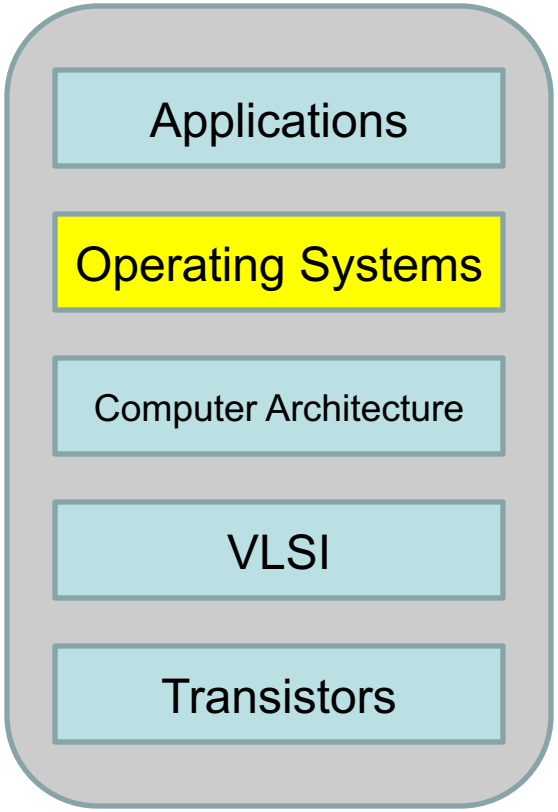
Operating Systems

Slide Courtesy: Dr. Chester Rebeiro, IITM

Layers in a Computer System



Use of Operating Systems

- Hardware Abstraction

  turns hardware into something that applications can use

- Resource Management

  manage system's resources (security)

A Simple Program

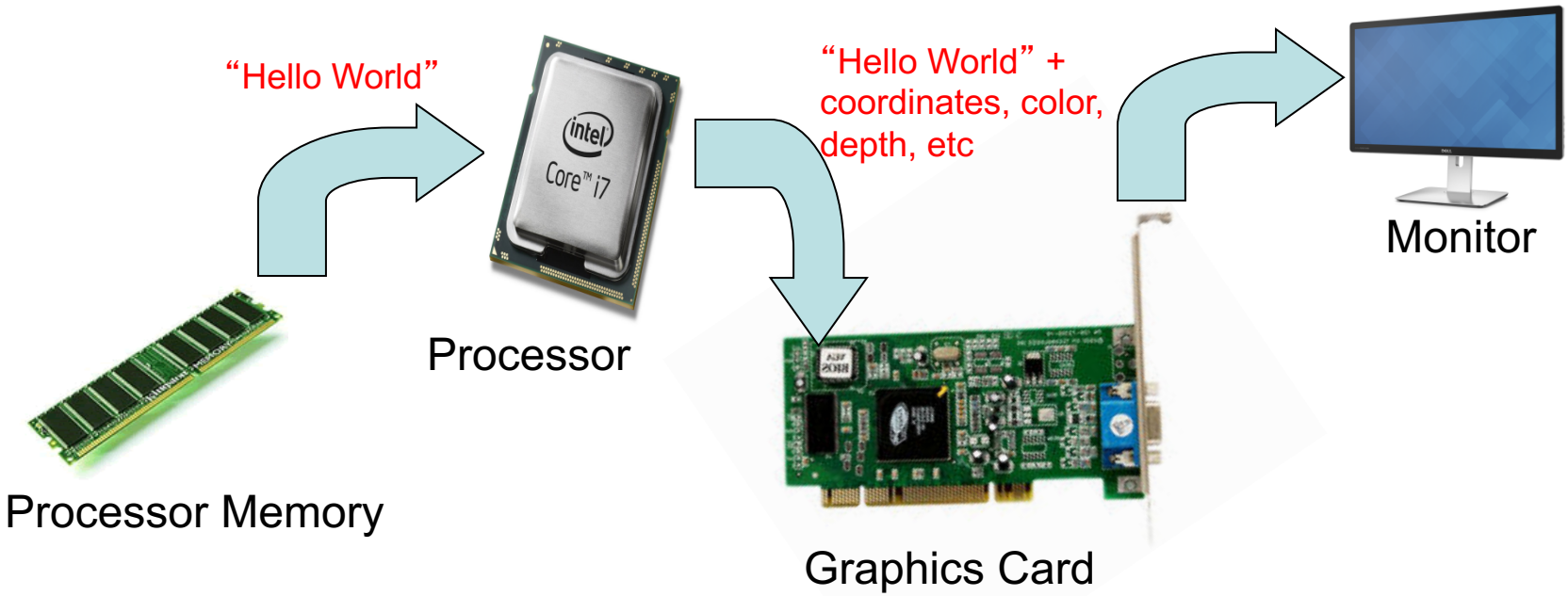
What is the output of the following program?

```c
#include <stdio.h>

int main(){
  char str[] = "Hello World\n";
  printf("%s", str);
}
```

How is the string displayed on the screen?

