⇒ Recap

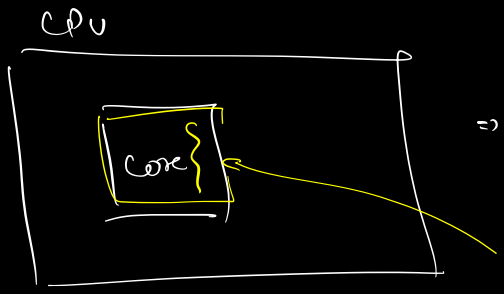
i) Scheduling Algo
- FCFS
- SRTF
- Round Robin

ii) Latency and Throughput
↓
SRTF is better than RR

iii) Threads and Processes

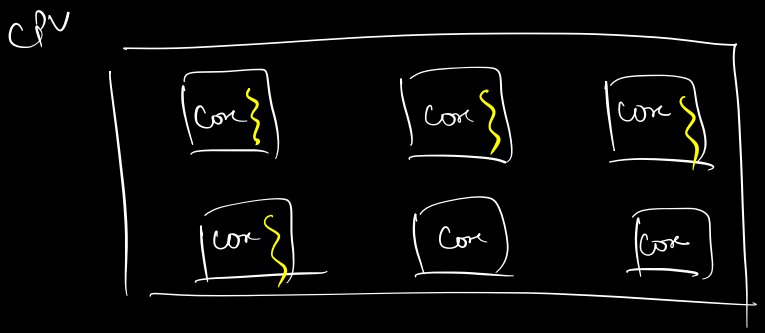
⇒ Multicore vs Single Core CPUs:

Core: execution or processing unit of a CPU.



⇒ Single core CPU.

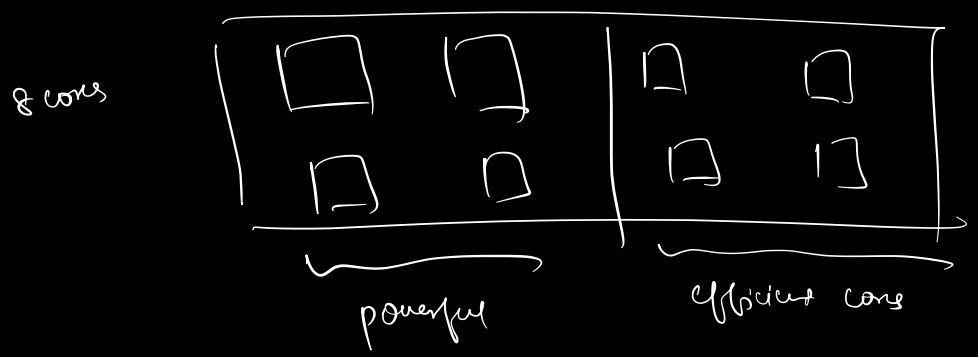
Thread is assigned to a core, which does the work.



⇒ Word processor:

i) Spell check ii) Grammar check iii) Auto save iv) Display

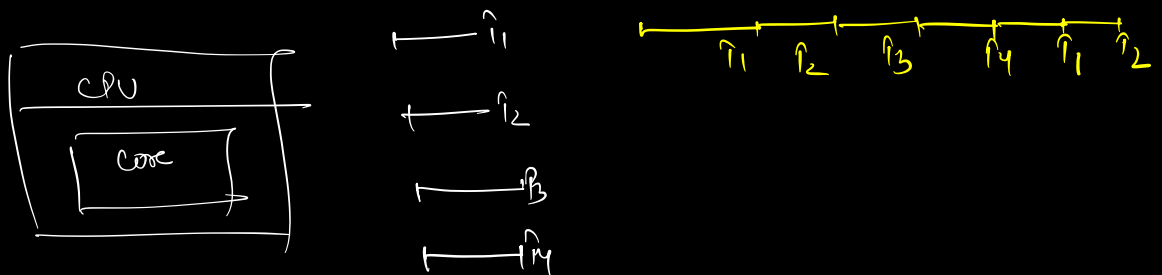
⇒ each core is independent, and each core can execute different threads at the same time.

