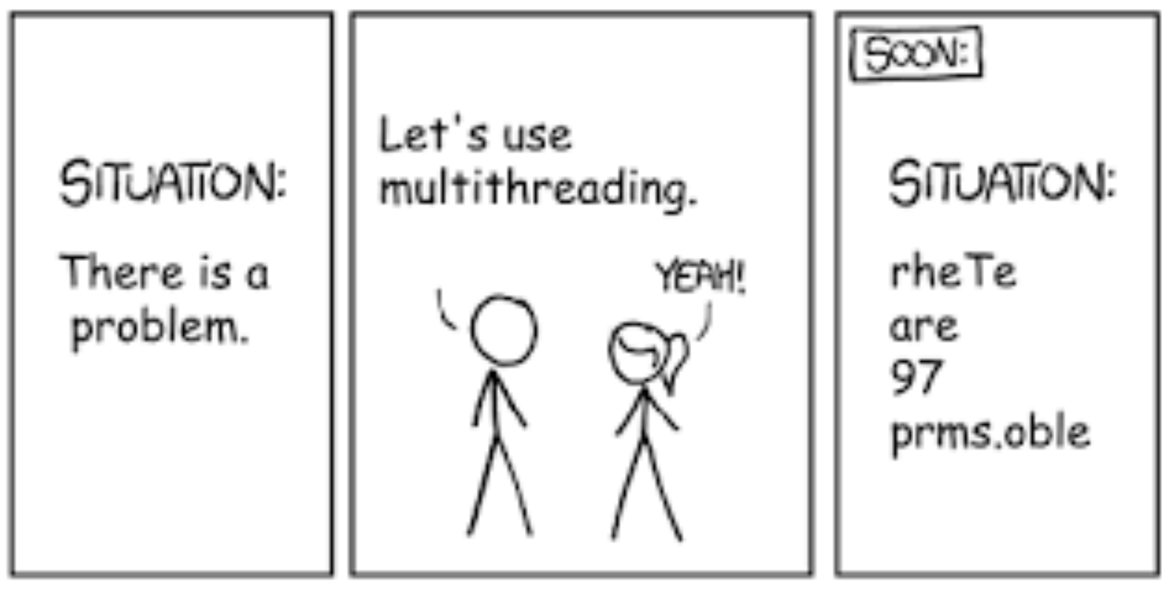
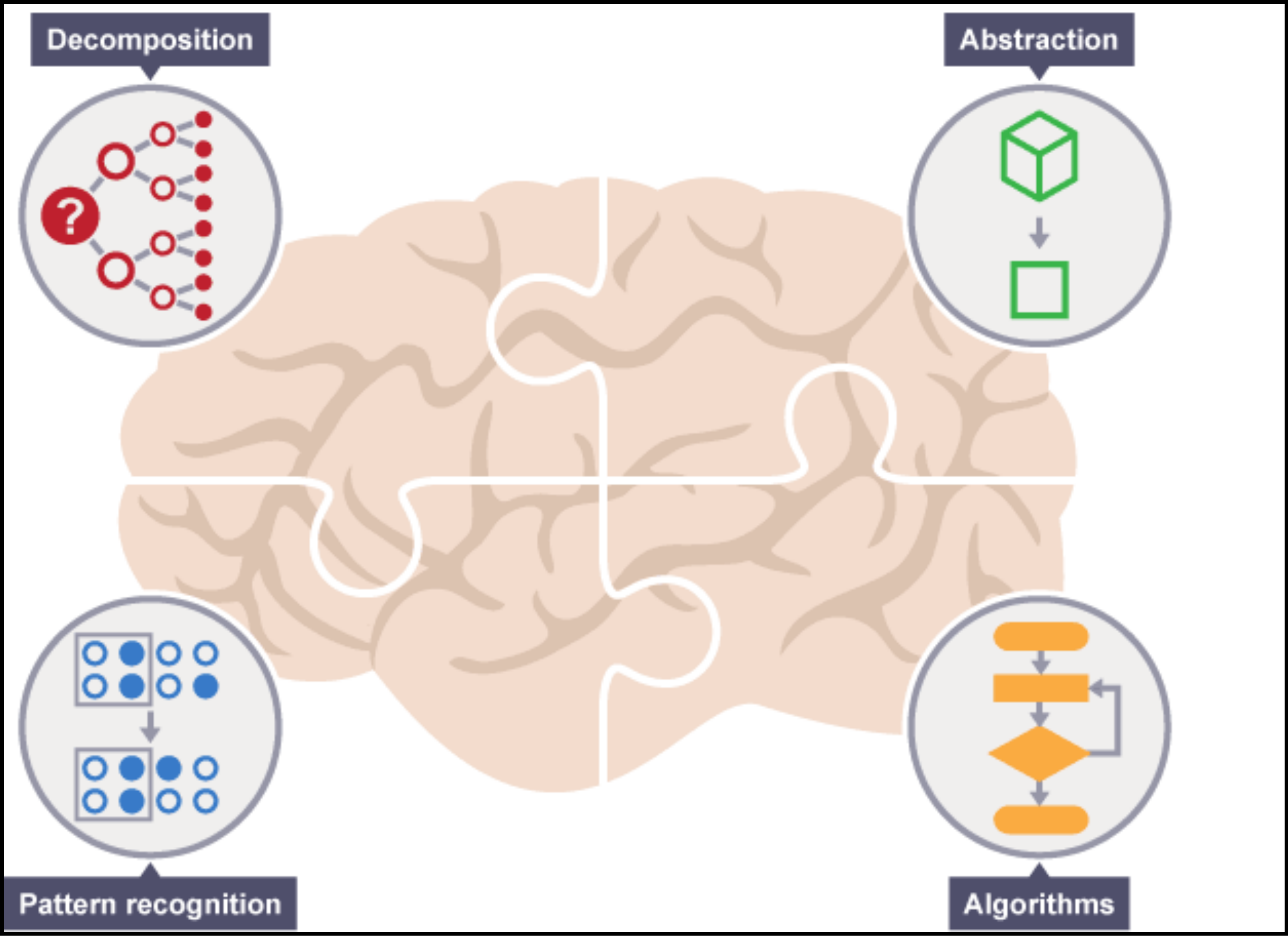
Computer Science: The Good Parts