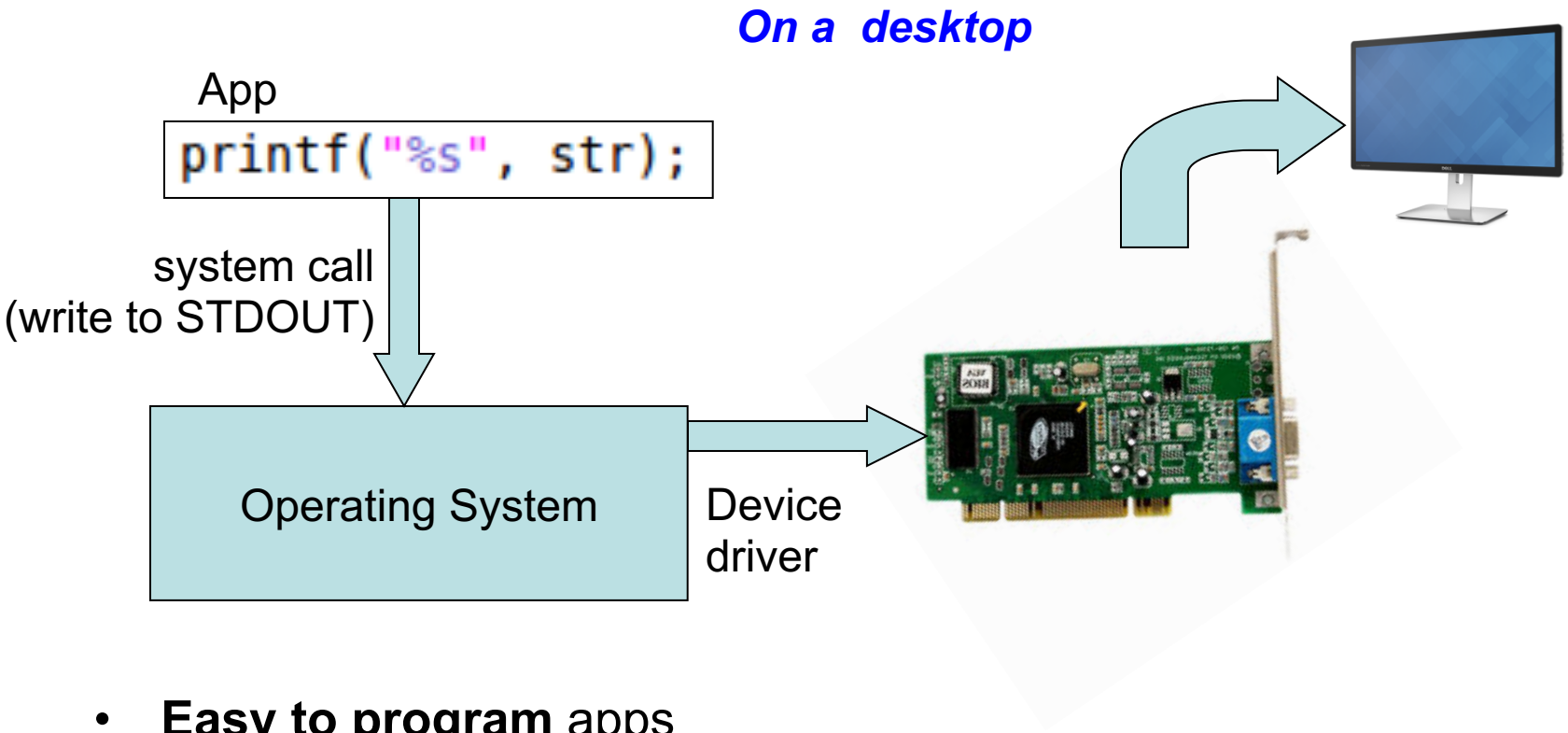
Displaying on the Screen



- Can be complex and tedious
- Hardware dependent

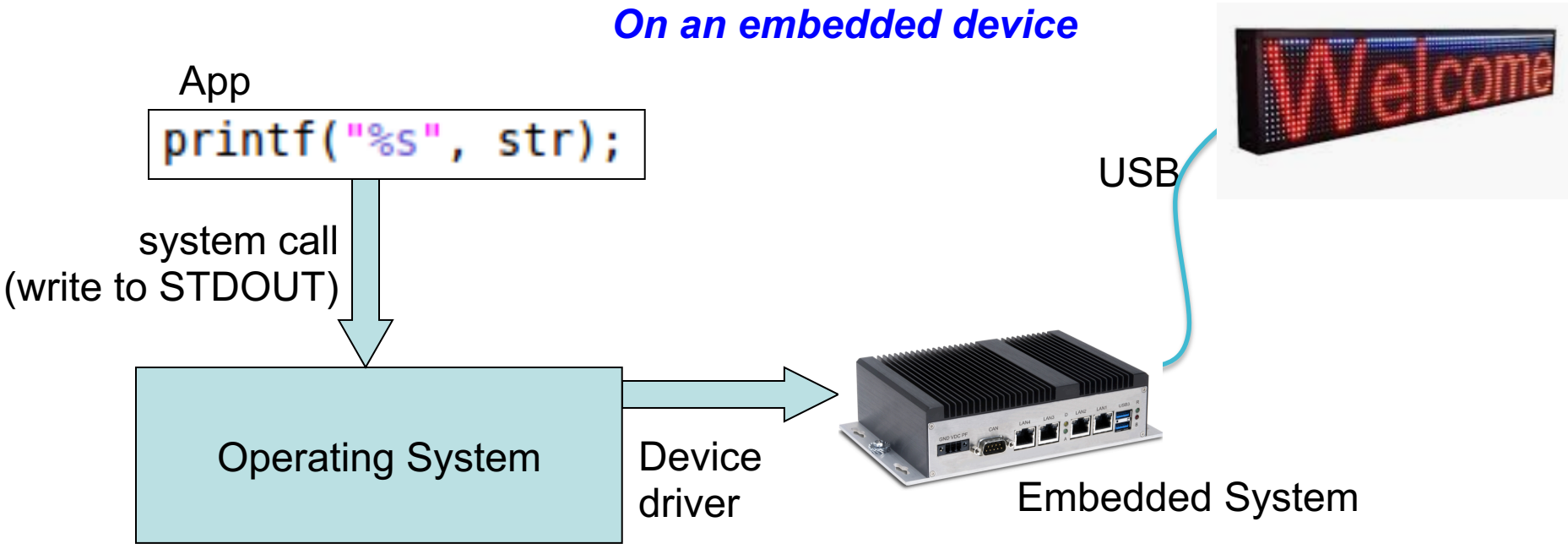
Without an OS, all programs need to take care of every nitty gritty detail.