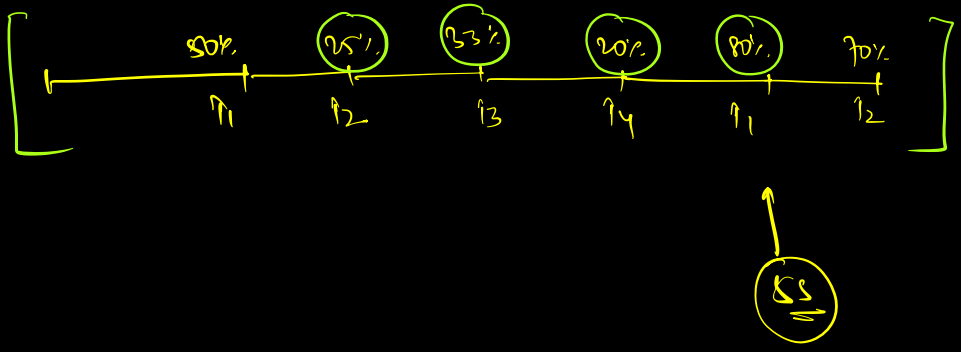


: CONCURRENCY vs PARALLELISM

⇒ **CONCURRENCY:-** when a system can have multiple threads in diff stages of execution at the same time, not necessarily making progress in parallel.

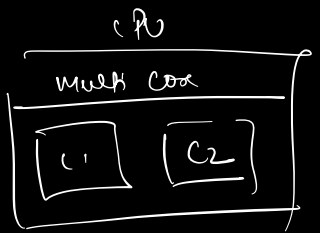
Single core

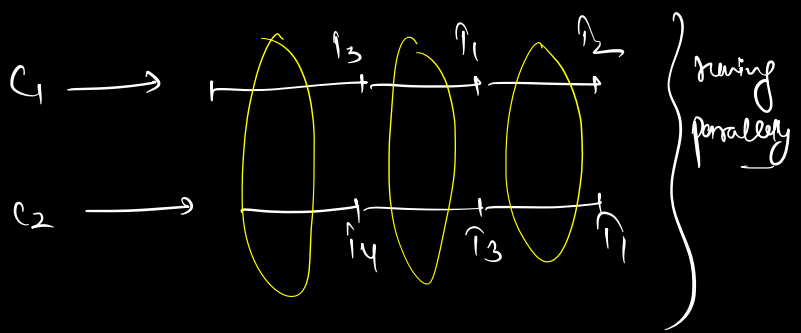




Multiple things happening in the system, but not necessarily parallely.

: PARALLELISM :- CONCURRENCY + progress in parallel





Concurrency → Single core CPU ✓ multi core CPU ✓

Parallelism → Single core CPU ✗ multi core CPU ✓

GPU ⇒ 1000 core ⇒ 1000 parallel computations

CPU 4 cores ⇒ 2.5 GHz → 2.5×10^9

GPU 1000 core ⇒ 400 MHz ⇒ 400×10^6

↓

graphical intensive comp,

mathematic model comp

[repetitive task → brute force]

⇒ How to write multi threaded code :-

* What are the tasks that you want to run in multiple threads ?

[run parallely]

→ Print Hello world from a diff thread (not main thread)

① For every task that needs to run via a separate thread, create a class of it—

```
class HelloWorldPrinter { }
```

② make this class, implement Runnable interface

```
class HelloWorldPrinter implements Runnable { }
```

(3) Implement the task (that needs to be done) inside run() method of the class

```
class HelloWorldPrinter implements Runnable {

    void run() {

        print("hello world")

    }

}
```

(4) At the place where you want to start a new thread, create an object of the task class

```
HelloWorldPrinter hw = new HelloWorldPrinter();
```

And, create the thread with the object

```
Thread t = new Thread(hw);
t.start();
```

↓

Don't call run method

⇒ Why start() and not run()?

start() → 1) Creates a new thread

2) calls the run() on that new thread

run() → 1) Starts executing on the current thread.

