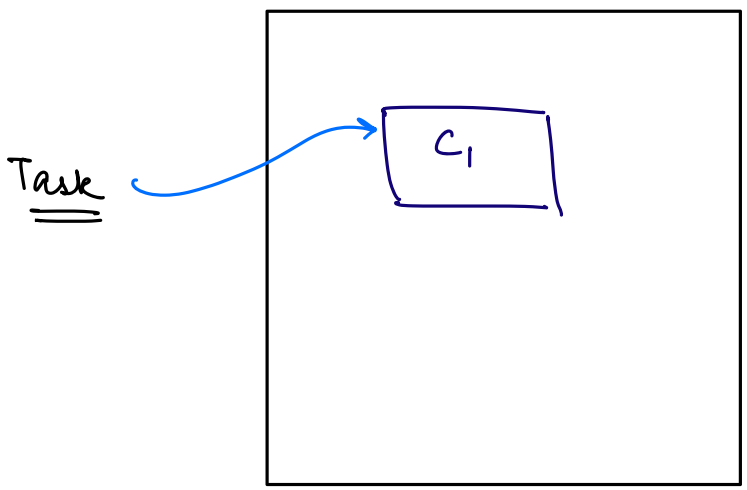
Agenda.

→ Intro to Multithreading
→ Hello World from new thread.
→ Print no's from 1 to 100
→ Executor framework
→ Thread Pool

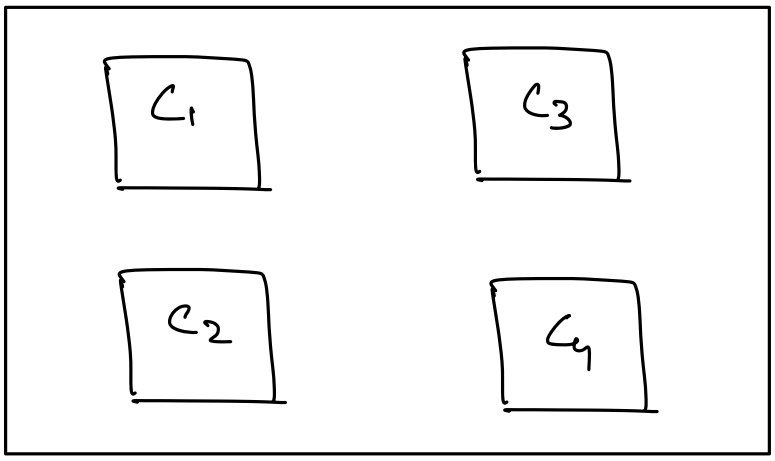
⇒



⇒ Core

⇒ CPU executes the task with the help of Cores

CPU
- Quad Core ⇒ 4
- Octa Core ⇒ 8



⇒ Single Core CPU.

↳ One task at a time.

1 Core ⇒ 1 GHz

⇒ 10^9 operations / sec.

T1 ⇒ 10^6 operations.

10^9 ⇒ 1 sec

1 ⇒ $\frac{1}{10^9}$ sec

10^6 ⇒ $\frac{10^6}{10^9}$ sec

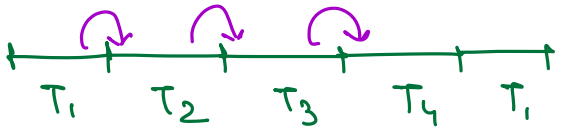
⇒ 10^{-3} sec

⇒ 1 ms.