Operating Systems Provide Abstraction



- **Easy to program** apps
  - No more nitty gritty details for programmers
- **Reusable functionality**
  - Apps can reuse the OS functionality

Operating Systems Provide Abstraction

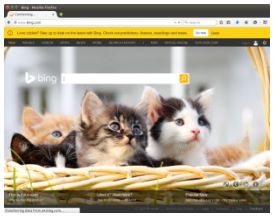


- **Portable**
  - OS interfaces are consistent. The app does not change when hardware changes

OS as a Resource Manager

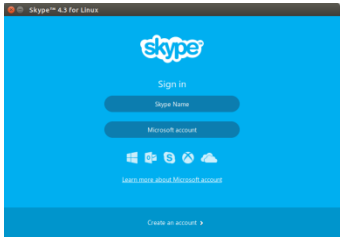
- Multiple apps but limited hardware

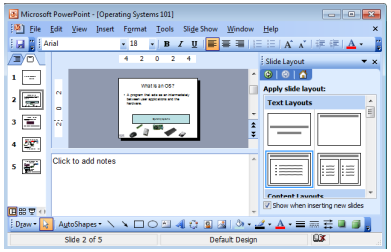
Apps



```
#include <stdio.h>

int main(){
  char str[] = "Hello World\n";
  printf("%s", str);
}
```





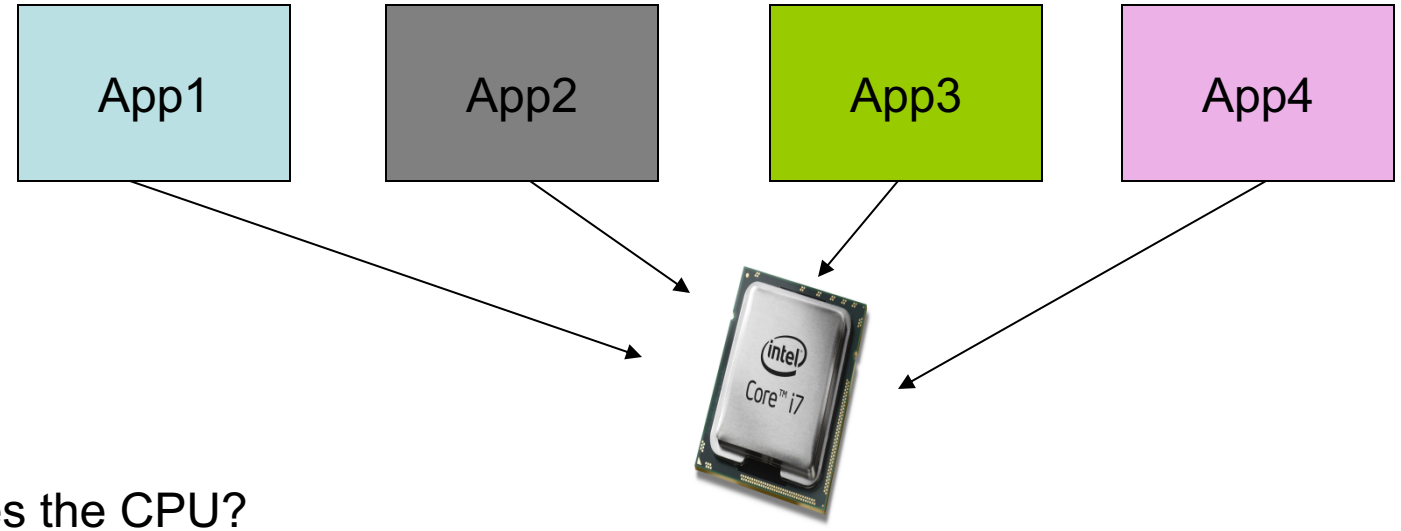
Operating Systems

A few processors

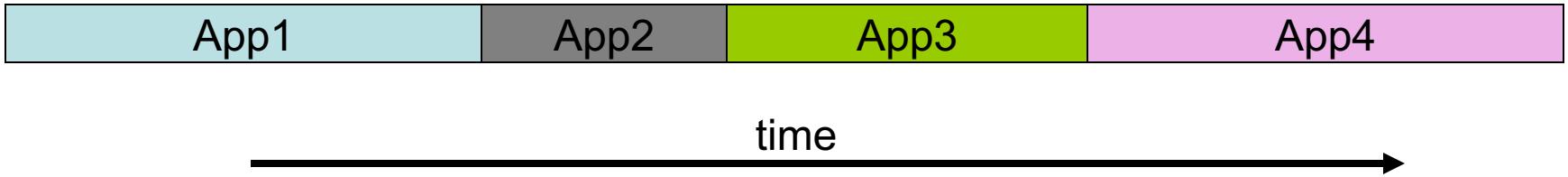
OS as Resource Manager

- OS must manage CPU, memory, network, disk etc…
- Resource management
  - allows multiple apps to share resources
  - protects apps from each other
  - Improves performance by efficient utilization of resources

