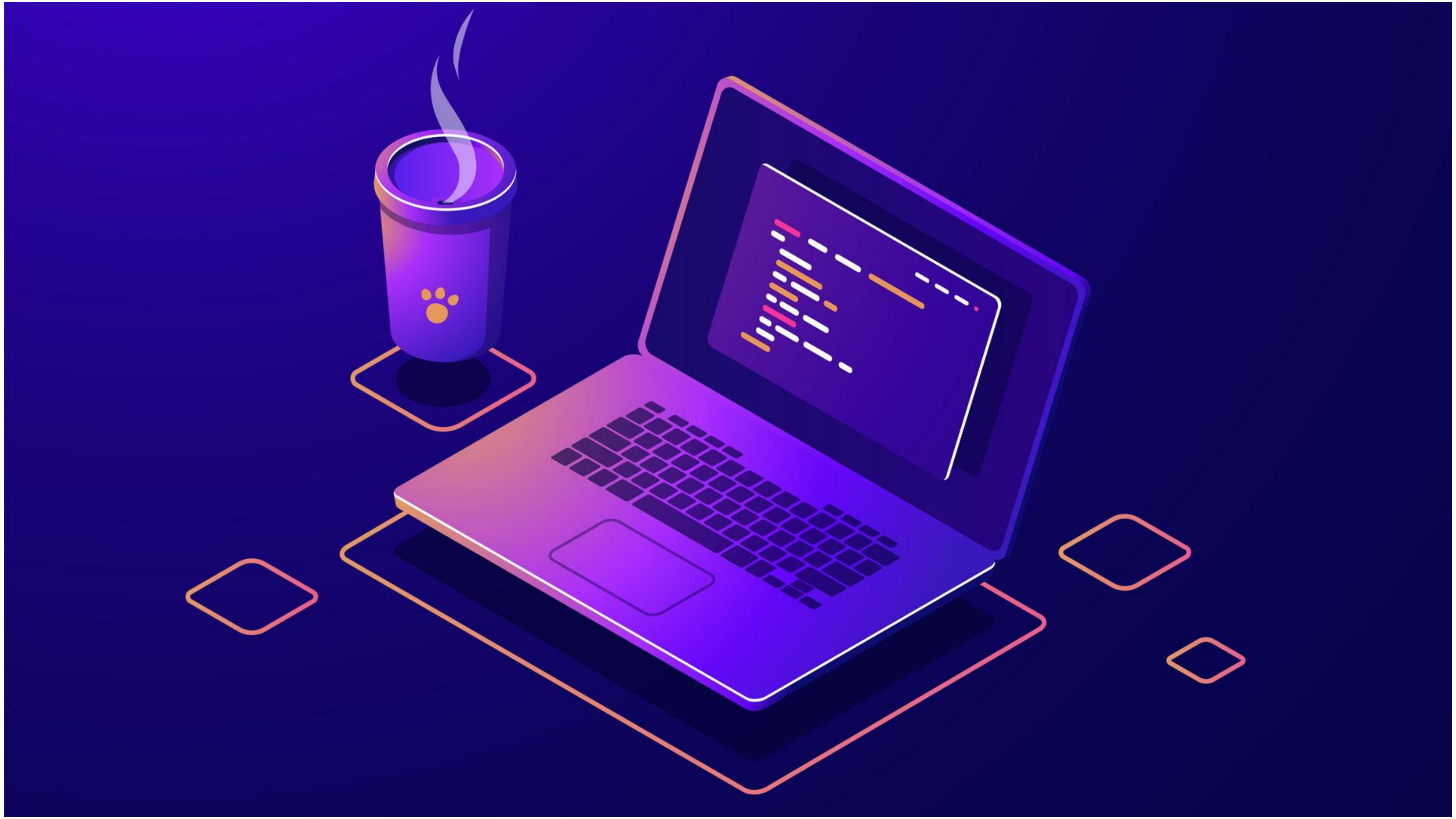
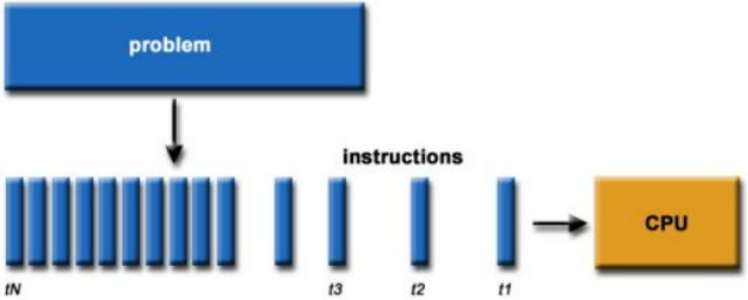
Introduction to Parallelism