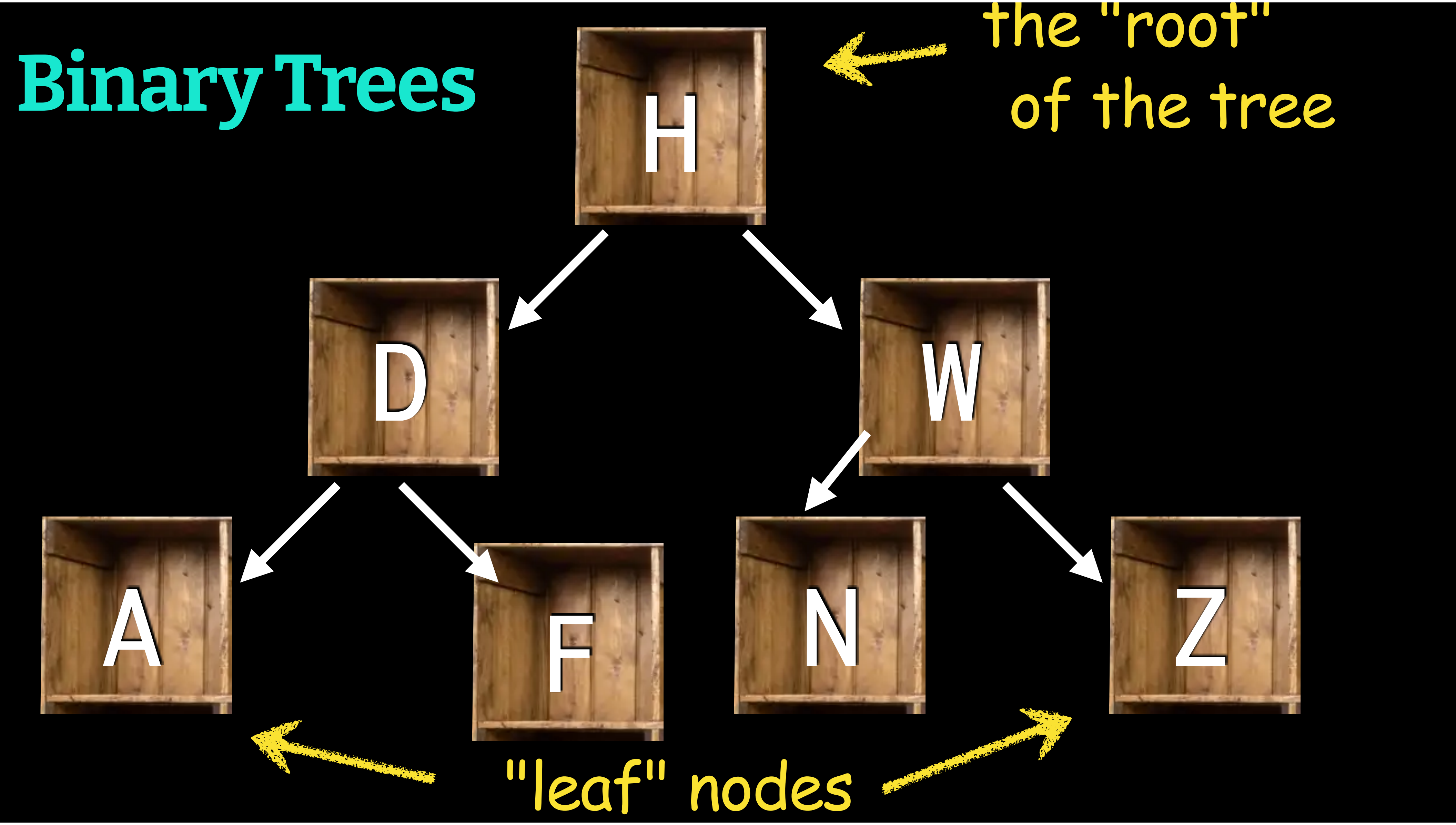
A Practical Journey for
Early-Career Developers

Part 3



Computational Thinking





Binary Trees

the "root" of the tree

"leaf" nodes

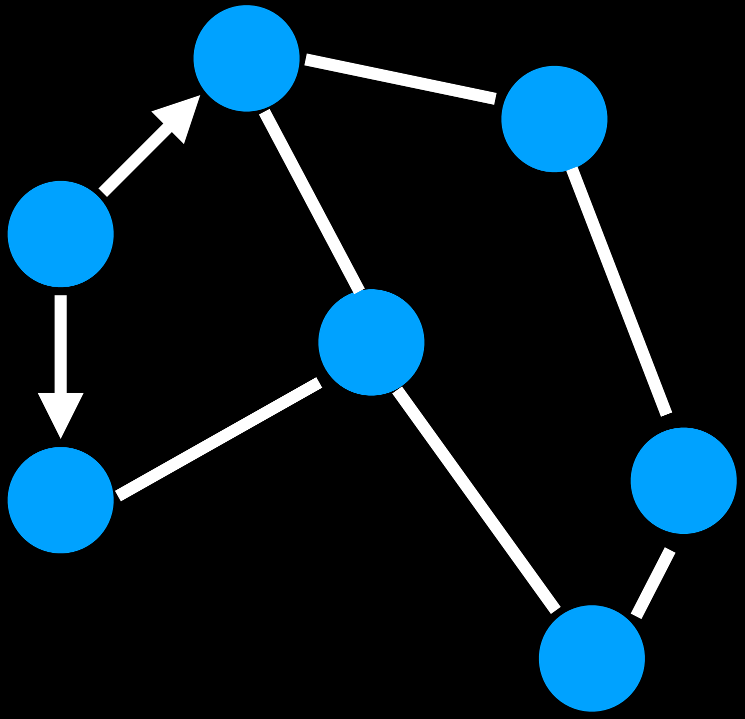
Recursion, Part 2

Graphs

In computer science, a *graph* is a connected set of nodes (i.e. a network) that has no special root node and no restrictions on the connections between them.

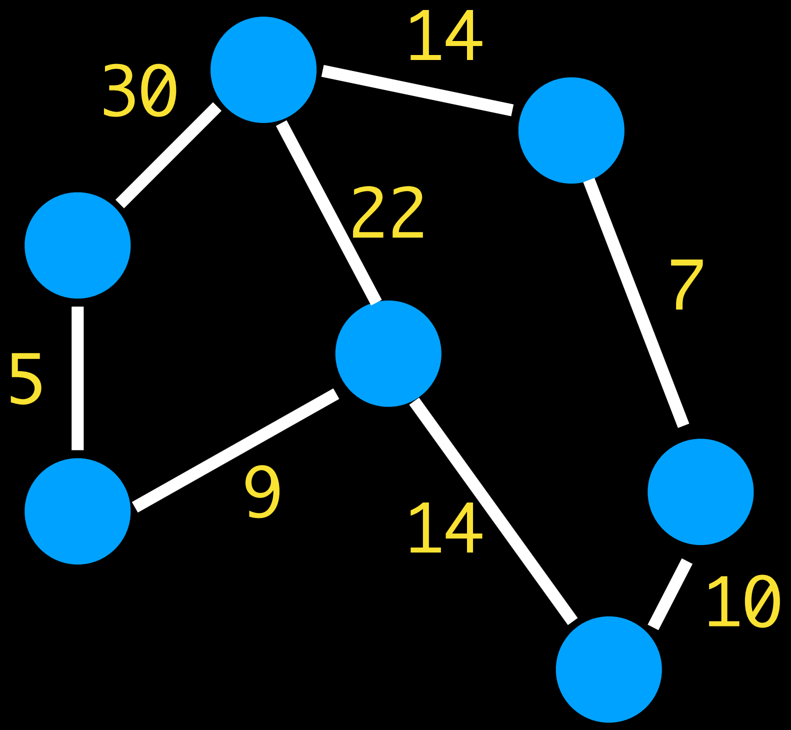
Graphs

- Connected or Disconnected
- Directed or Undirected
- Cyclic or Acyclic
- Weighted or unweighted



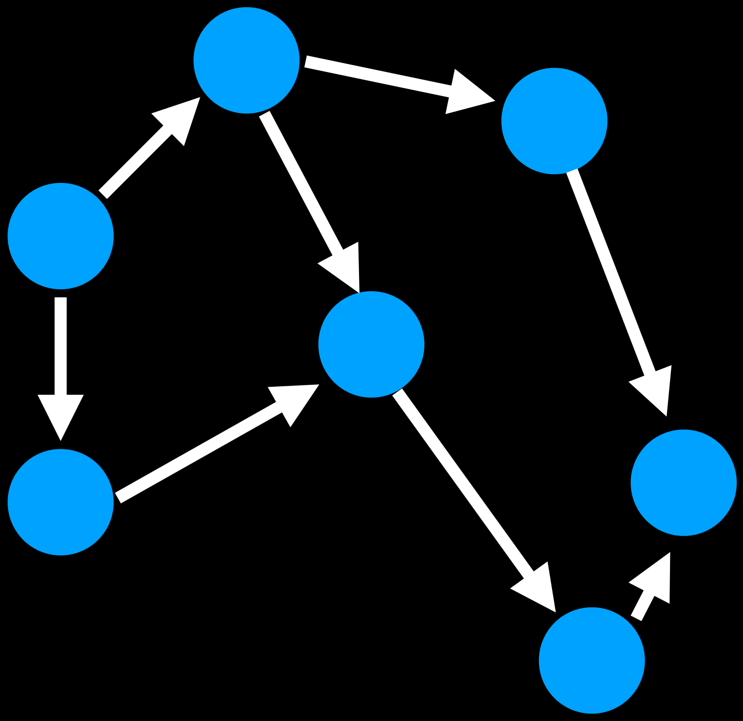
Graphs

- Connected or Disconnected
- Directed or Undirected
- Cyclic or Acyclic
- Weighted or unweighted