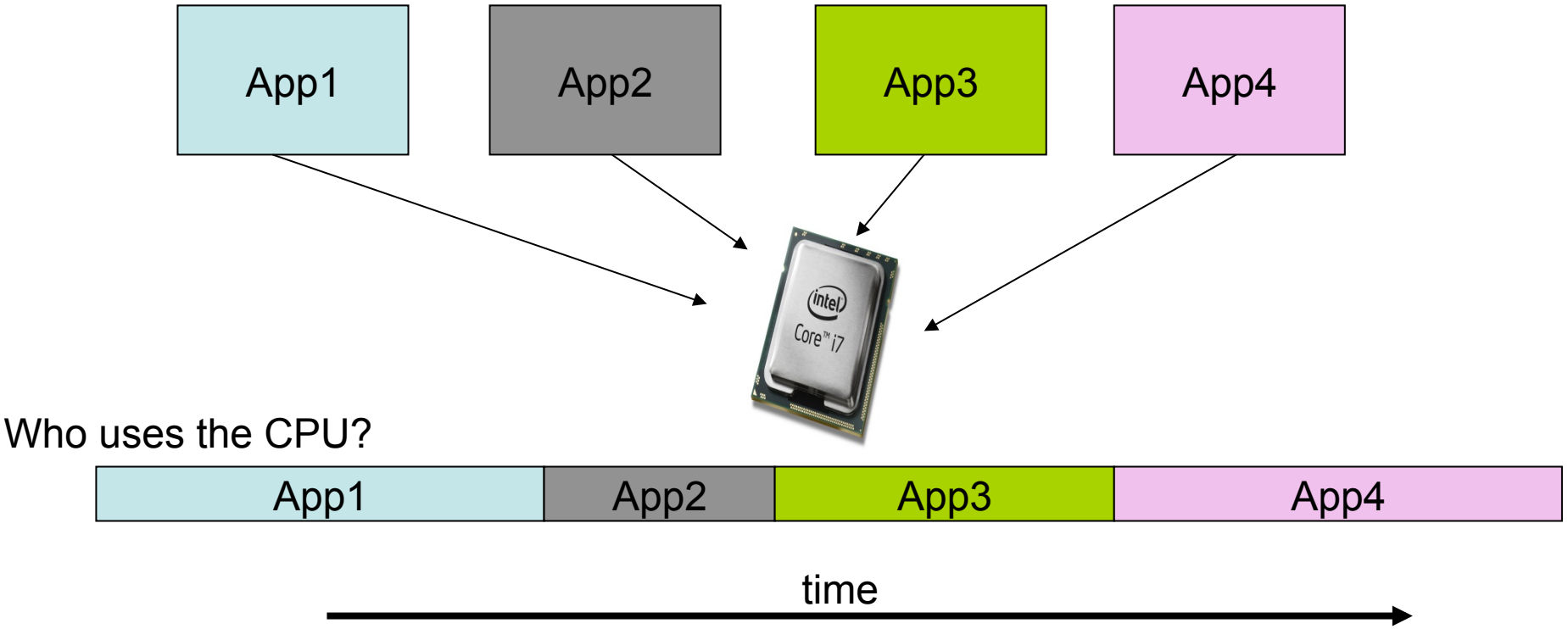
Sharing the CPU



Who uses the CPU?