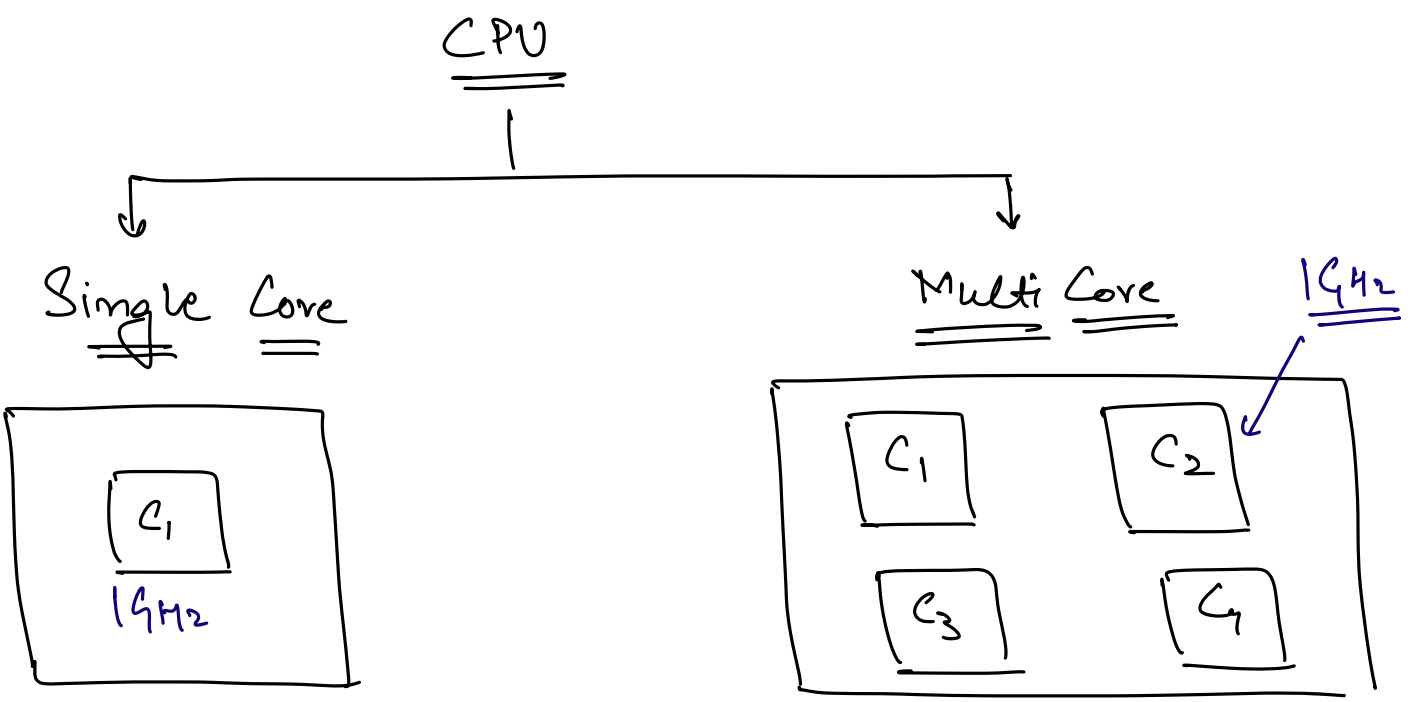
T2

T3

T4



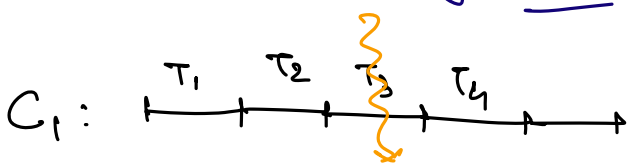
⇒ Context Switching.



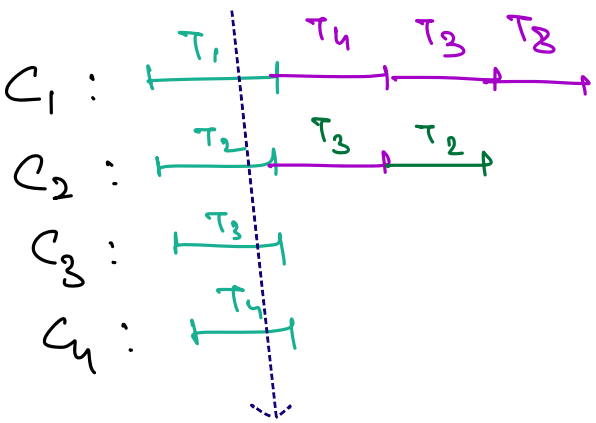
⇒ Only one task a time

⇒ Context Switches ↑

⇒ Concurrent System.



⇒



⇒ Context Switches ↓

⇒ Parallelism.

Quad Core
- 4 ⇒ 1GHz
- 4 ⇒ 4GHz.

GPU.

↳ 1000's of Core.

