COMPSCI: Principles of Operating Systems

Lecture 1: Introduction

Anton Burtsev

January, 2017

Class details

- Undergraduate
  - 22 students
- Instructor: Anton Burtsev
- Meeting time: 9:00-9:50am (M, W, F)
  - Discussions: 1:00-1:50am (F)
    - No regular discussion section unless scheduled
    - Feel free to stop by my office with questions (DBH 3066)
- No TAs
- Web page
  - http://www.ics.uci.edu/~aburtsev/143A

This course

- Inspired by
  - MIT 6.828: Operating System Engineering
    https://pdos.csail.mit.edu/6.828/2016/
  - Adapted for undergraduate students
- We will use xv6
  - Relatively simple (9K lines of code)
  - Reasonably complete UNIX kernel
  - https://pdos.csail.mit.edu/6.828/2016/xv6.html
- xv6 comes with a book
  - https://pdos.csail.mit.edu/6.828/2016/xv6/book-rev9.pdf
- And source code printout
  - https://pdos.csail.mit.edu/6.828/2016/xv6/xv6-rev9.pdf

Course Book

“Operating Systems: Three Easy Pieces”
Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau

- Free online version
  http://pages.cs.wisc.edu/~remzi/OSTEP/

Course organization

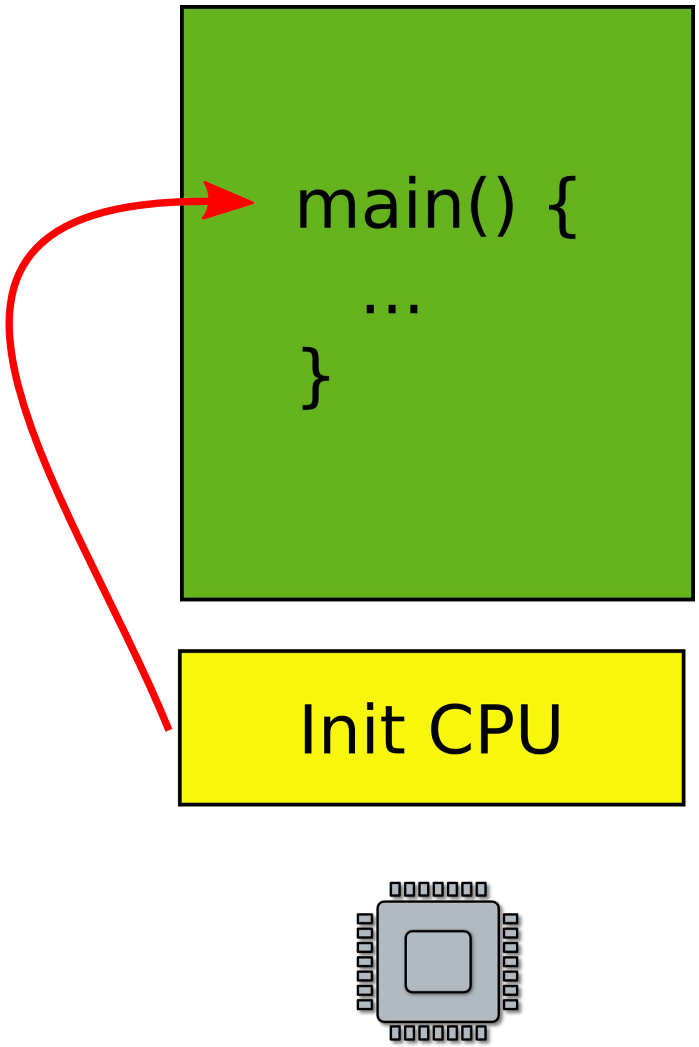
- Lectures
  - High level concepts and abstractions
- Reading
  - Xv6 book + source code
  - OSTEP book
- Homeworks
  - Coding real parts of the xv6 kernel
- Design riddles
  - Understanding design tradeoffs, explaining parts of xv6

Prerequisites

- Solid C coding skills
  - Xv6 is written in C
  - You need to read, code and debug
  - All homeworks are in C
  - Many questions will require explaining xv6 code
- Be able to work and code in Linux/UNIX
- Some assembly skills

What is an operating system?

Goal: Run your code on a piece of hardware



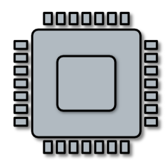
- Read CPU manual
- A tiny boot layer
  - Initialize CPU, memory
  - Jump to your code
- main()
  - **This is your OS!**

Print out a string

- On the screen or serial line

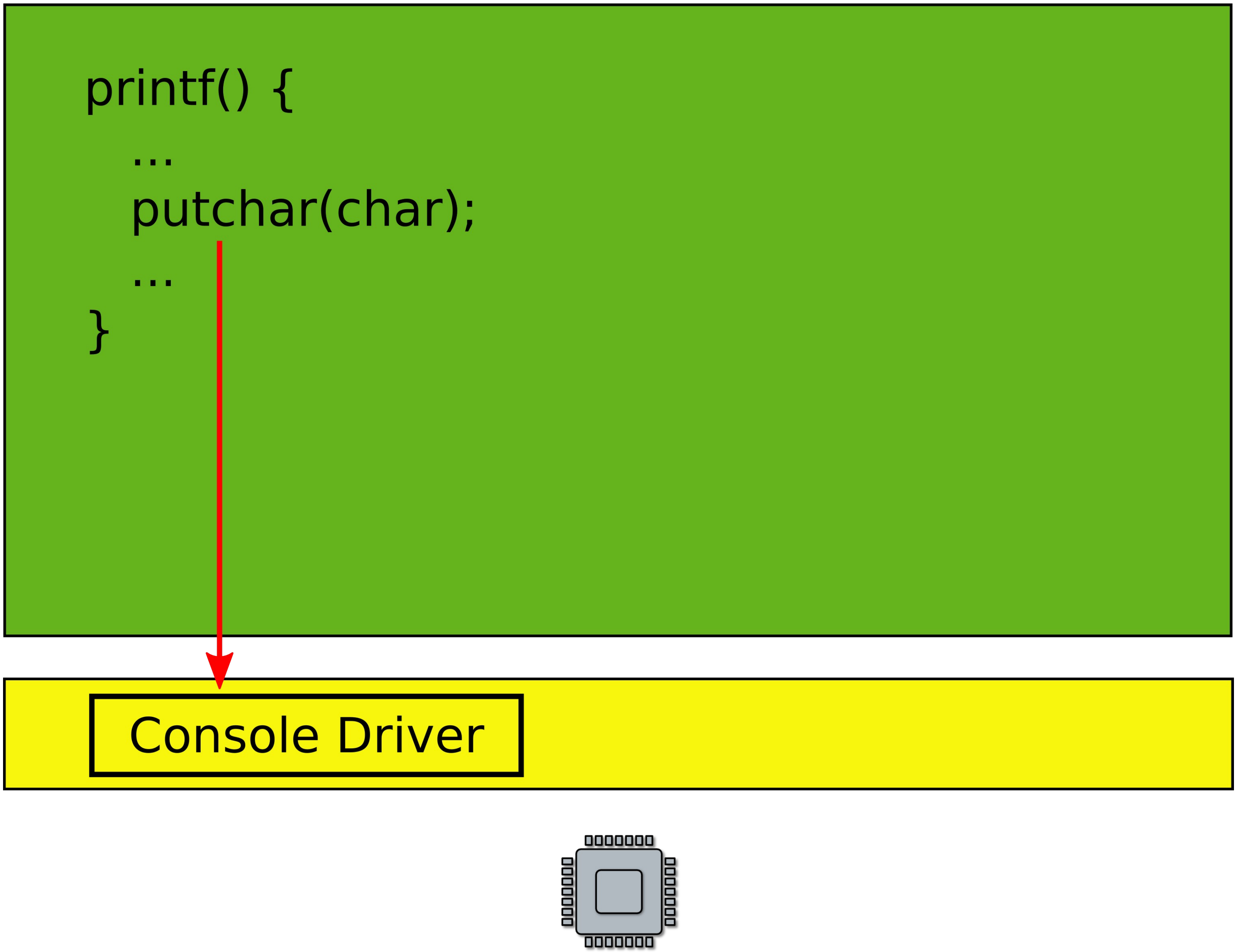
```
printf() {
   ...
   if (vga) {
       asm("mov <magic number 1>, char");
   } else if (serial) {
       asm("out <magic number 2>, char");
   }
   ...
}
```