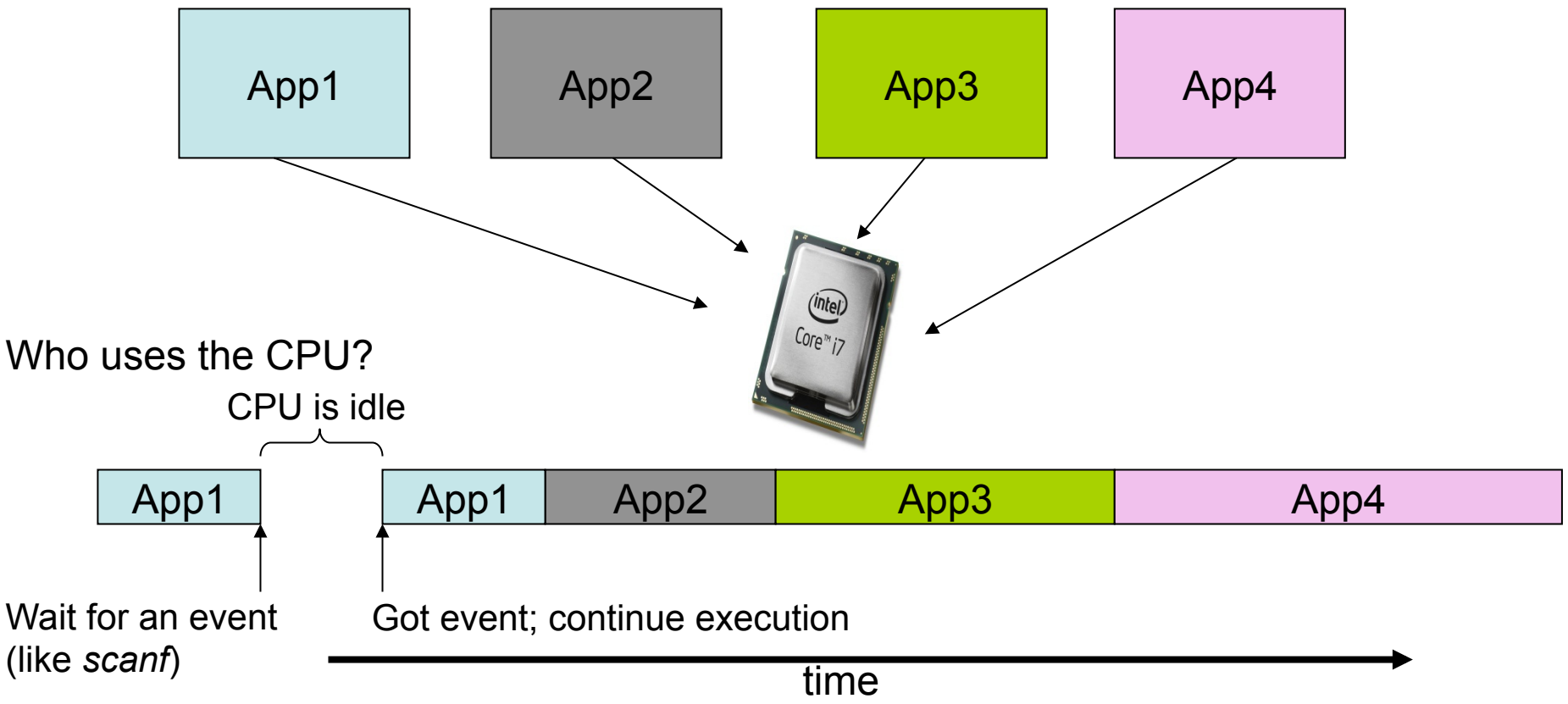


Sharing the CPU



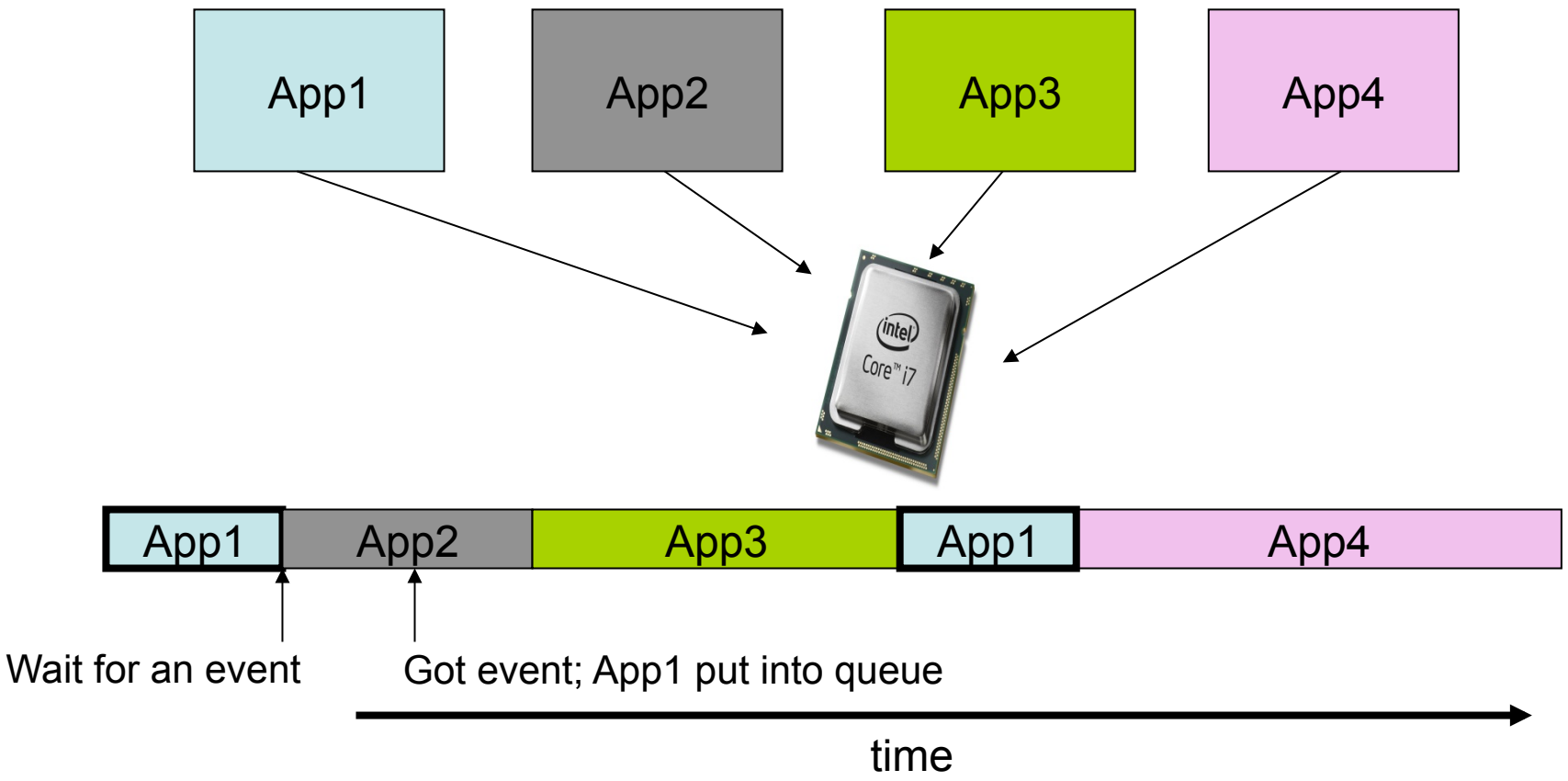
When one app completes the next starts

Idle CPU Cycles



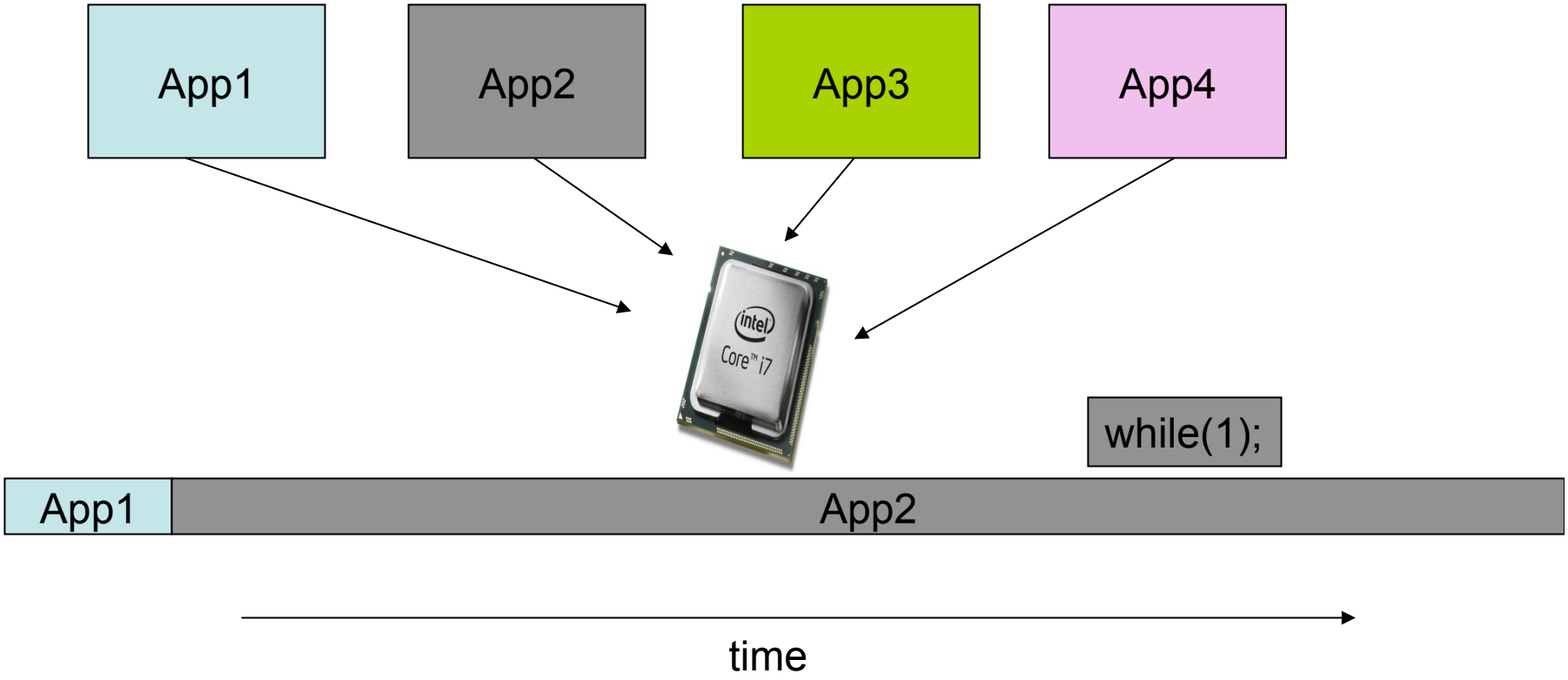
CPU is idle when executing app waits for an event.
Reduced performance.

