

# Concurrency

**01**

About

**02**

Threads

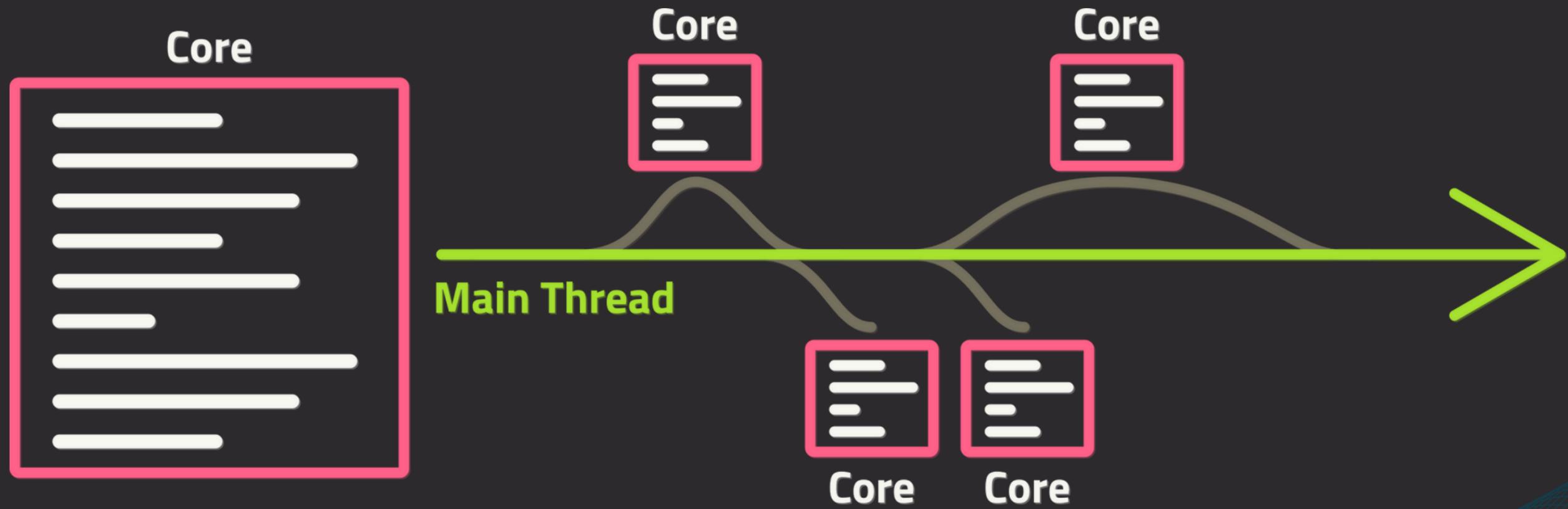
**03**

Async

# Concurrency

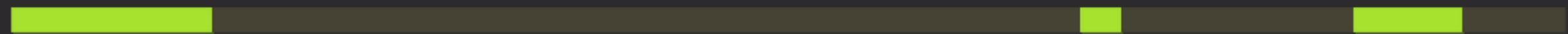
- | Code only executes line-by-line, one line at a time
- | Concurrency allows multiple lines to be executed
- | Two types of concurrent code:
  - | **Threaded:** code runs in parallel based on number of CPU cores
  - | **Asynchronous:** code can pause and resume execution
    - | While paused, other code can resume
- | Go will automatically choose the appropriate concurrency method