OS



A more general interface

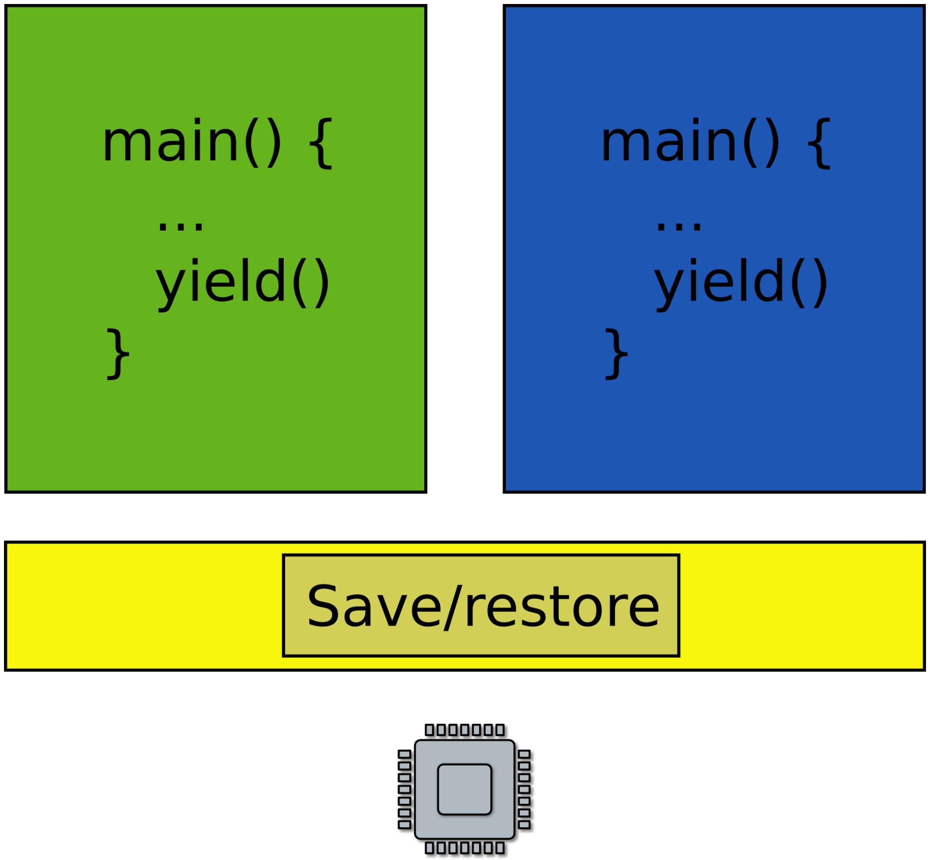
- First device driver



Device drivers

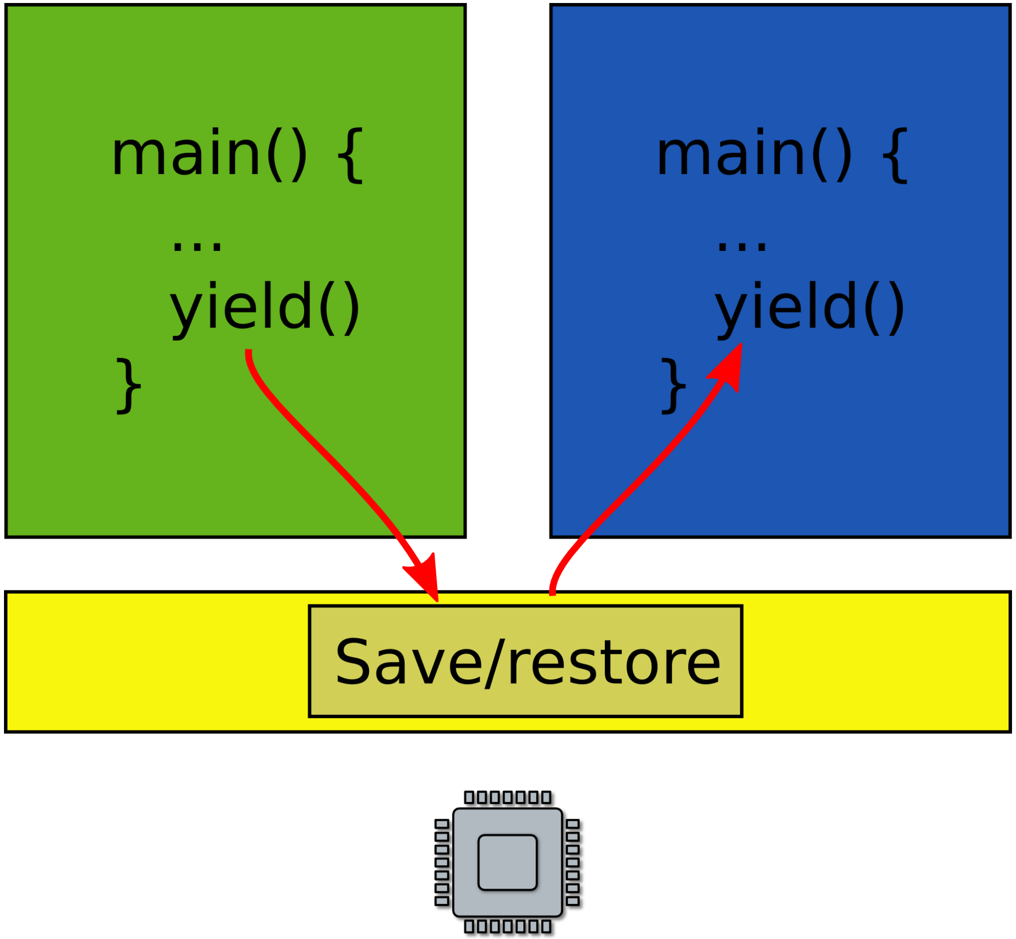
- Abstract hardware
  - Provide high-level interface
  - Hide minor differences
  - Implement some optimizations
    - Batch requests
- Examples
  - Console, disk, network interface
  - ...virtually any piece of hardware you know