When OS supports Multiprogramming



When CPU idle, switch to another app

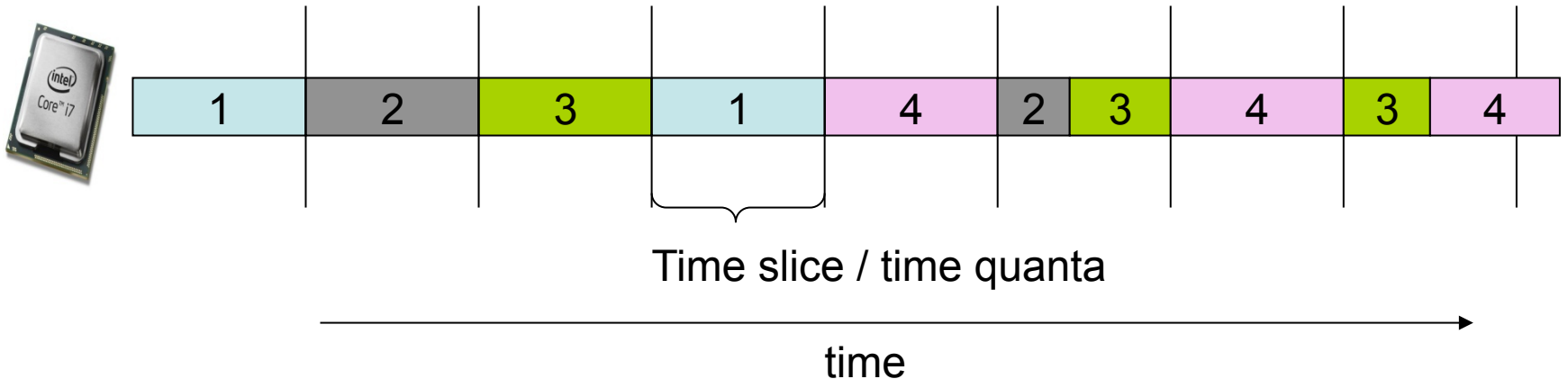
Multiprogramming could cause starvation



One app can hang the entire system

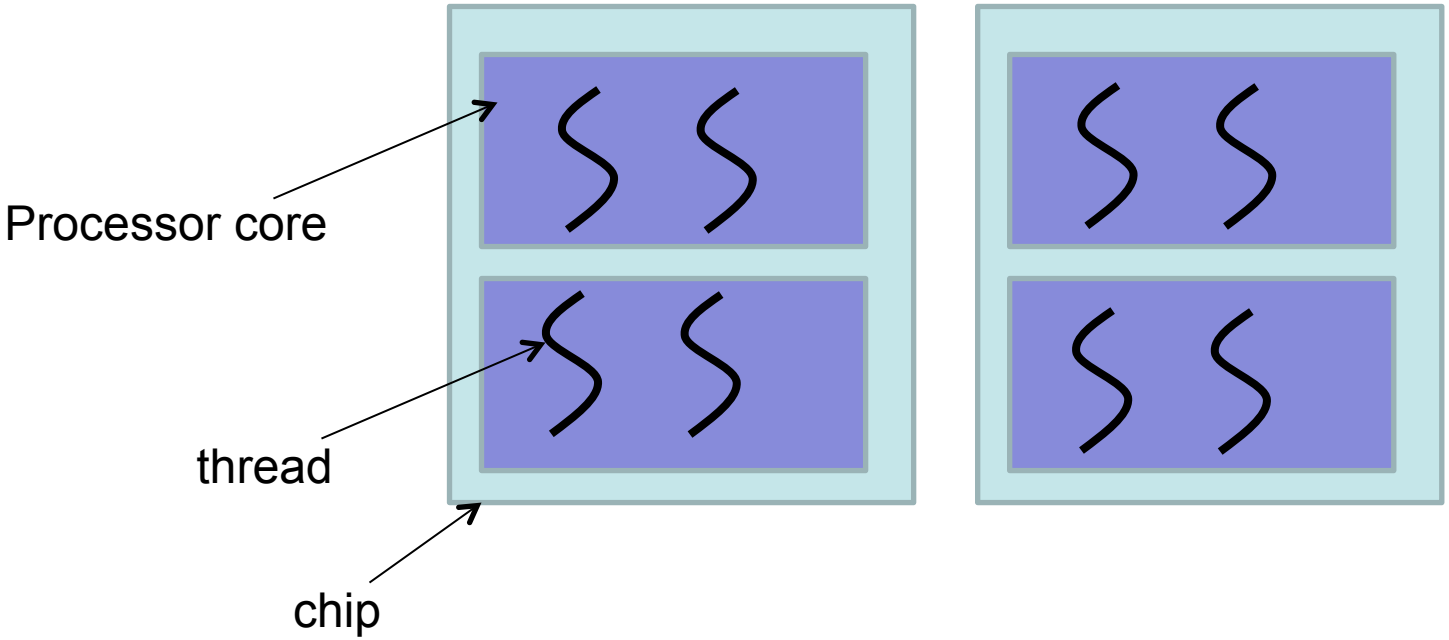
When OS supports Time Sharing (Multitasking)

- Time sliced
- Each app executes within a slice
- Gives impression that apps run concurrently
- No starvation. Performance improved