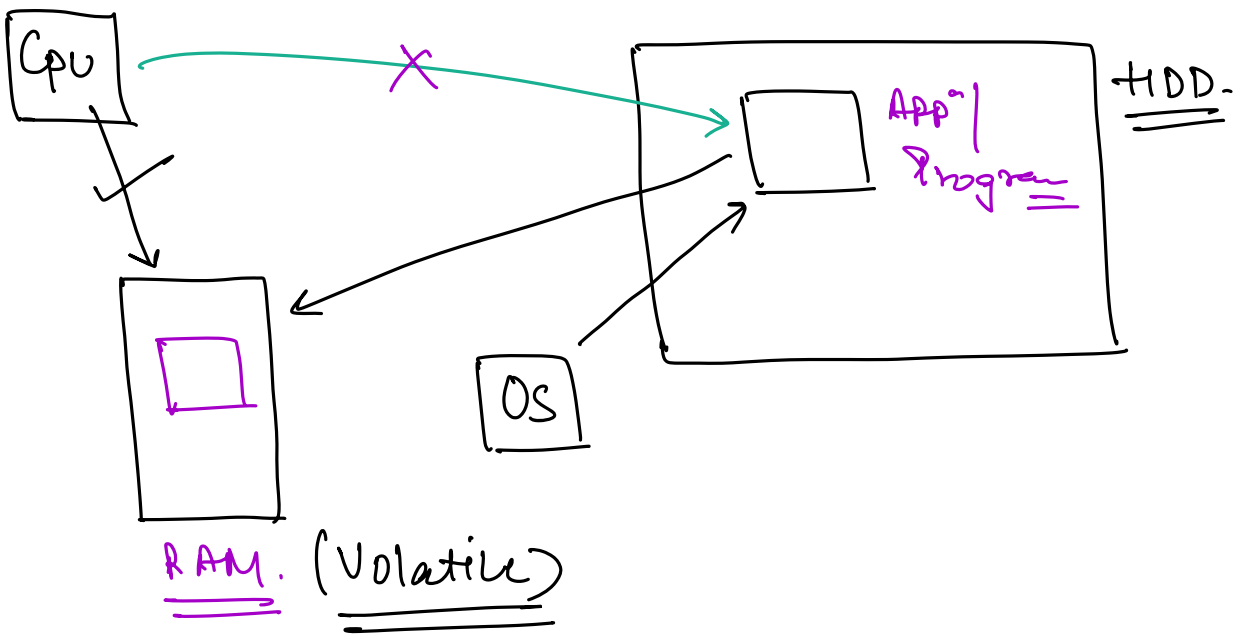
2.5 MHz

Download & Install an Appⁿ.

↓

Executable file-

- .exe
- .apk
- .dmg



CPU can't directly talk to HDD, because of the speed difference.

MS Word | Document Processing App^n

- → Auto save
- → Spell Checker
- → Grammar Check
- → Auto Suggestion

I ma teaching multithreading in JAVA.

⇒ Parallely.

```
int a = 10;
int b = 20
int c = a+b;
Print(c)
```

Program (HDD)