Goal: Want to run two programs



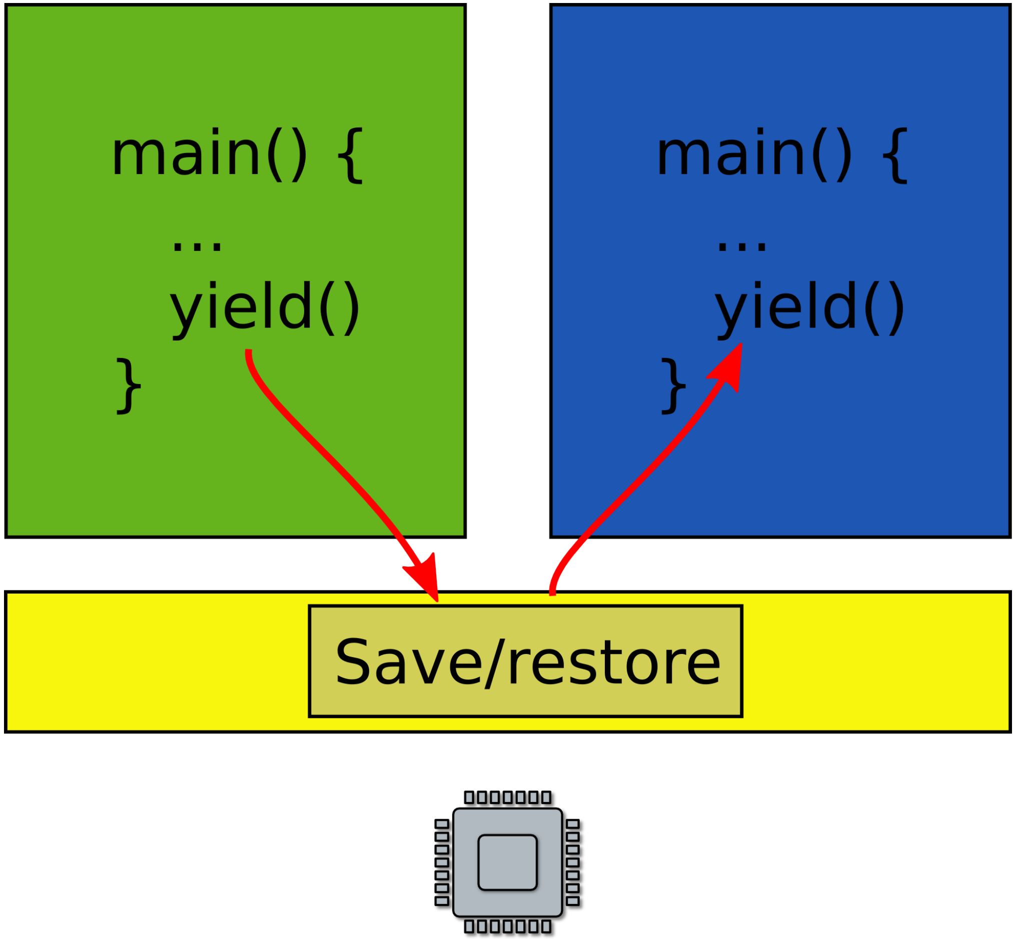
- What does it mean?
  - Only one CPU
- Run one, then run another one

Goal: Want to run two programs



- Exit into the kernel periodically
- Context switch
  - Save and restore context
  - Essentially registers

- What! Two programs, one memory?