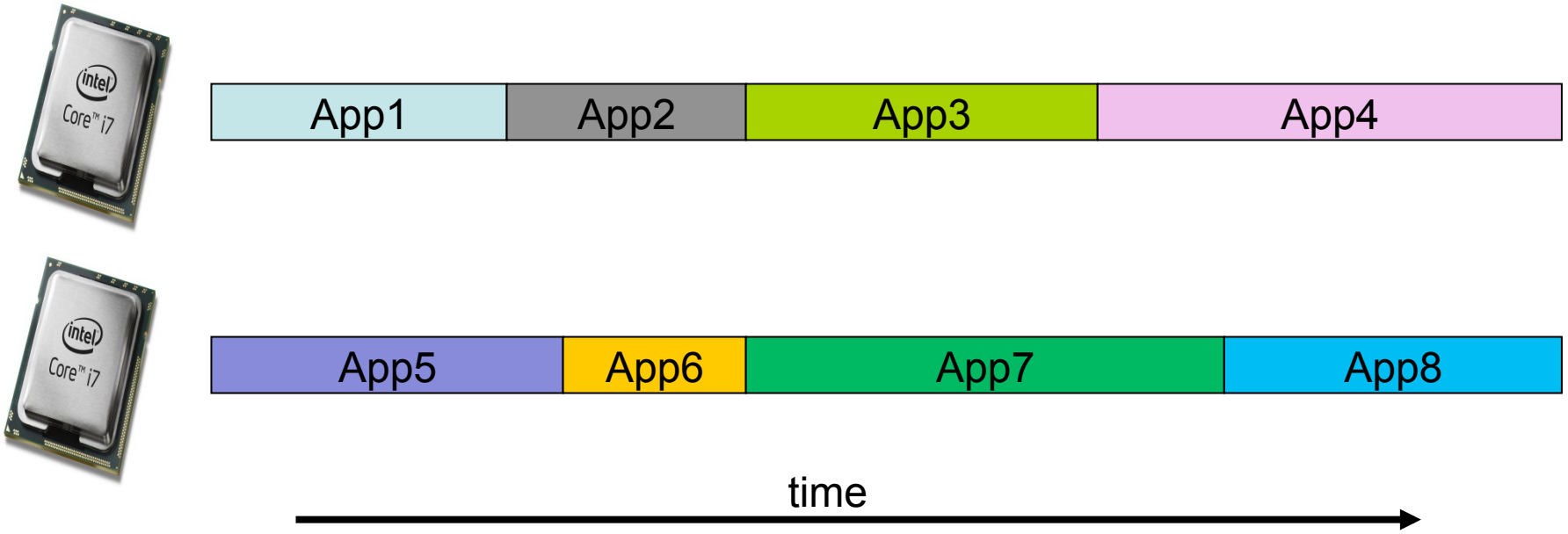
Multiprocessors

- Multiple processors chips in a single system
- Multiple cores in a single chip
- Multiple threads in a single core

