Parallelism

- Sequential: the program is broken down into a sequence of instructions that are executed one after the other.
  - we can execute one instruction at any given moment (no overlapping);
  - we use only one **thread**;
- Thread: threads are the software accounted parts to your processor.

Concurrency vs Parallelism

- Concurrency: it describes independent parts of a program to run in an arbitrary order without affecting the outcome. A concurrent application can execute multiple task over an overlapping period.
  - we can start a new task before the previous one is complete, but we cannot perform work on each task simultaneously.
- Parallelism: it means that independent parts of a program can be physically executed at the same time.
  - An application splits its tasks into smaller subtasks which can be processed on multiple CPUs at the same time;
  - assign one core to each task/sub-task.

"A system is said to be *concurrent* if it can support two or more actions in progress at the same time. A system is said to be *parallel* if it can support one or more actions executing simultaneously."

Program that runs in parallel is more rapid but we could have some problems:

- Thread-safety:
  - each task must be independent of each other: a thread may change a variable that another thread has not used yet;
  - only one thread can access to a variable per time;
- it cannot be possible to forecast which thread will access first to the variable: it is not a deterministic process (we have to synchronize the work);
- we can have problem with debug.

It can be very hard to maintain control over a parallel program. One way to keep control over threads and avoid some problems can be to **synchronize** the access:

"synchronized methods enable a simple strategy for preventing thread interference and memory consistency errors: if an object is visible to more than one thread, all reads or writes to that object's variables are done through synchronized methods."

Two or more threads cannot access to the synchronized method at the same time: a thread must wait until the other thread finishes the operation.

We safe the consistency of data but we lose some of the benefit of parallelism...