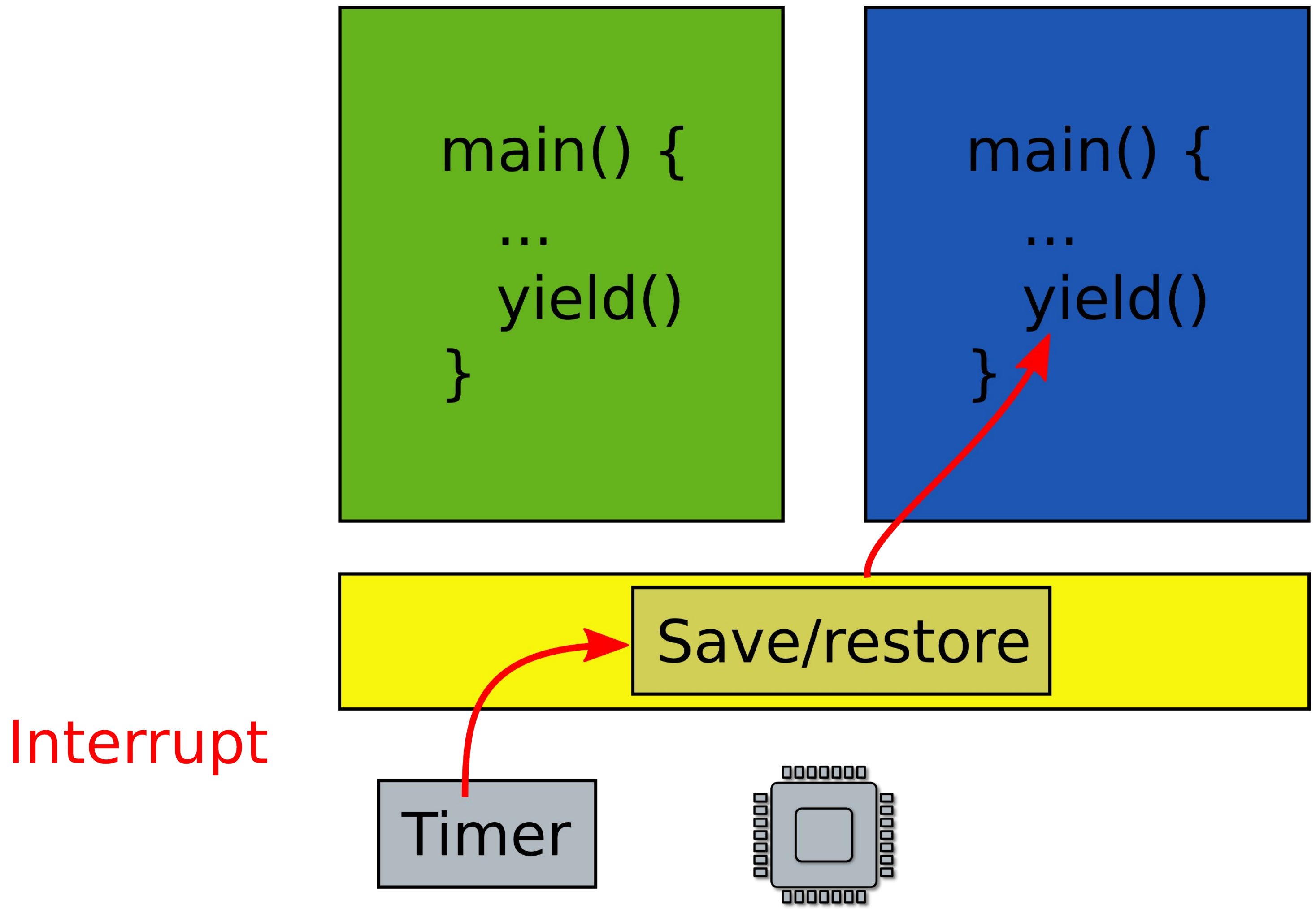


- Linker magic?

Virtual address spaces

- Illusion of a private memory for each application
  - Keep a description of an address space
  - In one of the registers
- All normal program addresses are inside the address space
- OS maintains description of address spaces
  - Switches between them

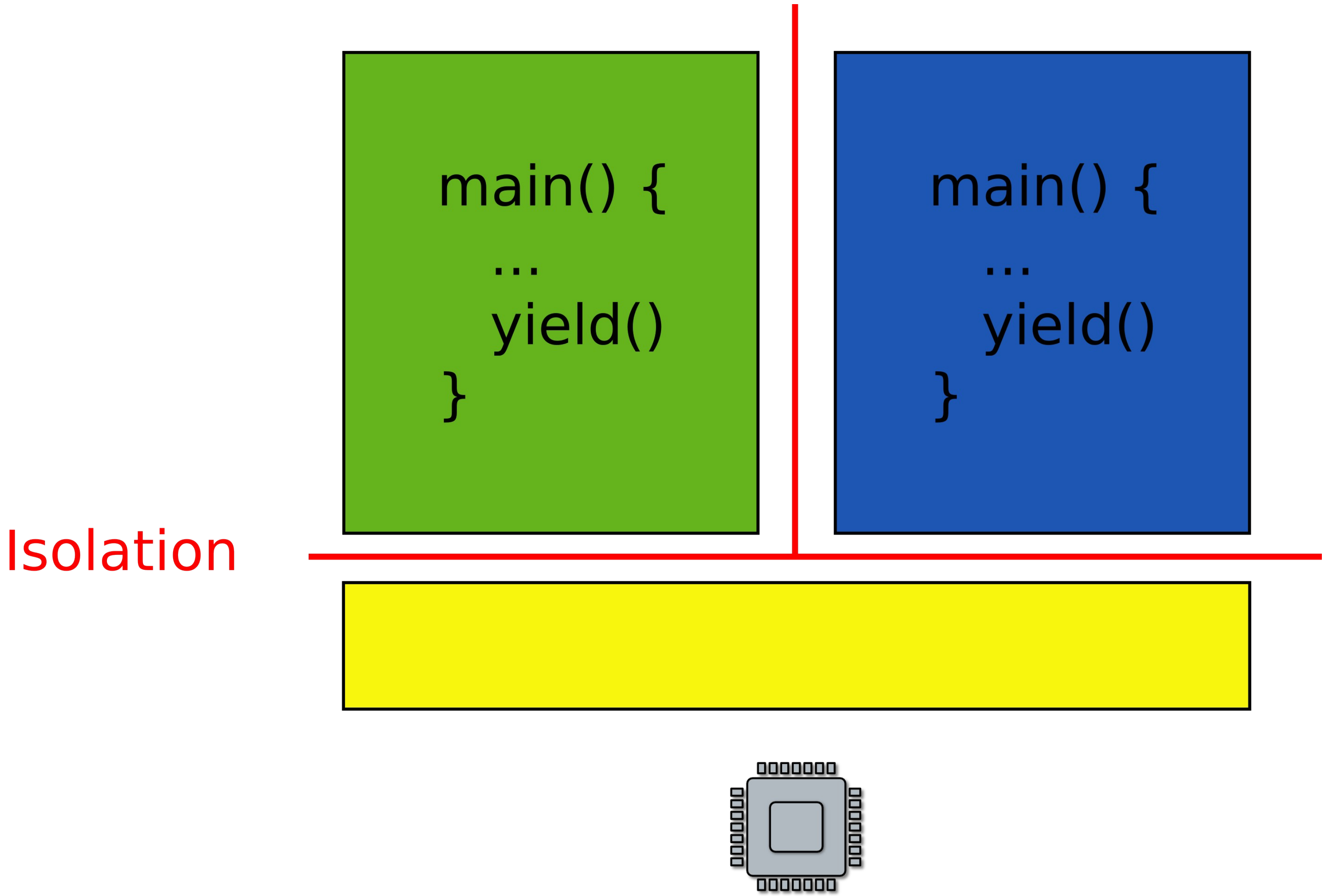
- What if one program fails to release the CPU?
- It will run forever. Need a way to preempt it. How?



Scheduling

- Pick which application to run next
  - And for how long
- Illusion of a private CPU for each task
  - Frequent context switching

- What if one faulty program corrupts the kernel?
- Or other programs?