↓

OS

↓

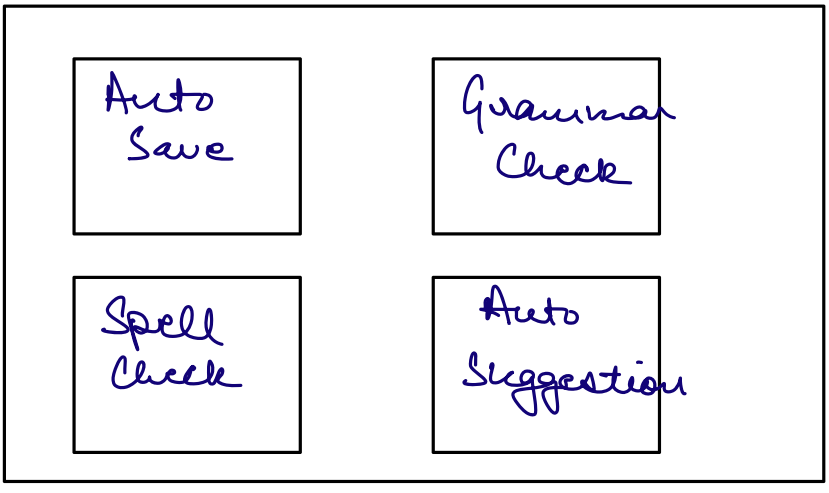
RAM

↓

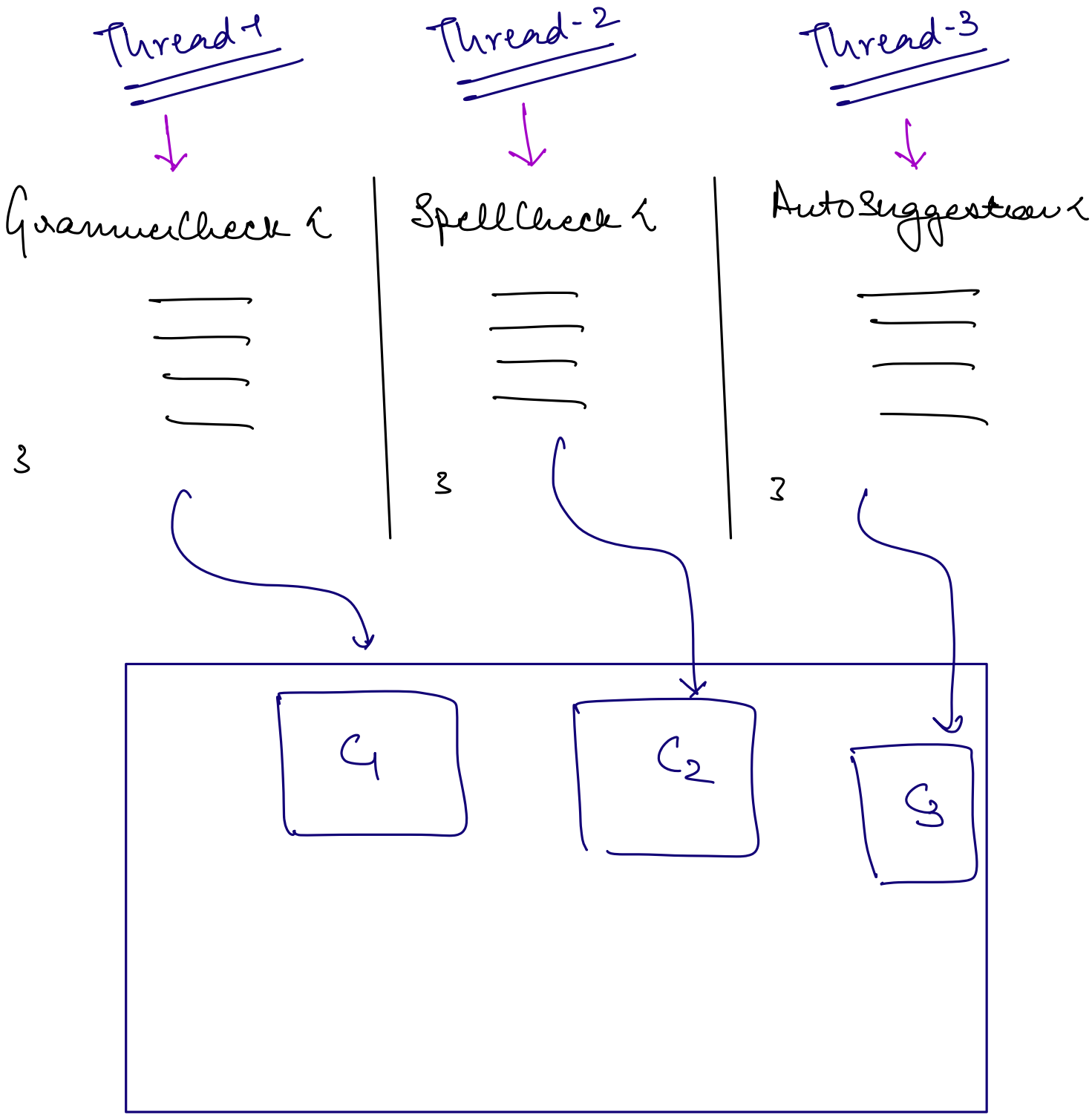
CPU

↓

Process. → Program in execution.



Quad Core



⇒ JAVA Program.

↳ print (Hello World)

↳ Main thread

↳ default thread.

Print Hello World from a new thread.

Steps.

I) Create a task Class that we want to execute in a separate thread.

```
Class HelloWorldPrinter {

}
```

II) Make the task class implement Runnable interface.

```
Class HelloWorldPrinter implements Runnable {

}
```

Runnable → Functional Interface.

III) Override the run() method

Task

```
Class HelloWorldPrinter implements Runnable {

    public void run() {
        Sout("Hello World");
    }
}
```

IV) Create a thread object assign the task to that.

```
HelloWorldPrinter task = new HelloWorldPrinter();
Thread (t) = new Thread(task);
```

V) Start the thread.

```
t.start();
```