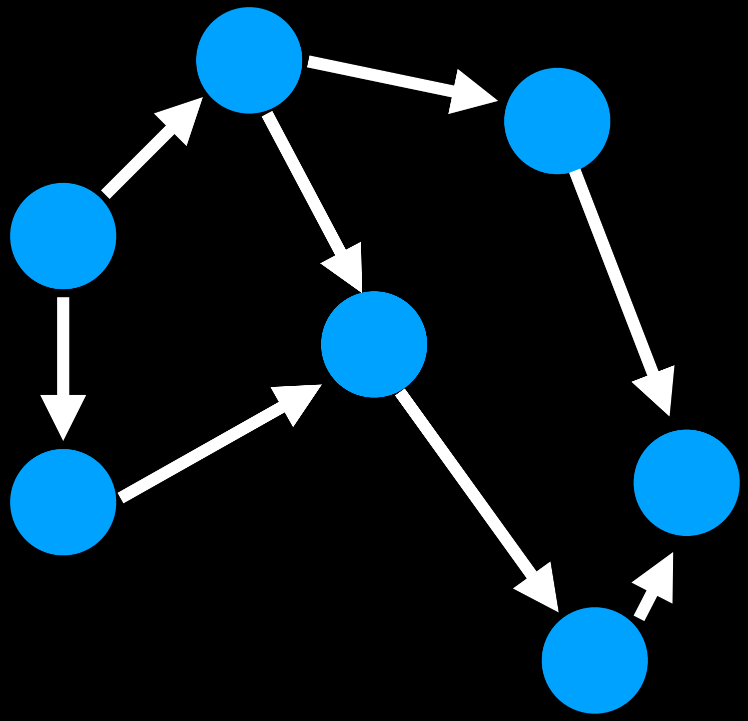
Graphs

- Connected or Disconnected
- Directed or Undirected
- Cyclic or Acyclic
- Weighted or unweighted

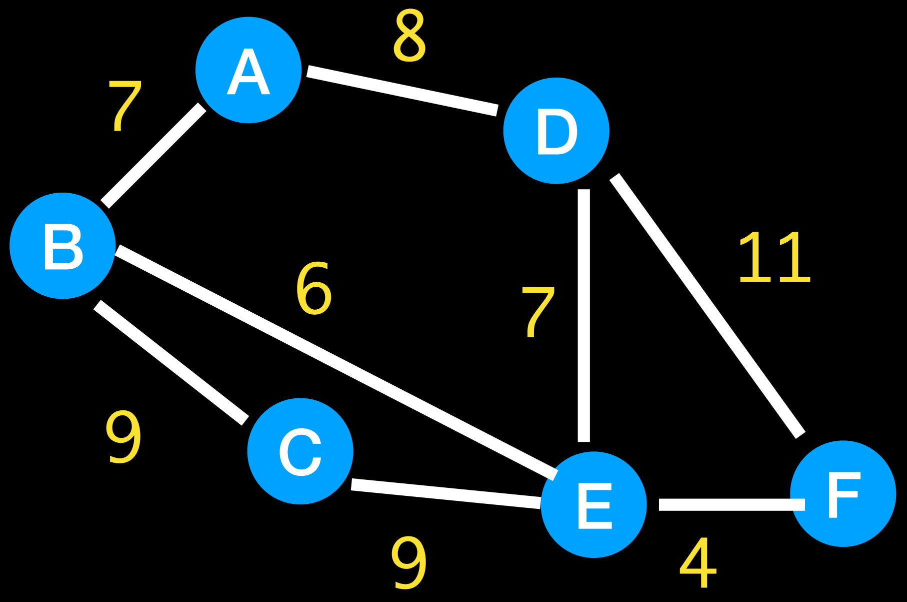


Algorithms

- Reachability
- Shortest Path
- Efficiency Path
- lots more



Djikstra's Algorithm



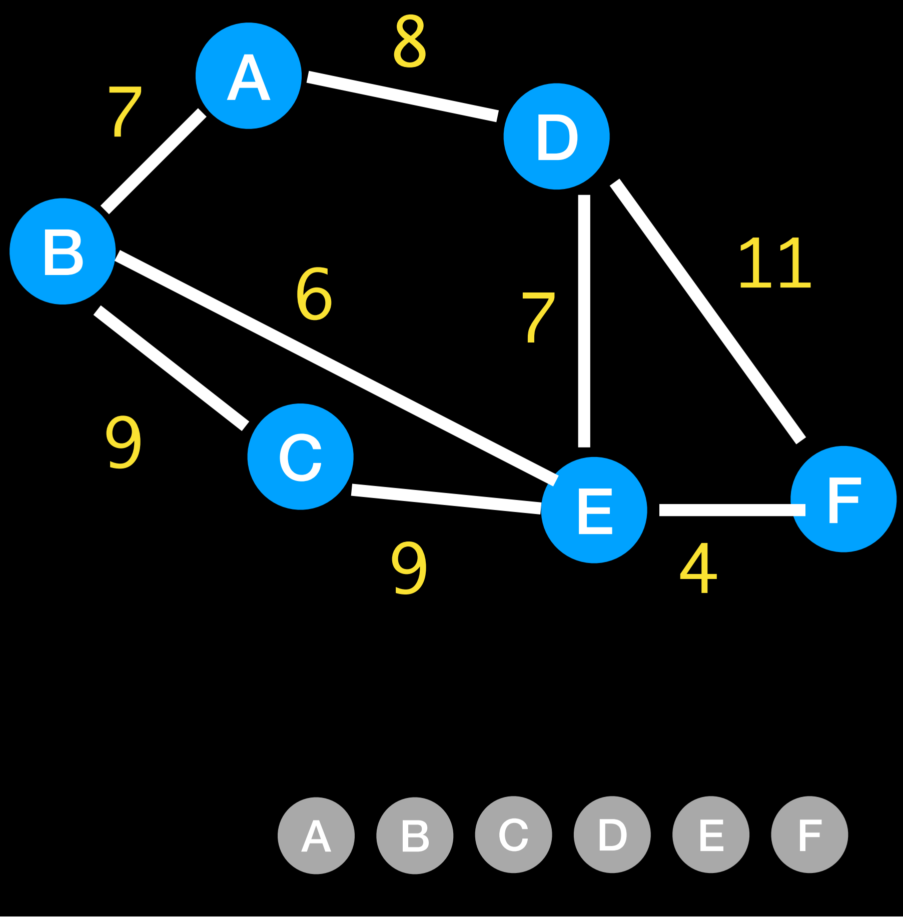
How would we travel from A to F?

How would we travel from (any node) to (any other node)?

Jot down some pseudocode for your thoughts.