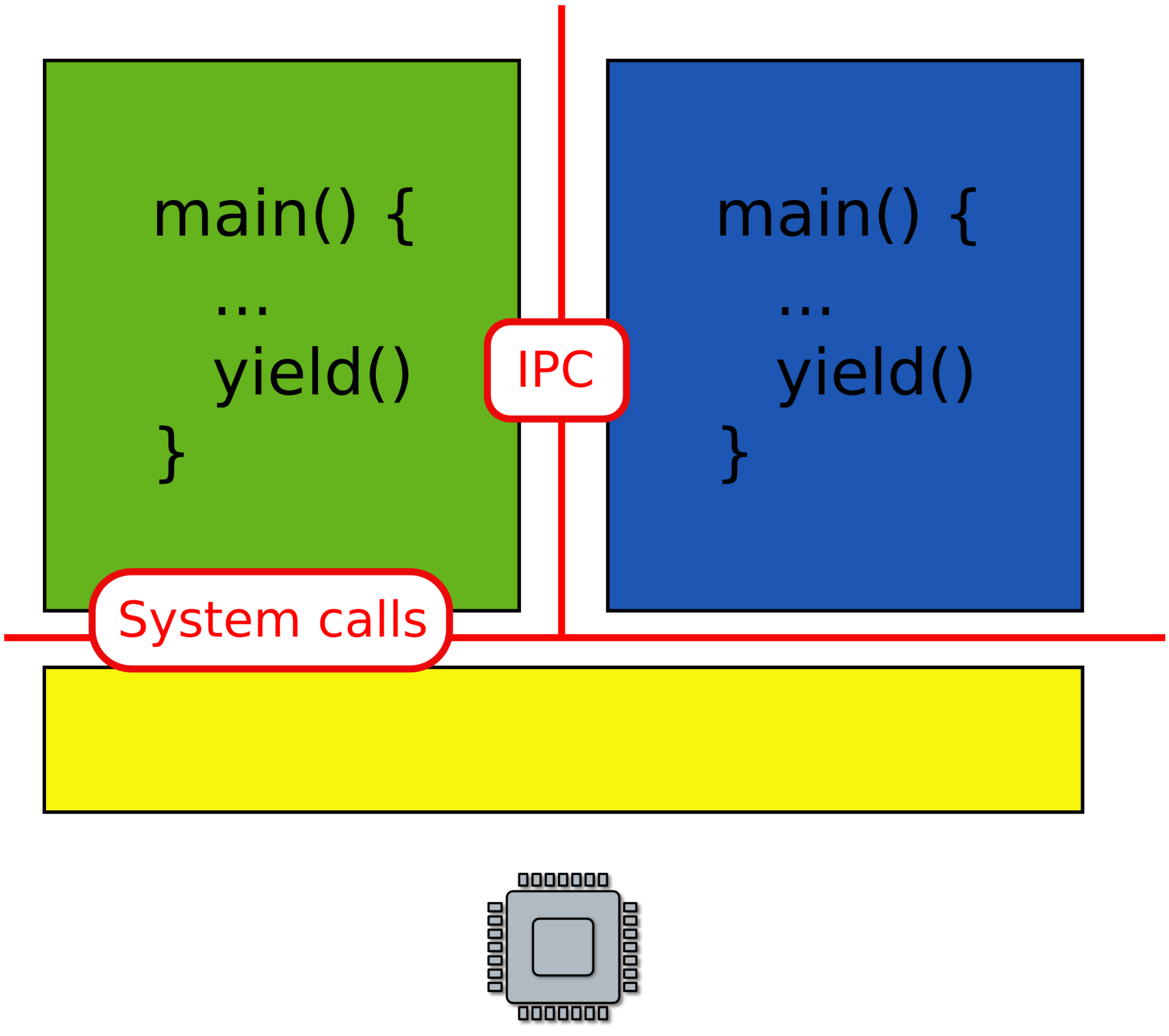


Isolation

- Today is done with address spaces in hardware
  - Many issues, e.g. shared device drivers, files, etc.

- Can it be done in software?

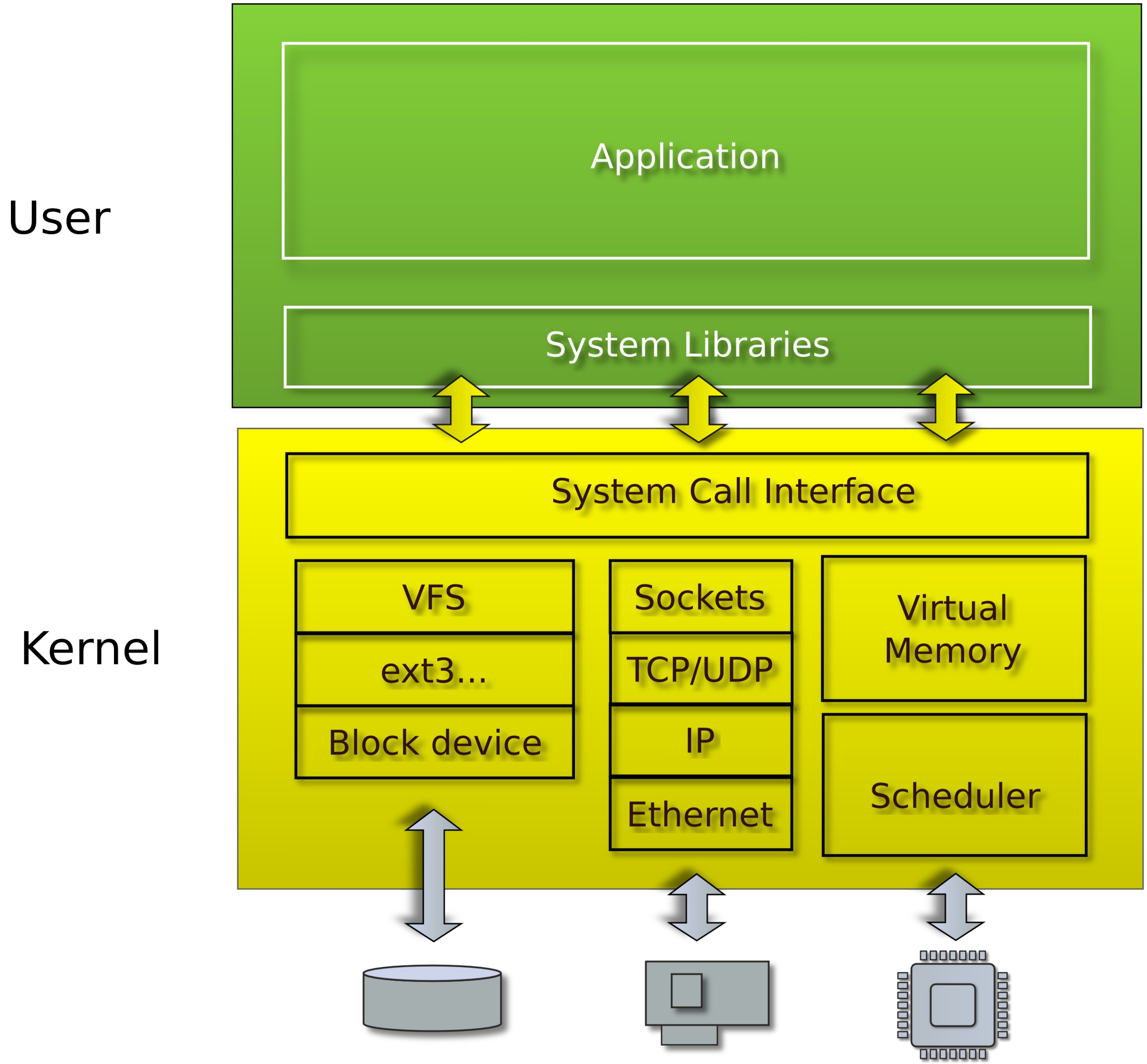
- What about communication?
- Can we invoke a function in a kernel?