Serial Execution

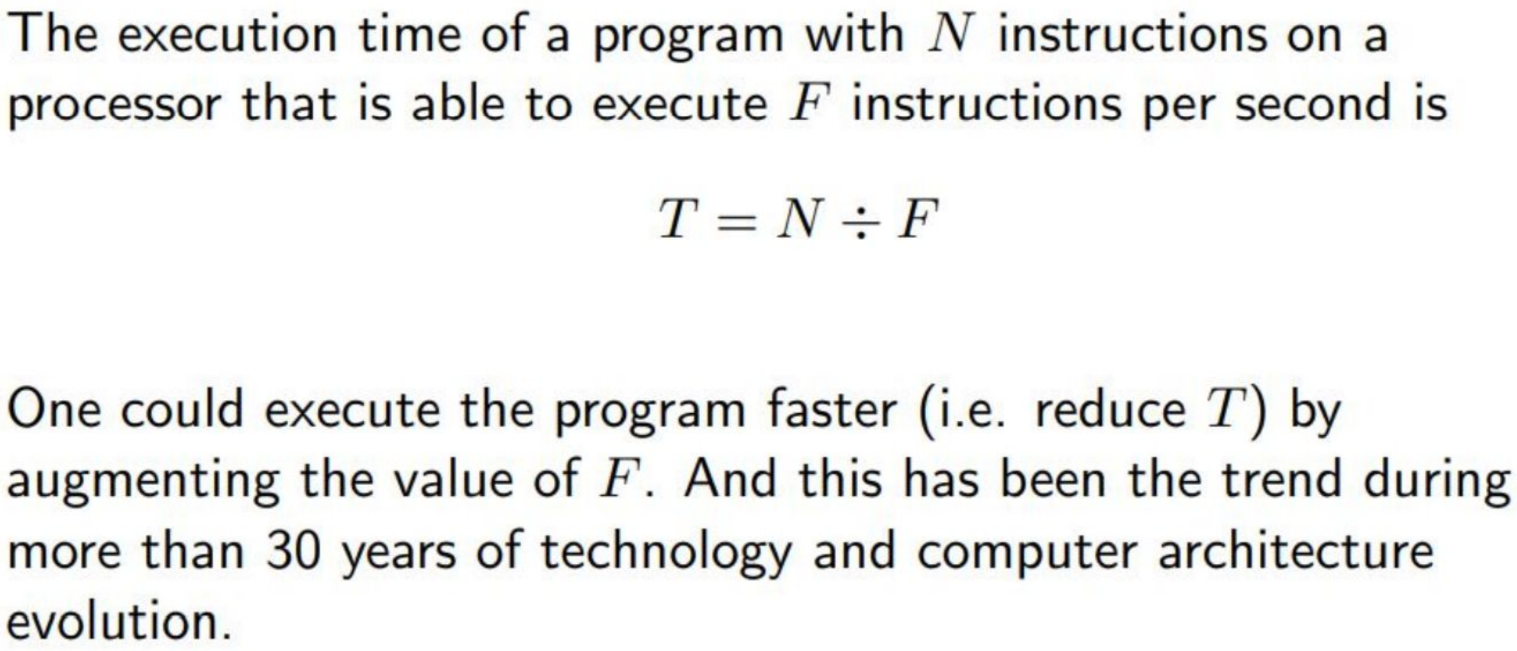
Usually programs are written with a serial execution model in mind

- Program is composed of a sequence of instructions (arithmetic, memory read and write, control, ...) ...
- ... to be run on a computer with a single processor (CPU)



- Instructions are executed one after another, only one at any moment in time

Serial Execution



The execution time of a program with N instructions on a processor that is able to execute F instructions per second is

$$T = N \div F$$

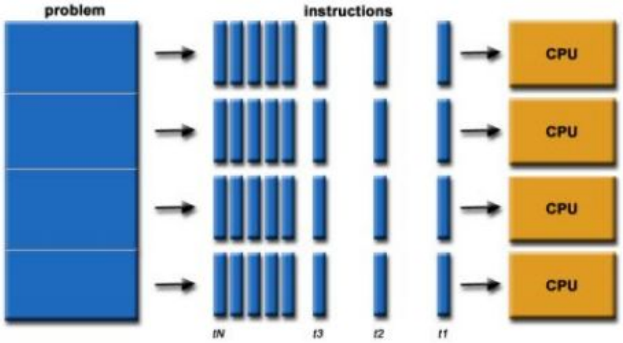
One could execute the program faster (i.e. reduce T) by augmenting the value of F. And this has been the trend during more than 30 years of technology and computer architecture evolution.