We will resume at 10:20 CT

Djikstra's Algorithm



Node	Distance
A	
B	
C	
D	
E	
F	

Cyclomatic Complexity

McCabe, 1976

Start by constructing a graph of your code.

Cyclomatic Complexity

$M = E - N + 2P$

E = # edges
N = # nodes
P = # external connections (exit points)

Closures and Bindings

```
fruit = "apple"
```

This code defines a binding between the name "fruit" and the memory address for "apple".

25089
"apple"
fruit

Closures and Bindings

```
def display_favorites
  fruit = "apple"
  color = "purple"
  puts fruit
  puts color
end
```

Bindings are specific to a given scope.

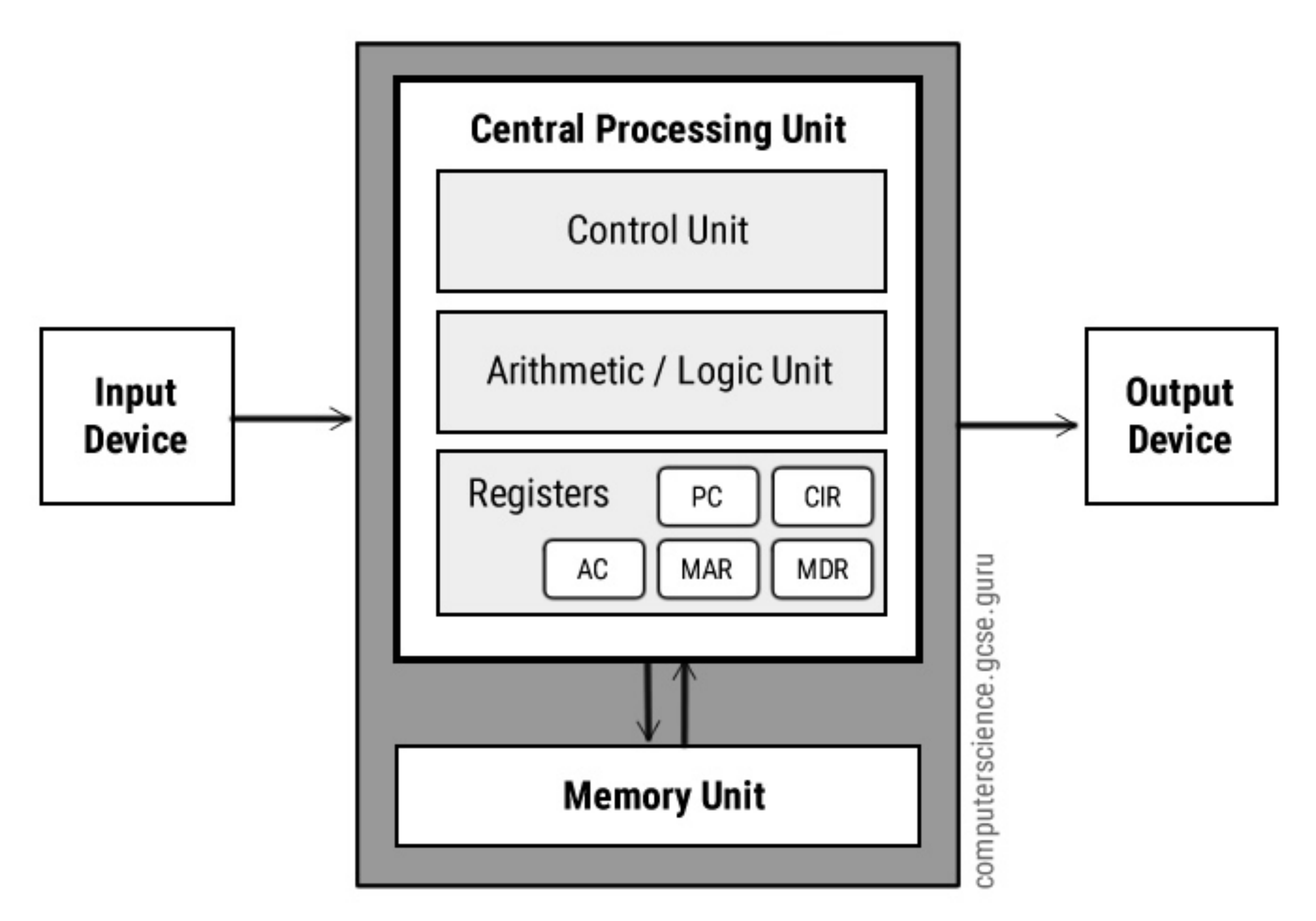
Closures and Bindings

Bindings demo

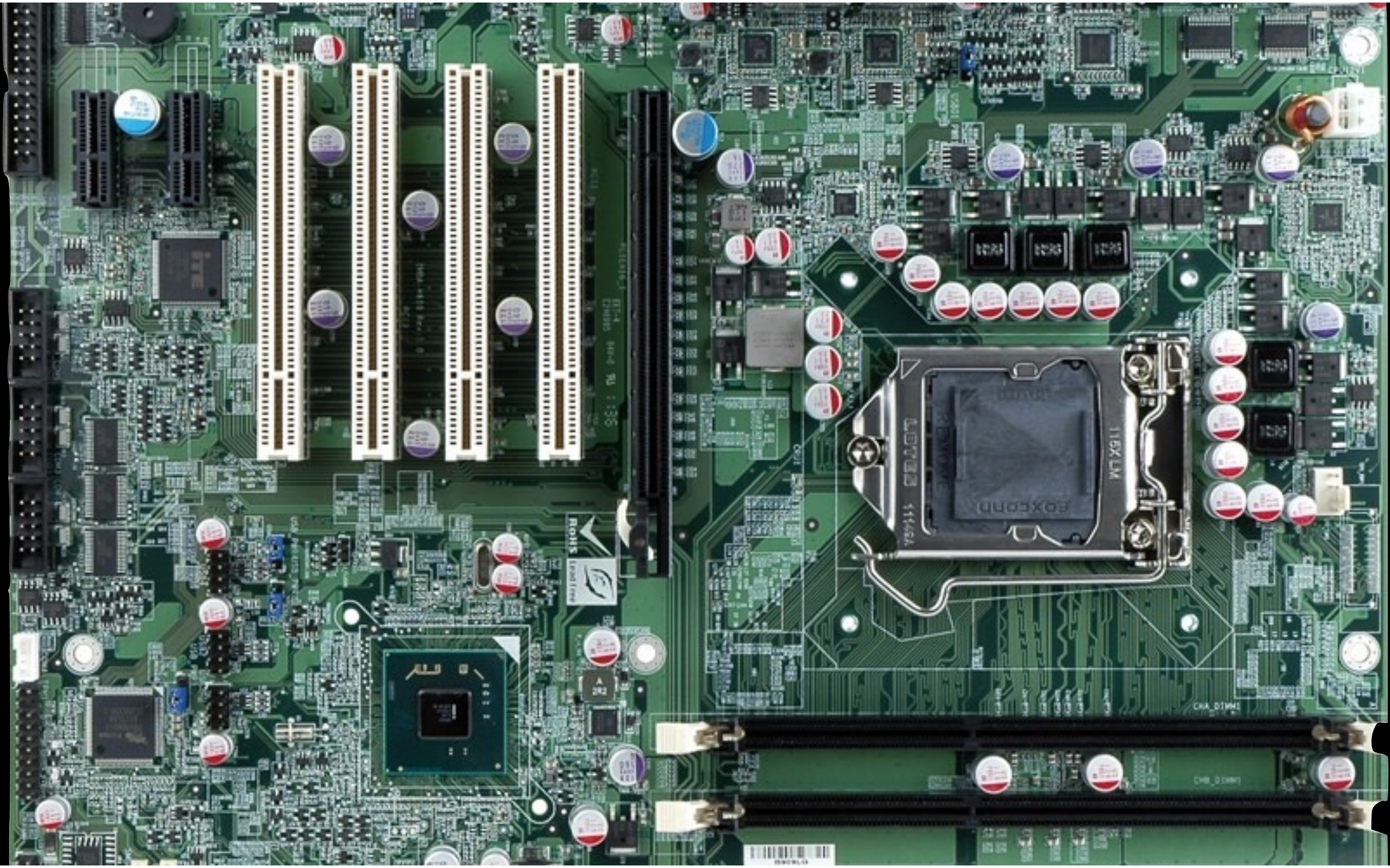
Closure demo

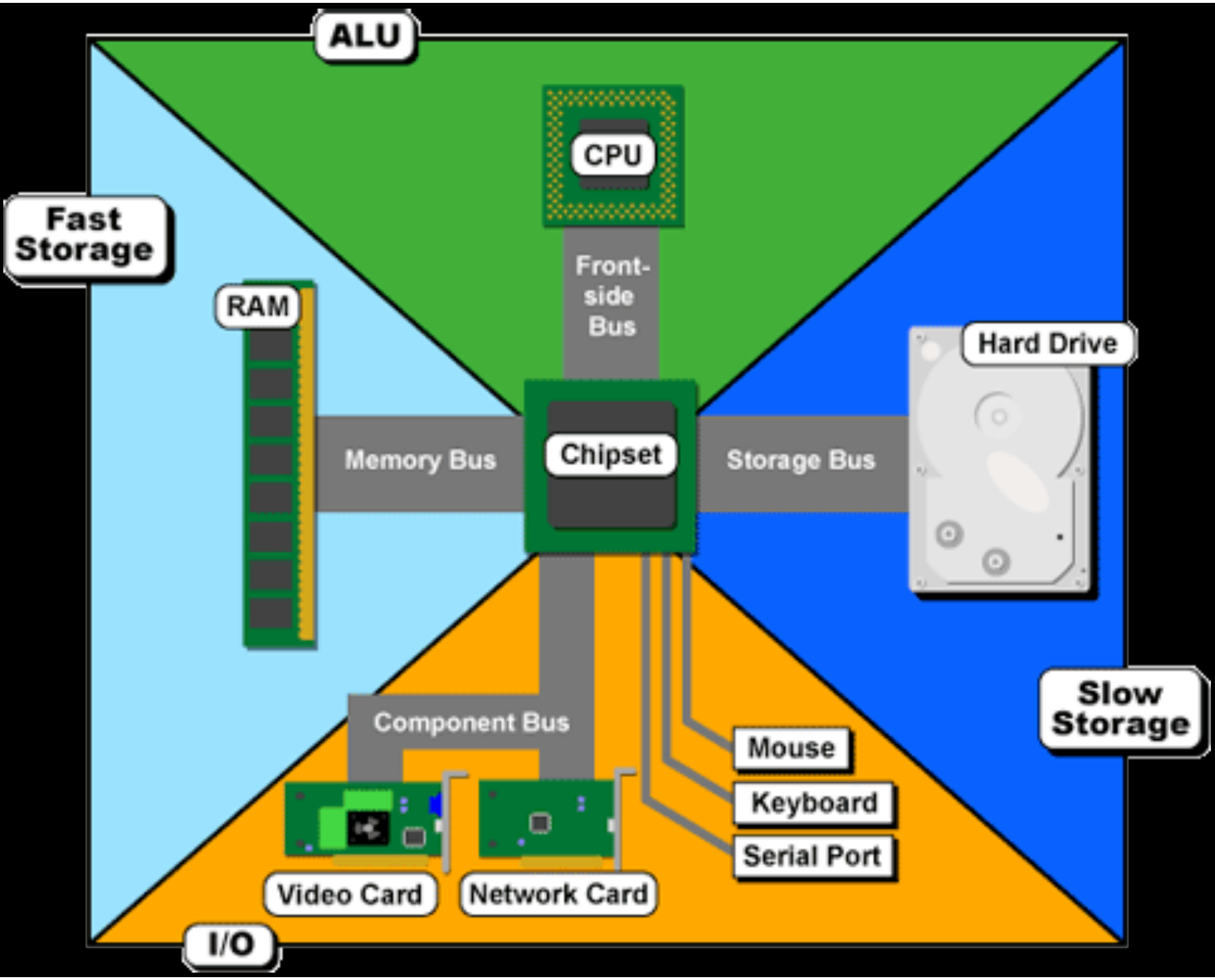
Computer Architecture

The von Neumann Architecture

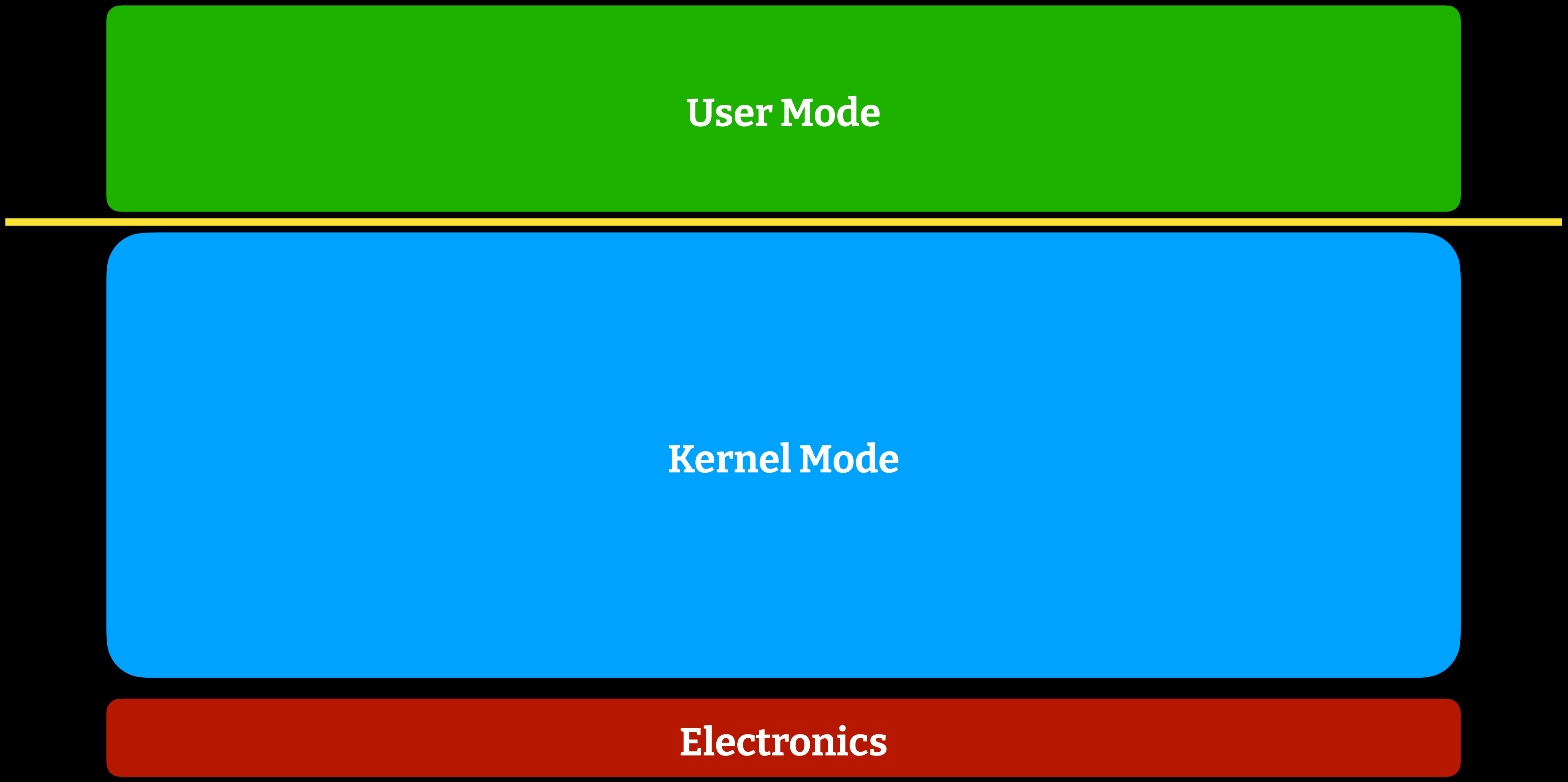


computerscience.gcse.guru




ALU
CPU
Front-side Bus
Fast Storage
RAM
Memory Bus
Chipset
Storage Bus
Hard Drive
Slow Storage
Component Bus
Mouse
Keyboard
Serial Port
Video Card
Network Card
I/O