- What if you want to save some data?
- Permanent storage
  - E.g., disks
- But disks are just arrays of blocks
  - wrtie(block_number, block_data)
- Files
  - High level abstraction for saving data
  - fd = open("contacts.txt");
  - fpritnf(fd, "Name:%s\n", name);

- What if you want to send data over the network?
- Network interfaces
  - Send/receive Ethernet packets (Level 2)
  - Two low level
- Sockets
  - High level abstraction for sending data

- Linux/Windows/Mac

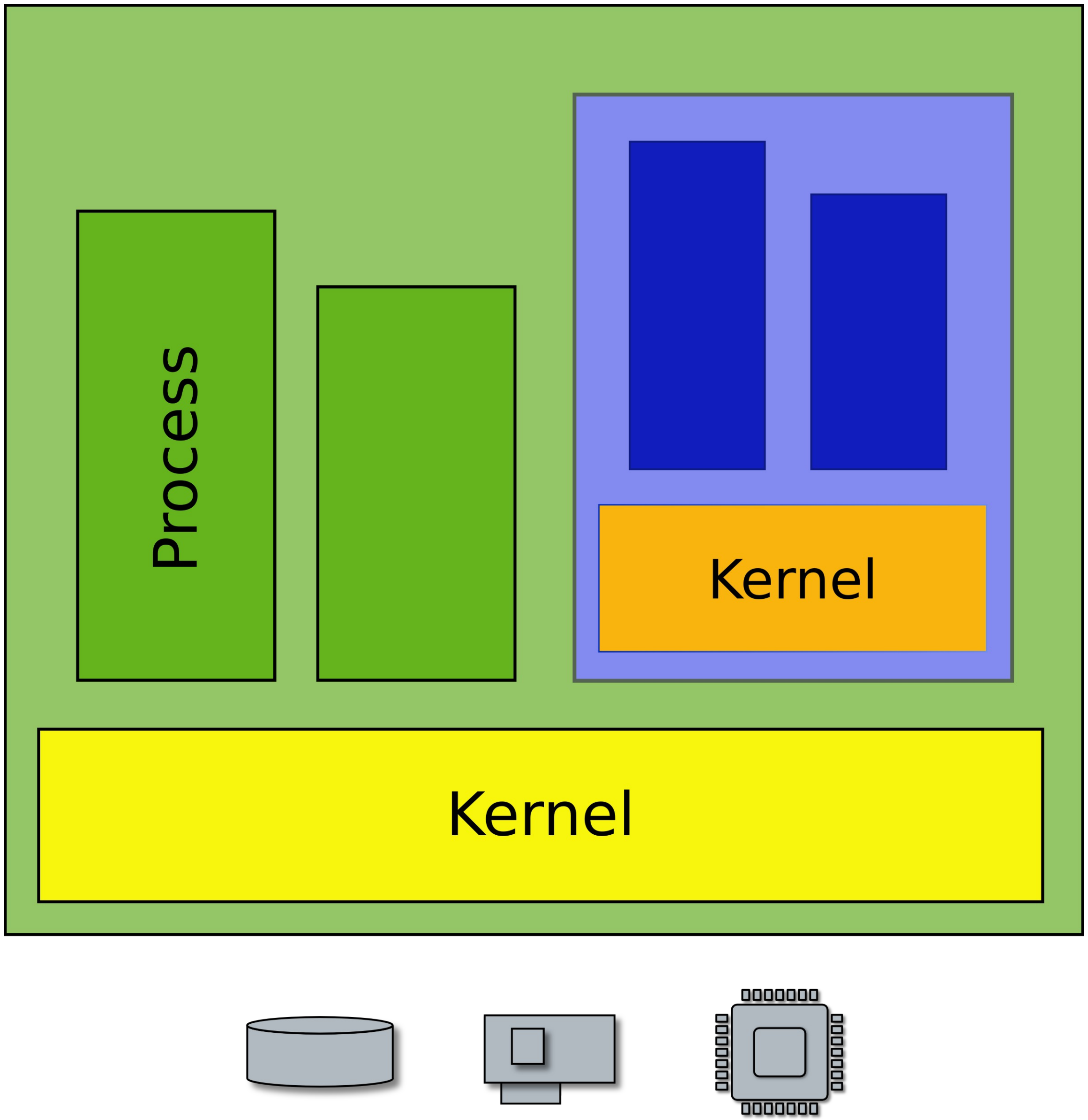


Multiple levels of abstraction

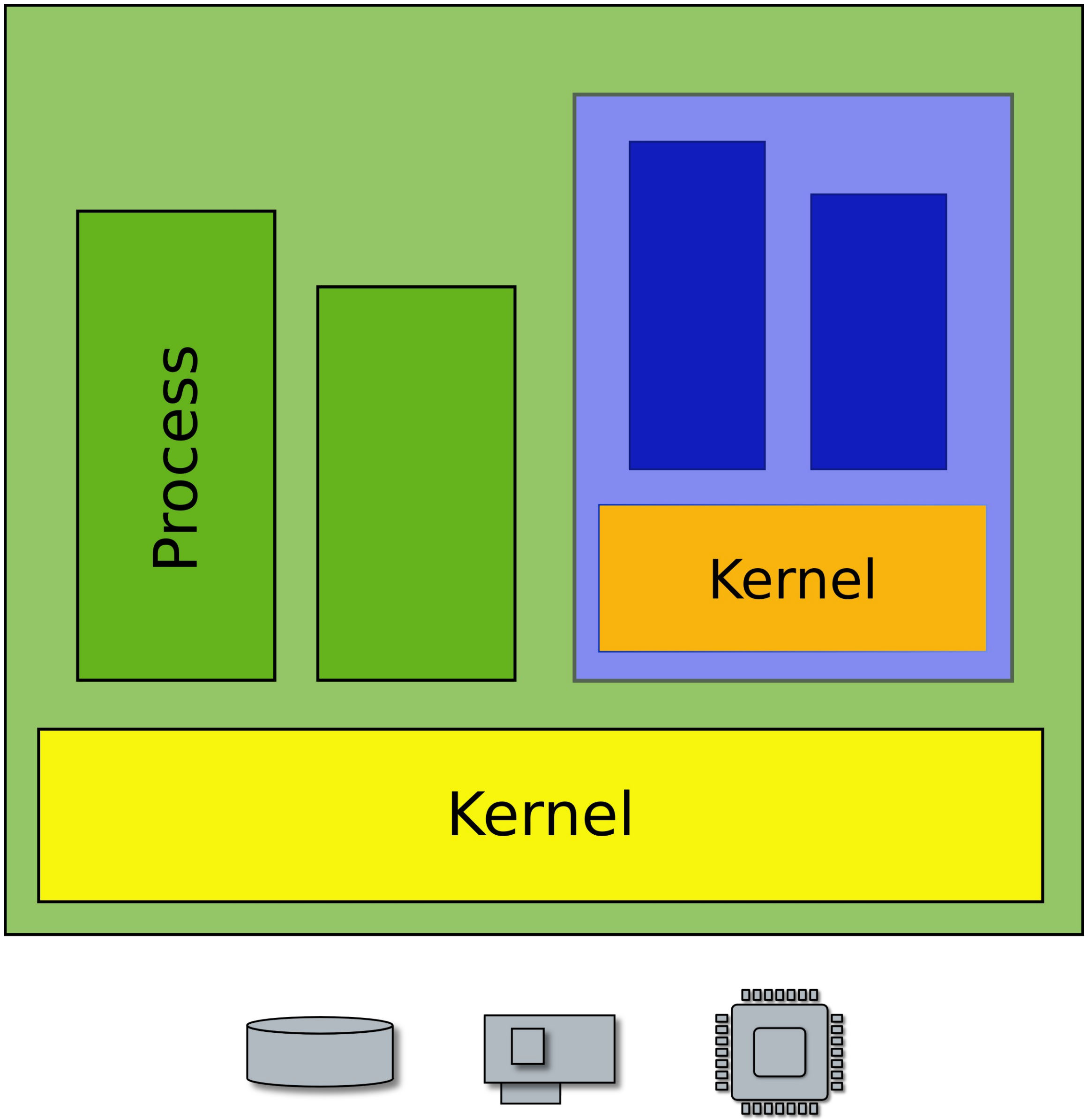
- Multiple programs
  - Each has illusion of a private memory and CPU
  - Context switching, scheduling, isolation, communication
- File systems
  - Multiple files, concurrent I/O requests
  - Consistency, caching
- Network protocols
  - Multiple virtual network connections
- Memory management