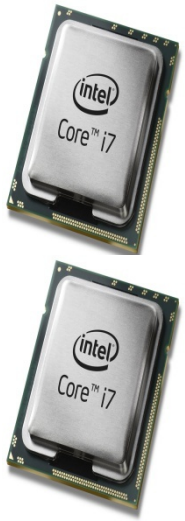


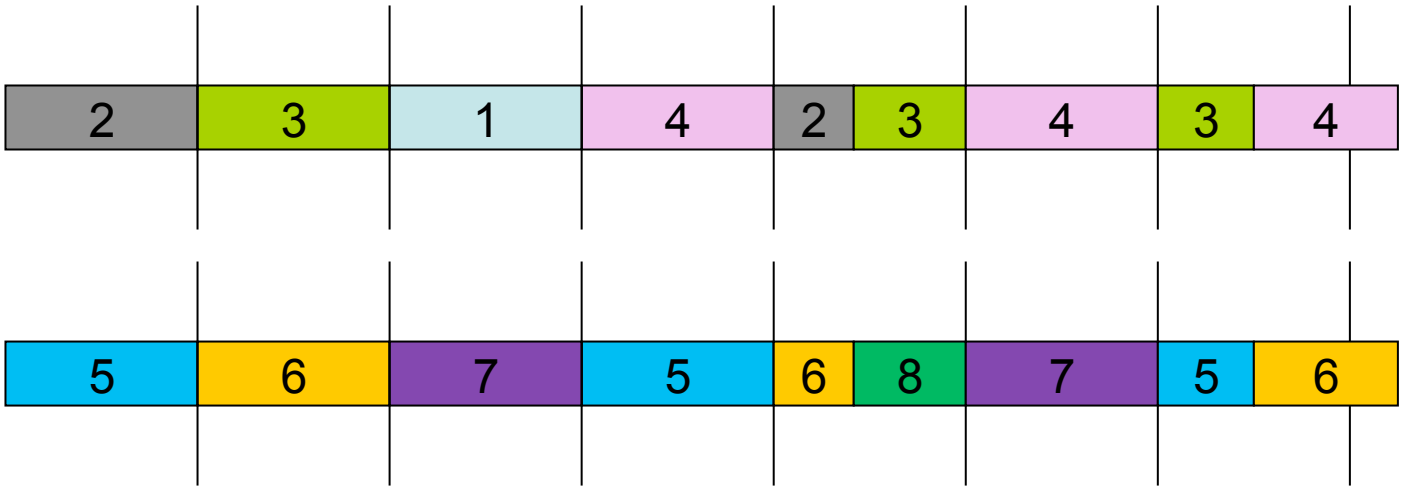
Multiprocessors

- Each processor can execute an app independent of the others

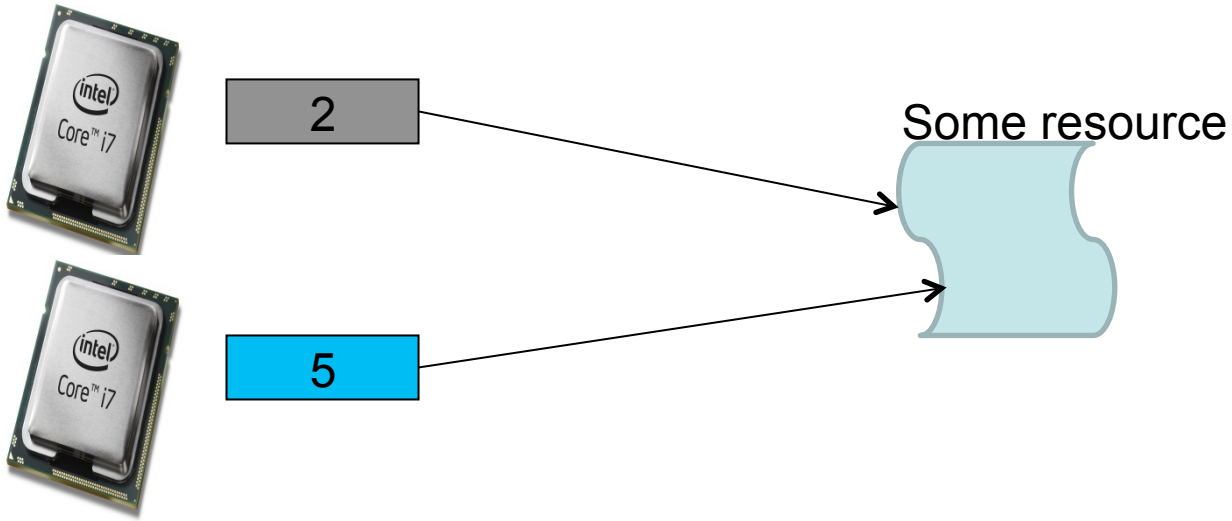


Multiprocessors and Multithreading



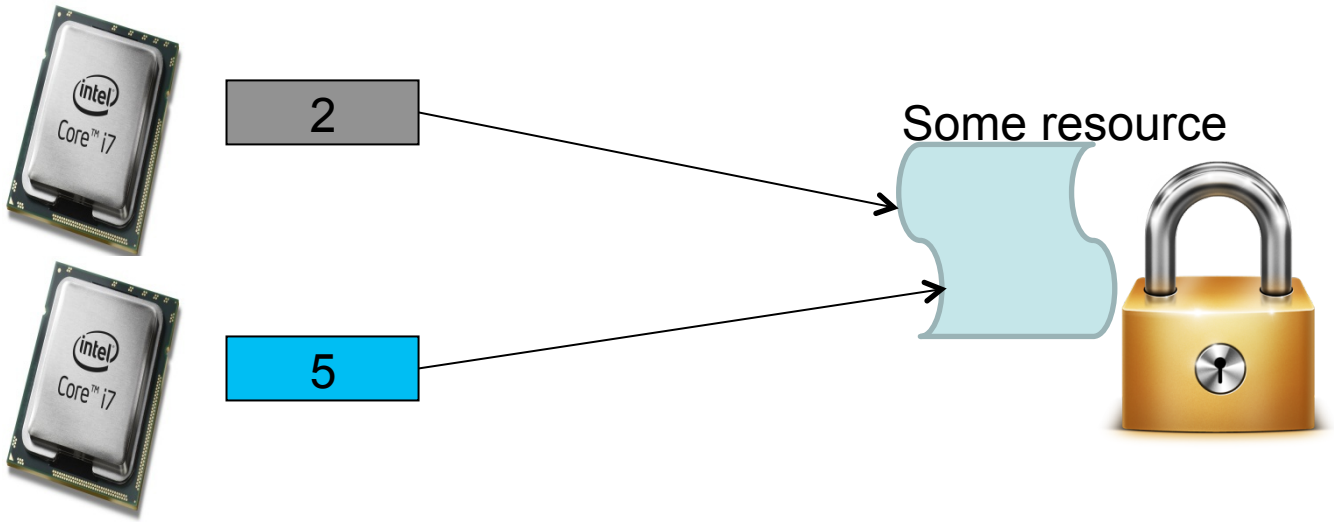


Race Conditions



- App2 and App5 want to write into some resource (like a file) simultaneously
- This results in a race condition
  - Need to synchronize between the two Apps

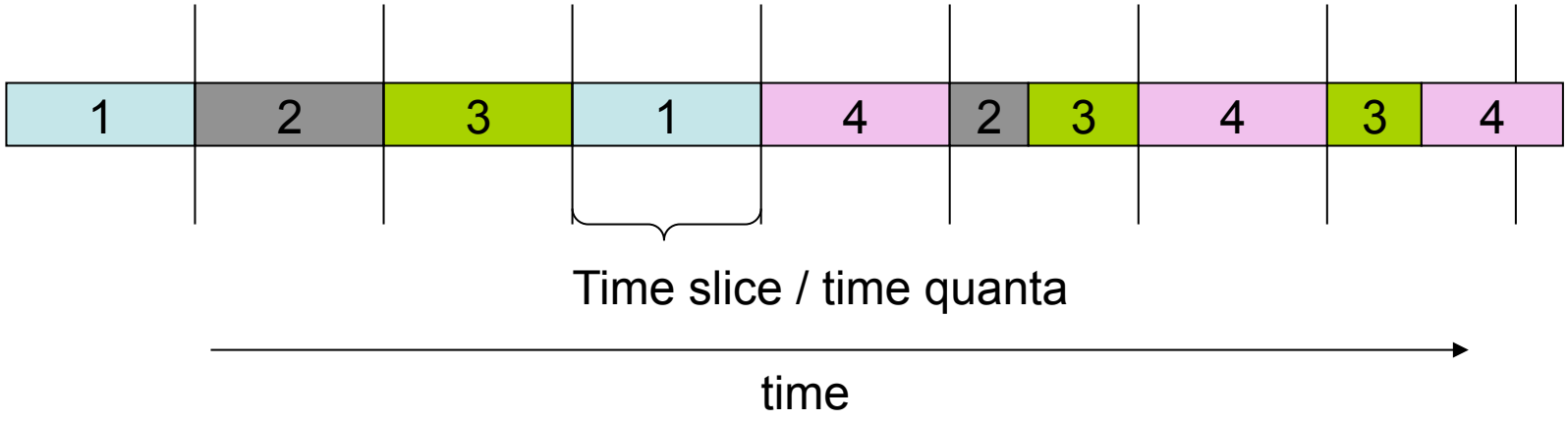
Synchronization



- The shared file is associated with a lock
- The lock ensures that only one App can access the resource at a time
- Sequence of Steps
  - App X locks the resource
  - App X accesses the resource, while App Y waits
  - App X unlocks the resource
  - App Y can now lock and then access the resource

Who should execute next?

- Scheduling
  - Algorithm that executes to determine which App should execute next
  - Needs to be fair
  - Needs to be able to prioritize some Apps over the others



Priority Based Systems

- Based on user choice or a heuristics, some apps are given higher priority compared to others.