Operating Systems

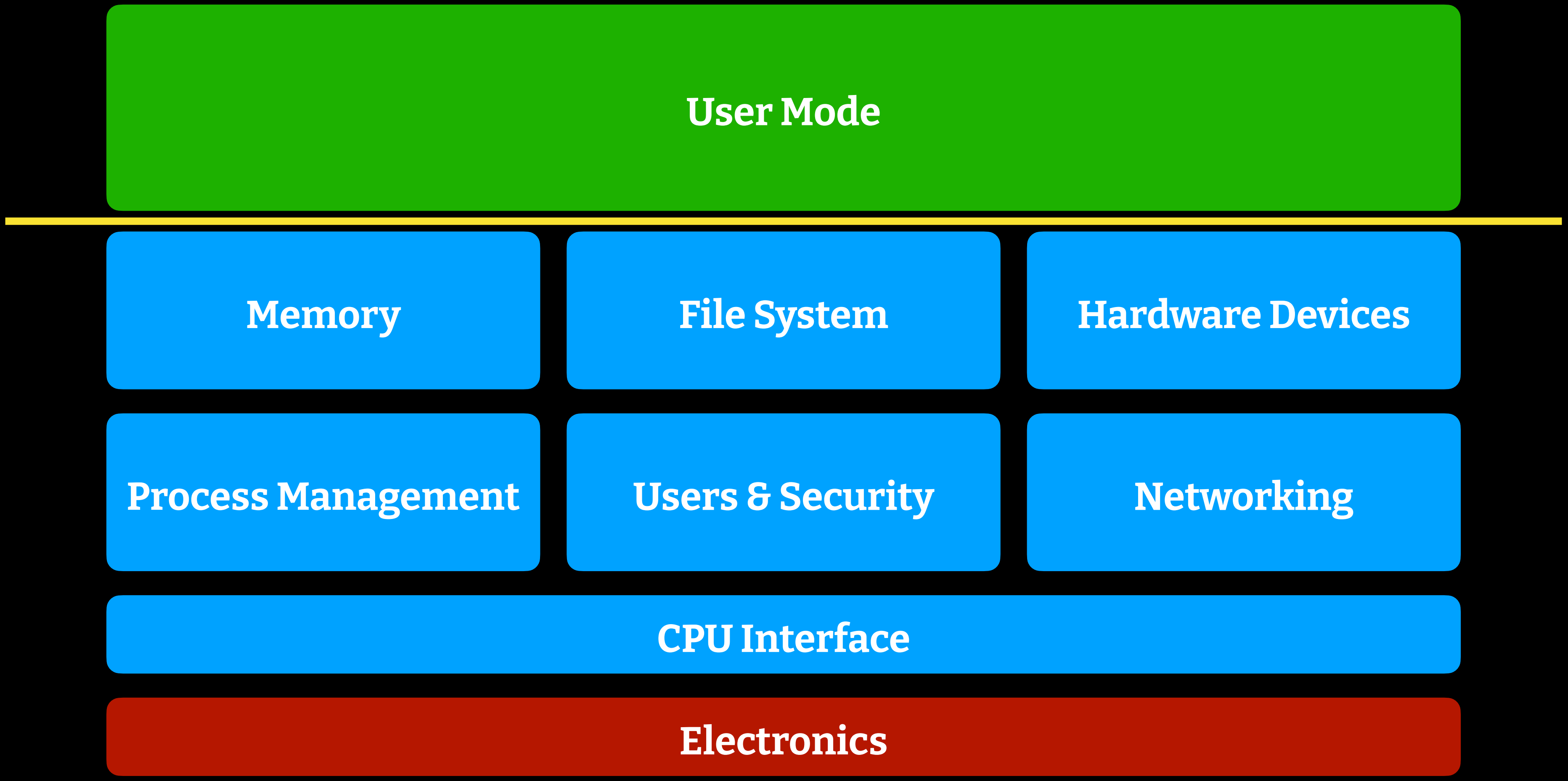


User Mode

Kernel Mode

Electronics

Operating Systems



User Mode

Memory

File System

Hardware Devices

Process Management

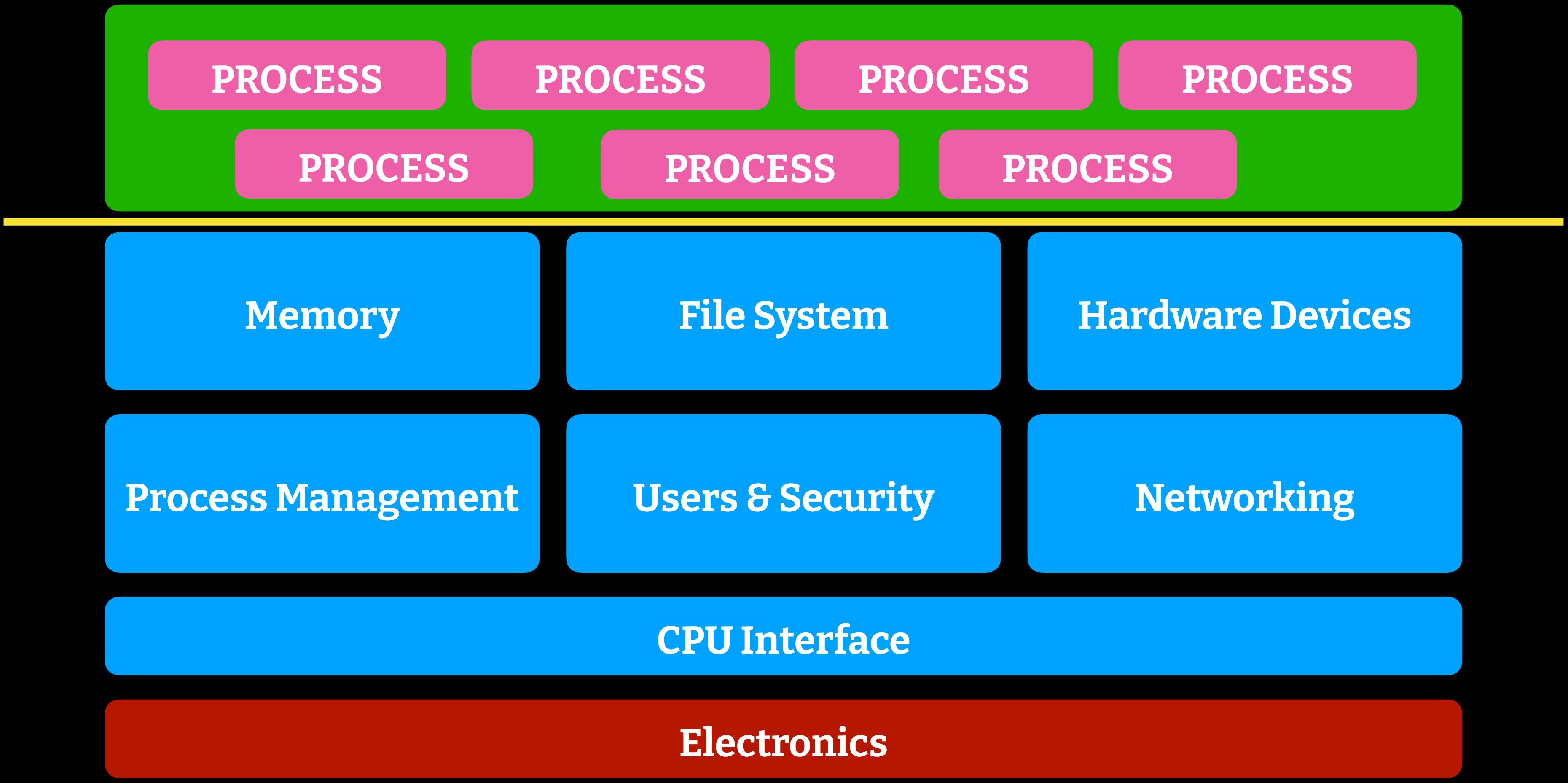
Users & Security

Networking

CPU Interface

Electronics

Operating Systems



PROCESS PROCESS PROCESS PROCESS

PROCESS PROCESS PROCESS

Memory

File System

Hardware Devices

Process Management

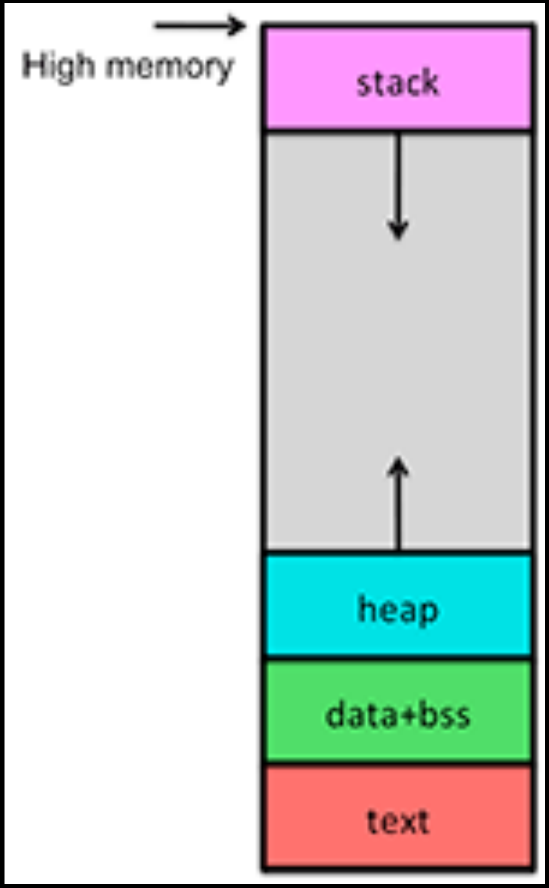
Users & Security

Networking

CPU Interface

Electronics

Process Model



Every process has:

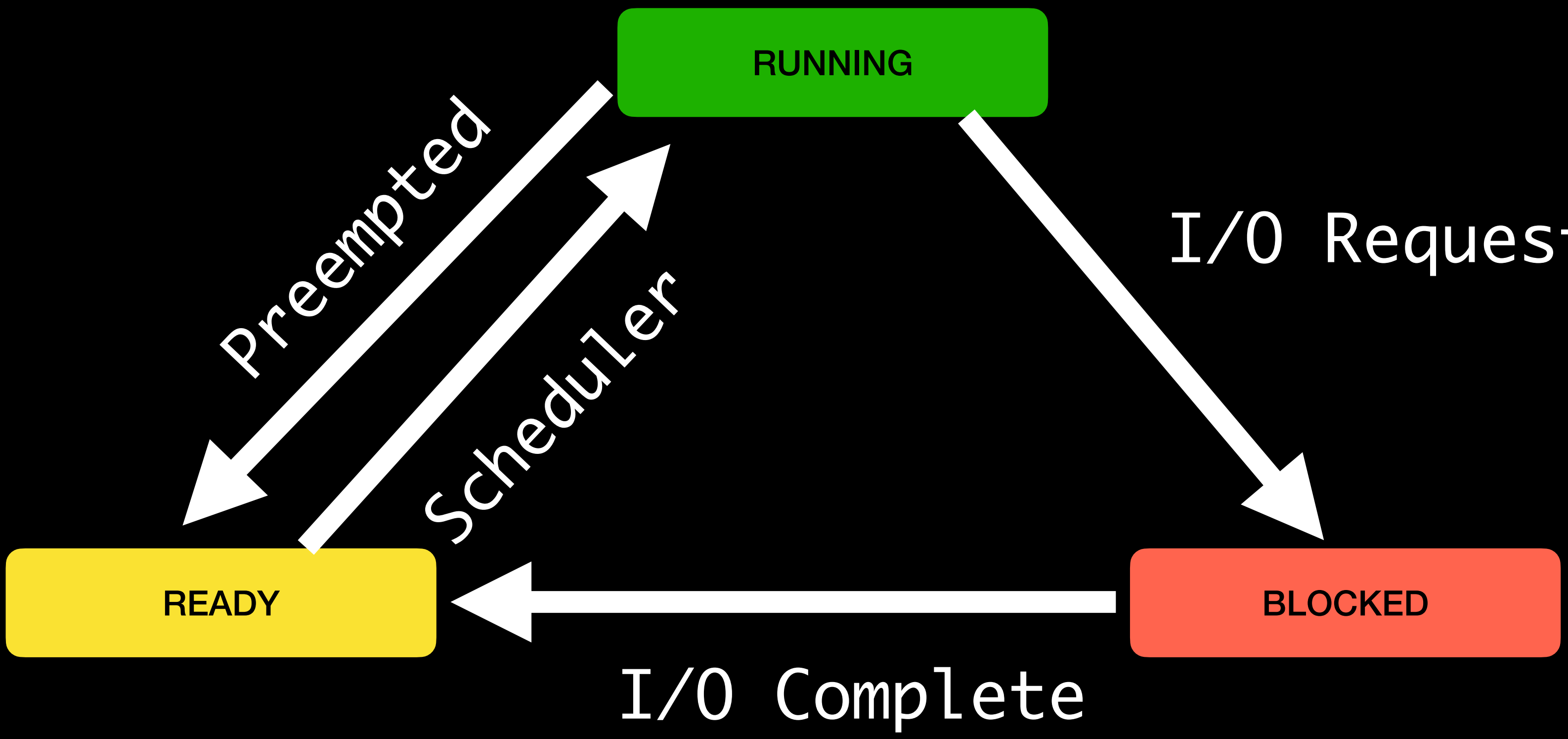
ID

Parent ID

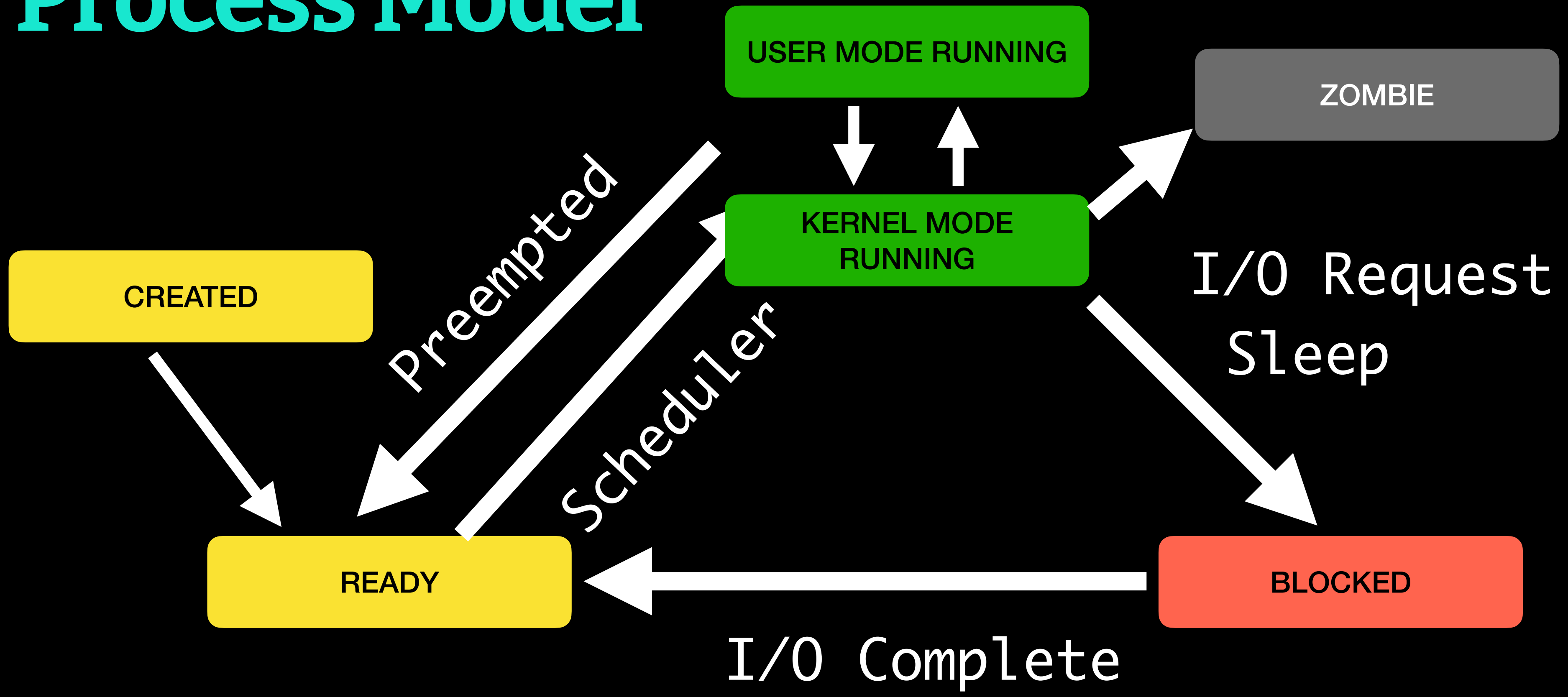
CPU ID

1 or more Threads

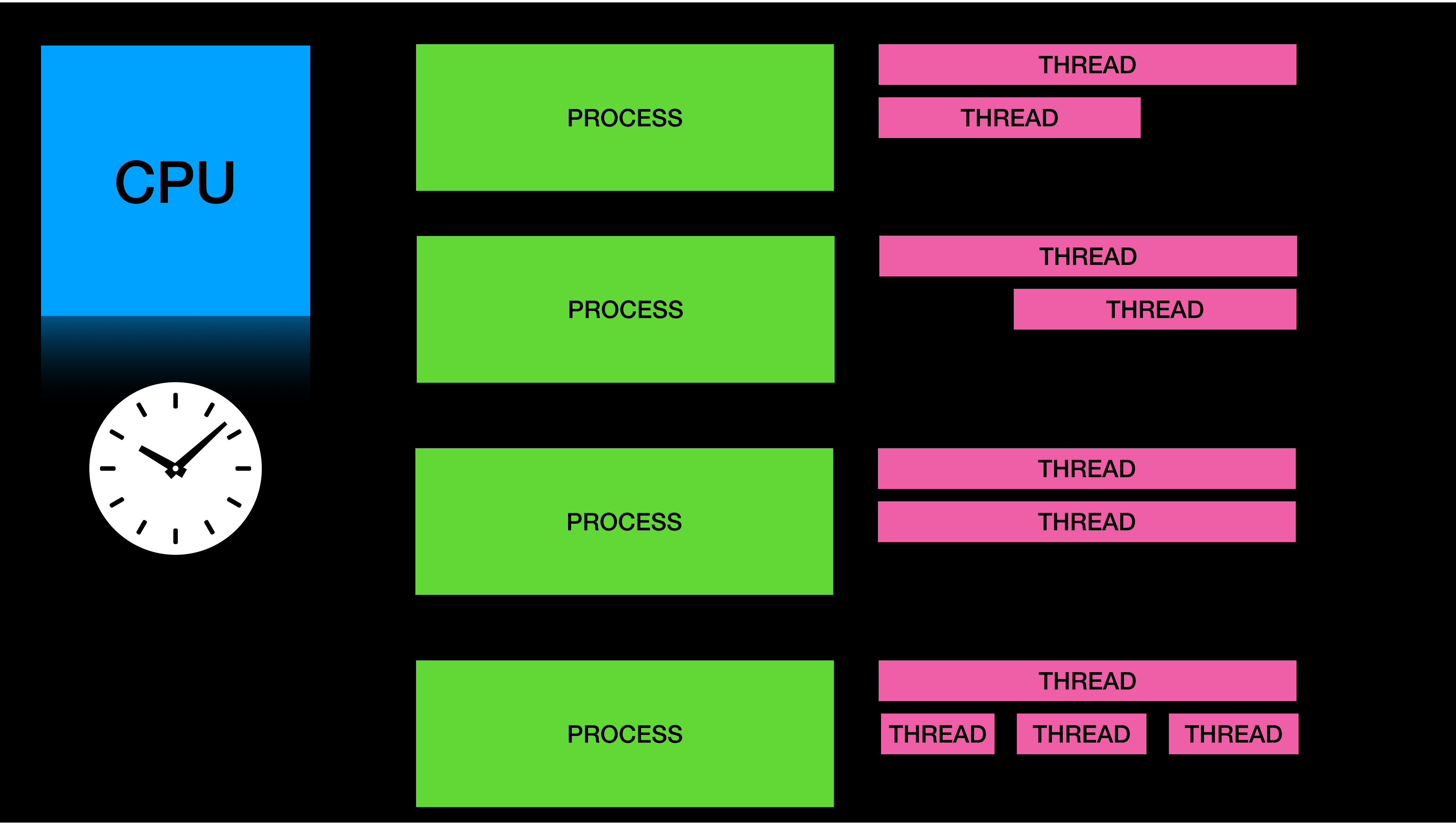
Process Model



Process Model



Threading Model



CPU

PROCESS

THREAD

THREAD

PROCESS

THREAD

THREAD

PROCESS

THREAD

THREAD

PROCESS

THREAD

THREAD

THREAD

THREAD

Threading Model

Every thread has:

ID

Process ID

Private Memory ("thread-local")

Call Stack

Threading Model

Code "runs" on a thread.

By default, all code runs on the main thread.

You can start other threads in order to run other code "concurrently" with the main thread.