Virtualization

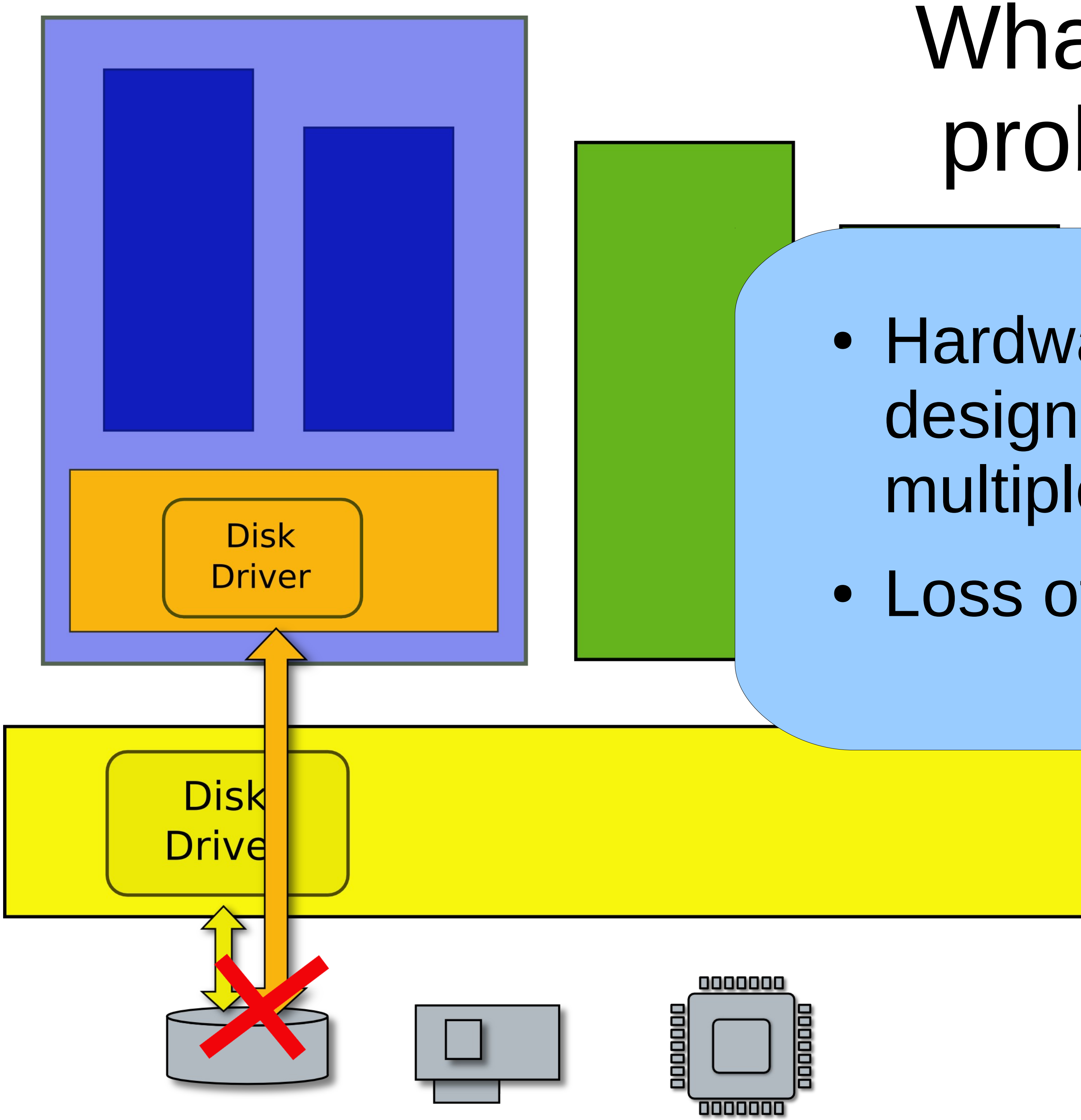
- Want to run a Windows application on Linux?



- Want to run a Windows application on Linux?

