2. Implement Runnable

   Class HelloWorld implements Runnable{

   void run() {

   }

   }

3. Create a Thread and give this class object (task object) to the thread.

4. Start running the Thread.