# Threaded Execution

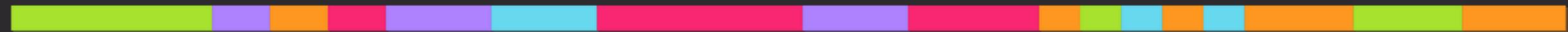
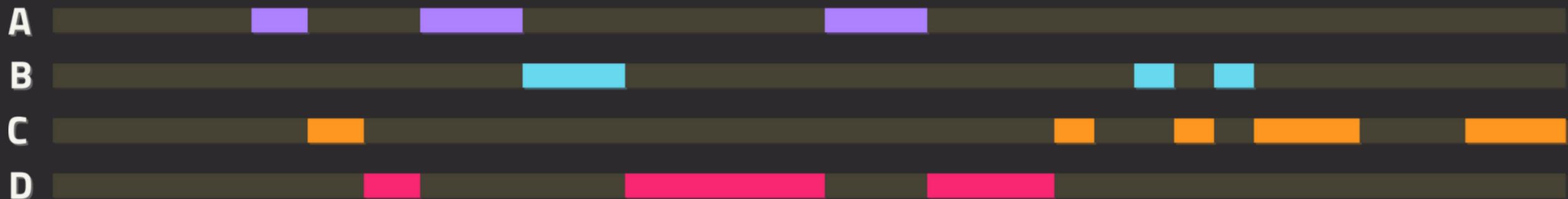


# Asynchronous Execution

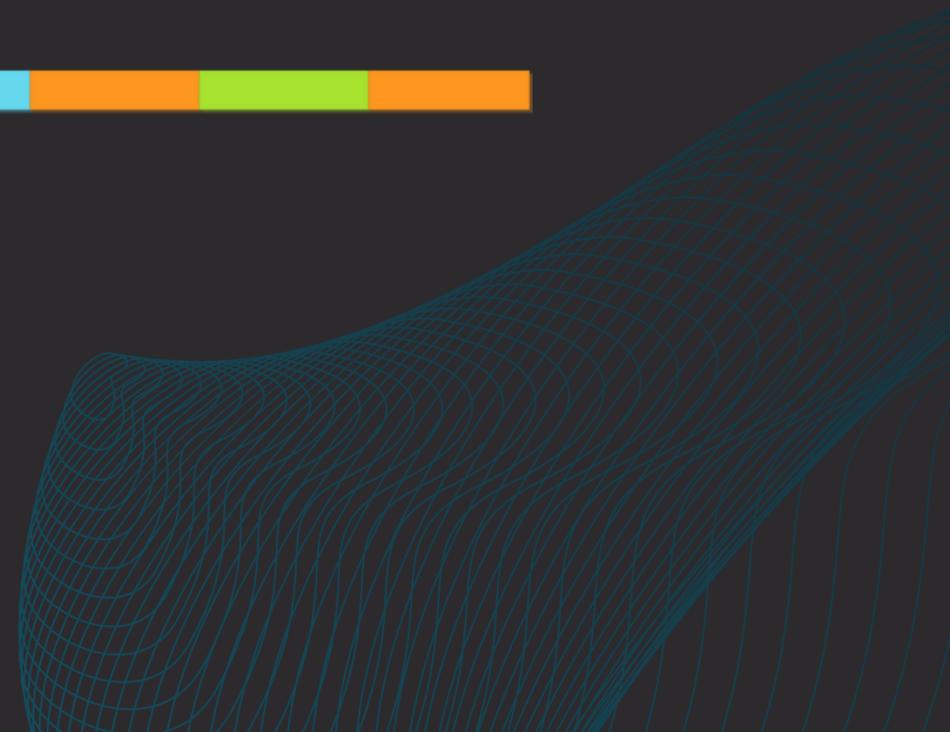
## Main Thread



## Jobs



## CPU Utilization



# Details

- | Single-threaded code runs **deterministically**
  - | Each run will produce the same result
- | Concurrent code runs **non-deterministically**
  - | Code no longer executes line-by-line in a predefined order
    - | Cannot rely on results being the same each program run
- | Extra care should be taken to ensure results are in order / sorted properly
  - | Accomplished using **synchronization** or by checking the final results in a single thread

# Recap

- | Concurrent code allows full utilization of available compute resources
- | Go automatically abstracts threads and asynchronous operations
  - | Threaded code is used to make parallel computations on cores
  - | Asynchronous code may be paused/resumed and is used for waiting on resources (like networks)
- | Concurrent code runs non-deterministically
  - | **Synchronization** or other techniques are required to ensure proper program behavior