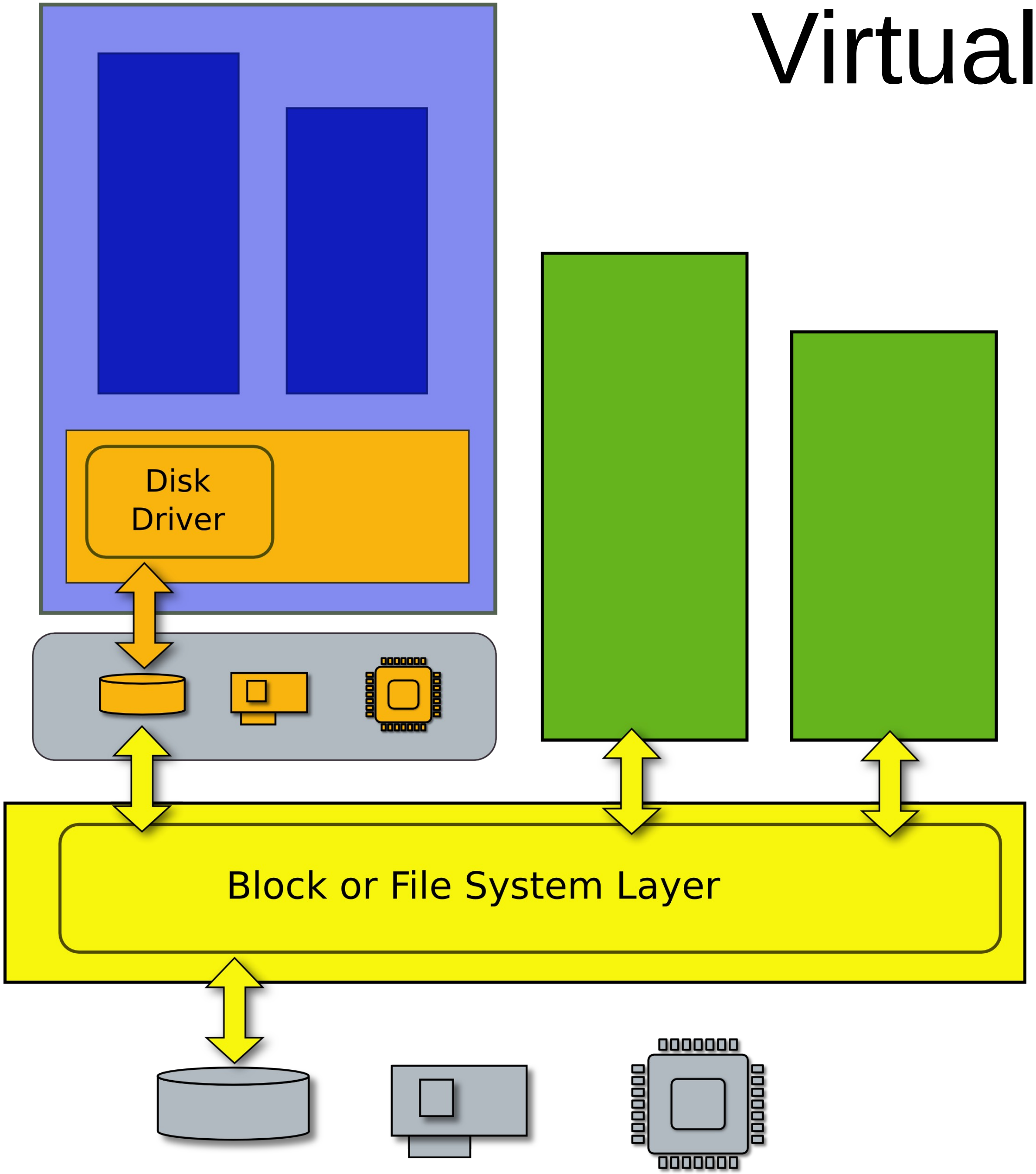


What is the problem?



- Hardware is not designed to be multiplexed
- Loss of isolation

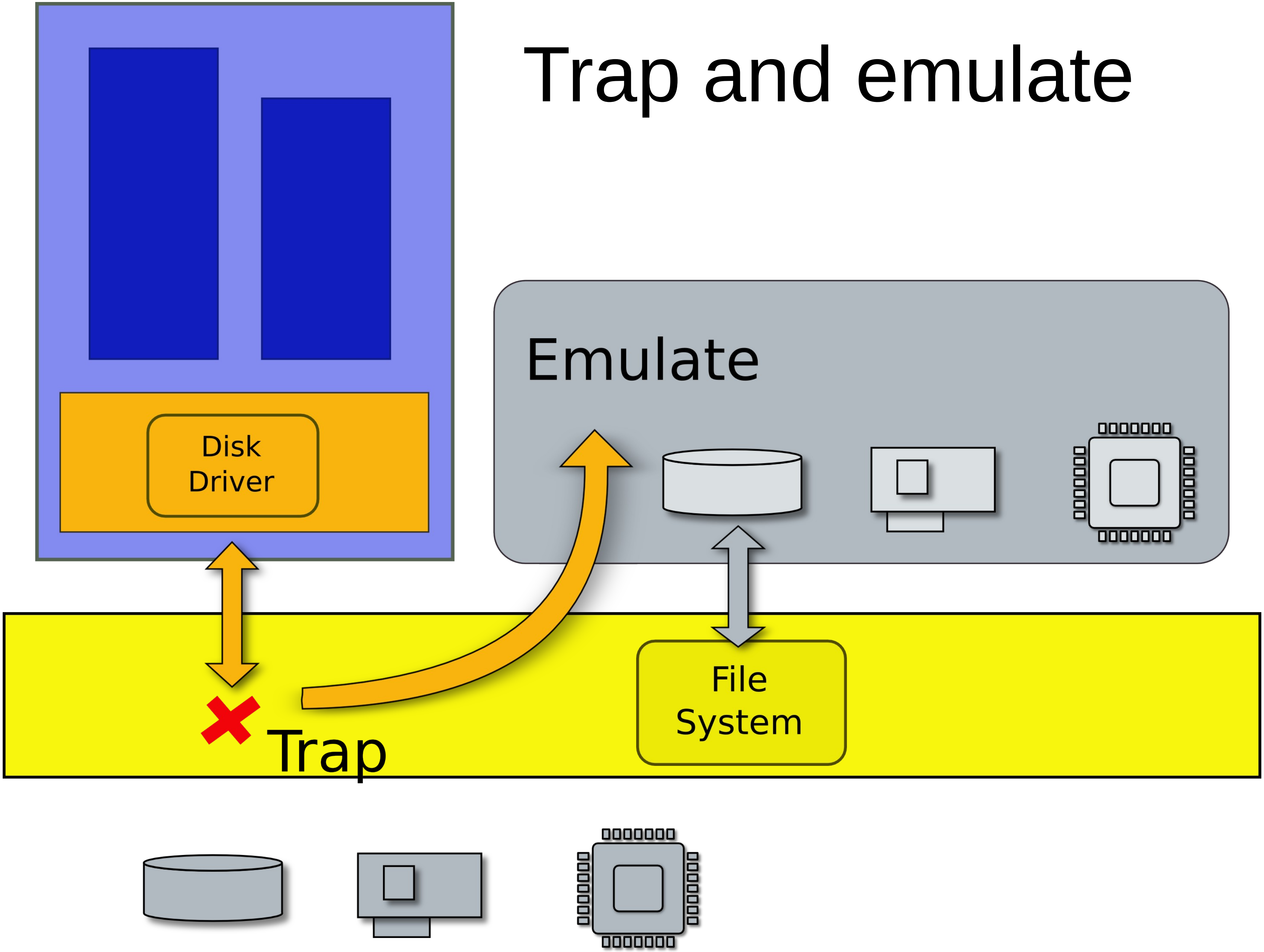
Virtual machine



Efficient duplicate of a real machine

- Compatibility
- Performance
- Isolation

Trap and emulate



Questions?