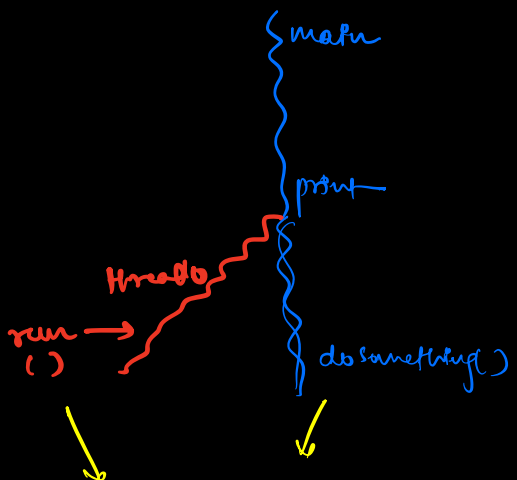
```
doSomething()

main() { }
    print(Hello world)
    t1.start() → run()
    doSomething()
```

print

Hello world!

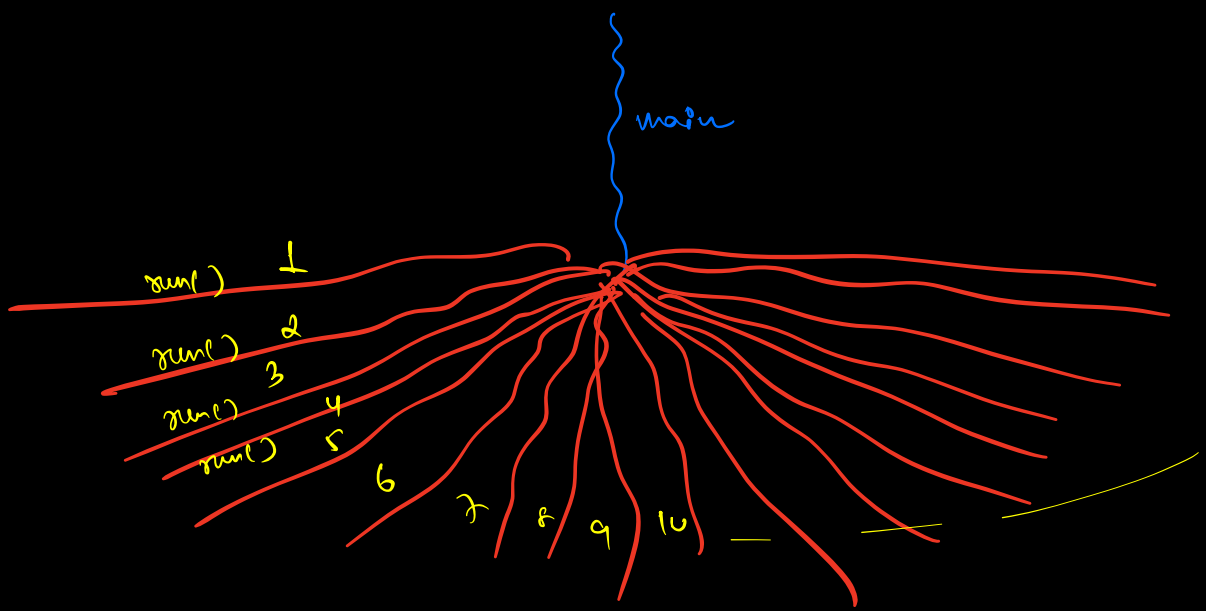
run() { }



Since thread, and main is running parallely, the seq. of call of methods is

Undeterministic

⇒ Print 1 to 100, each no. with a diff thread:-



→ Order of execution will be determined by CPU

→ undeterministic