Parallel Execution

So another way to reduce the execution time of a program T

$$T = N \div F$$

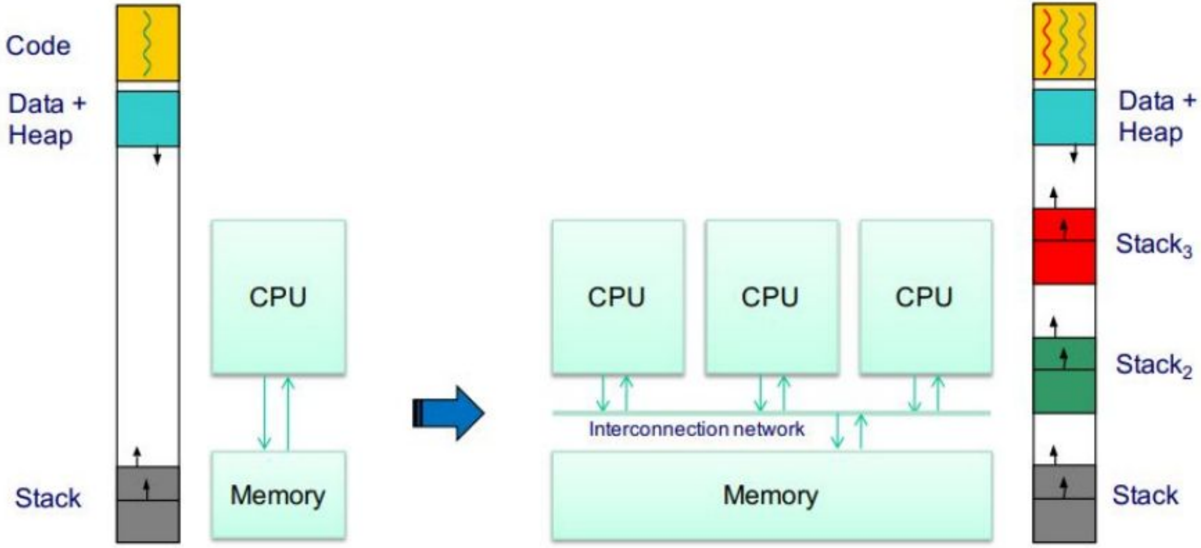
would be to split the program into discrete parts, to be called tasks, and use multiple processors (CPUs) to execute them at the same time



And data?

Parallel Execution

Shared-memory architecture and memory address space

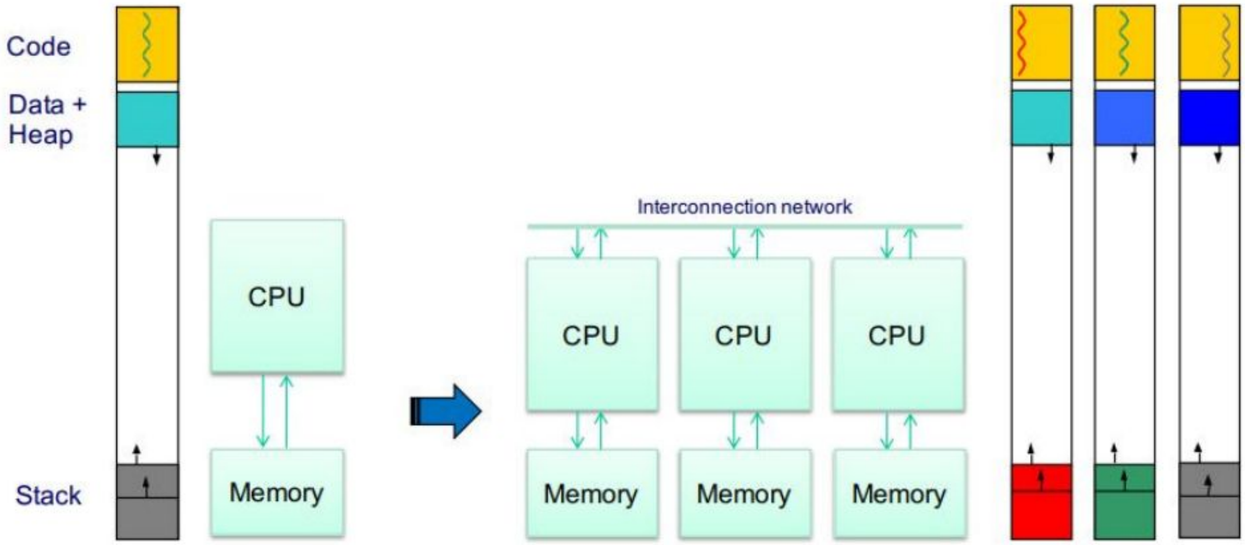


Hardware support for coherent data sharing and tight synchronization

Lucas Bazilio - Udemy

Parallel Execution

Distributed-memory architecture and memory address space



Hardware support for remote data accesses and communication

Lucas Bazilio - Udemy

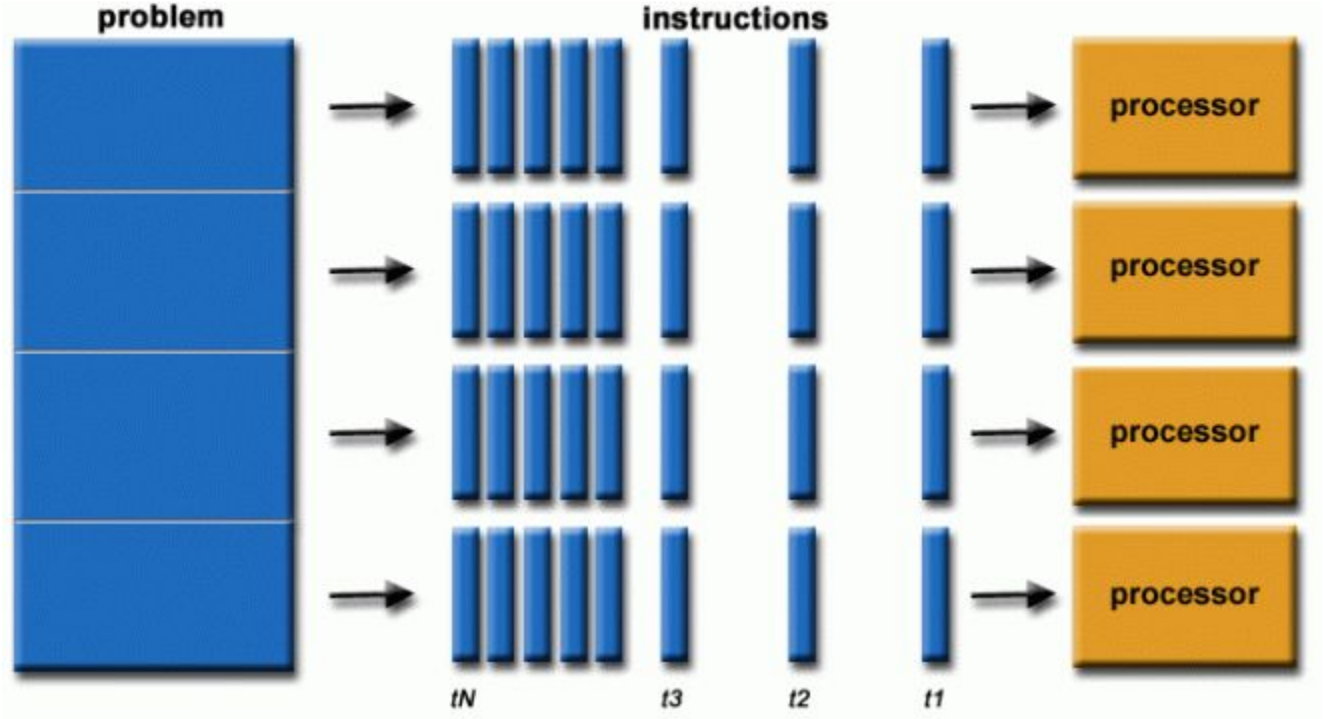
Parallel Execution

Ideally, each processor could receive $\frac{1}{P}$ of the program, reducing its execution time by P

$$T = (N \div P) \div F$$

Need to manage and coordinate the execution of tasks, ensuring correct access to shared resources

So How Can We Achieve This?



Lucas Bazilio - Udemy

Instructor Social Media

Youtube: Lucas Science

Instagram: lucaasbazilio

Twitter: lucasebazilio