Introduction to Communicating Distributed Processes
Topic 2 – The kernel abstraction

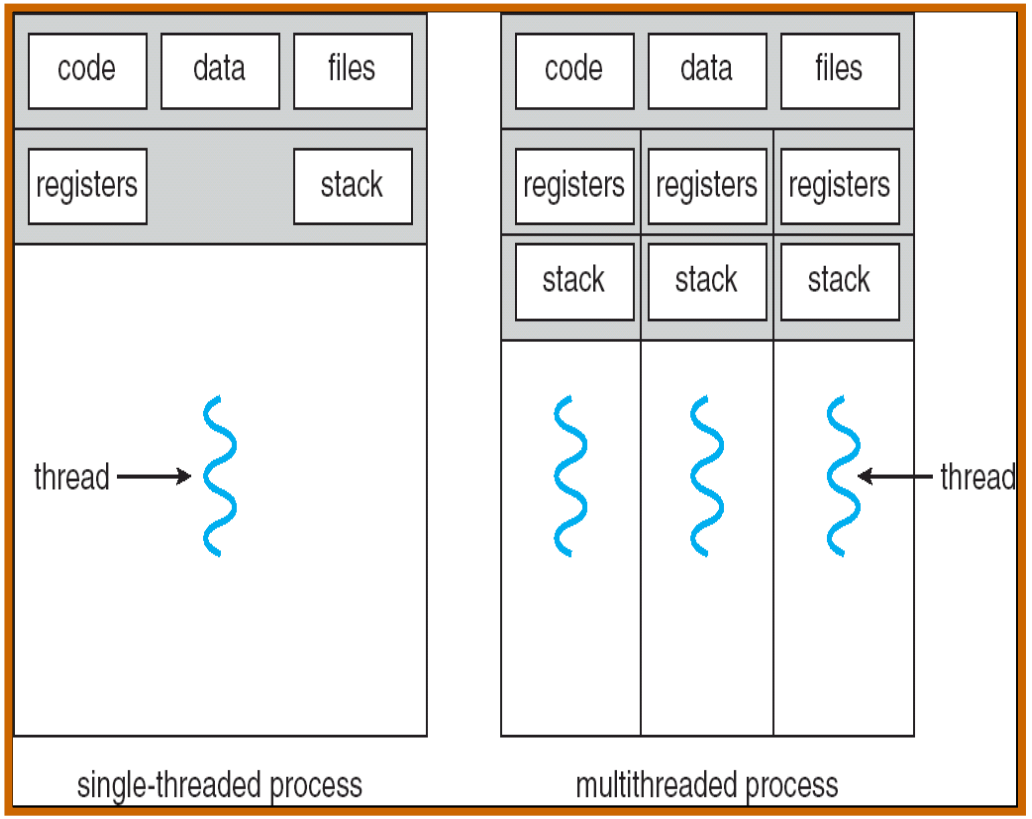
Third OS Concept: Process

- **Process: execution environment with Restricted Rights**
  - **Address Space with One or More Threads**
  - **Owns memory (address space)**
  - **Owns file descriptors, file system context, …**
  - **e.g. web-server, mail server**

- **Processes provides memory protection**

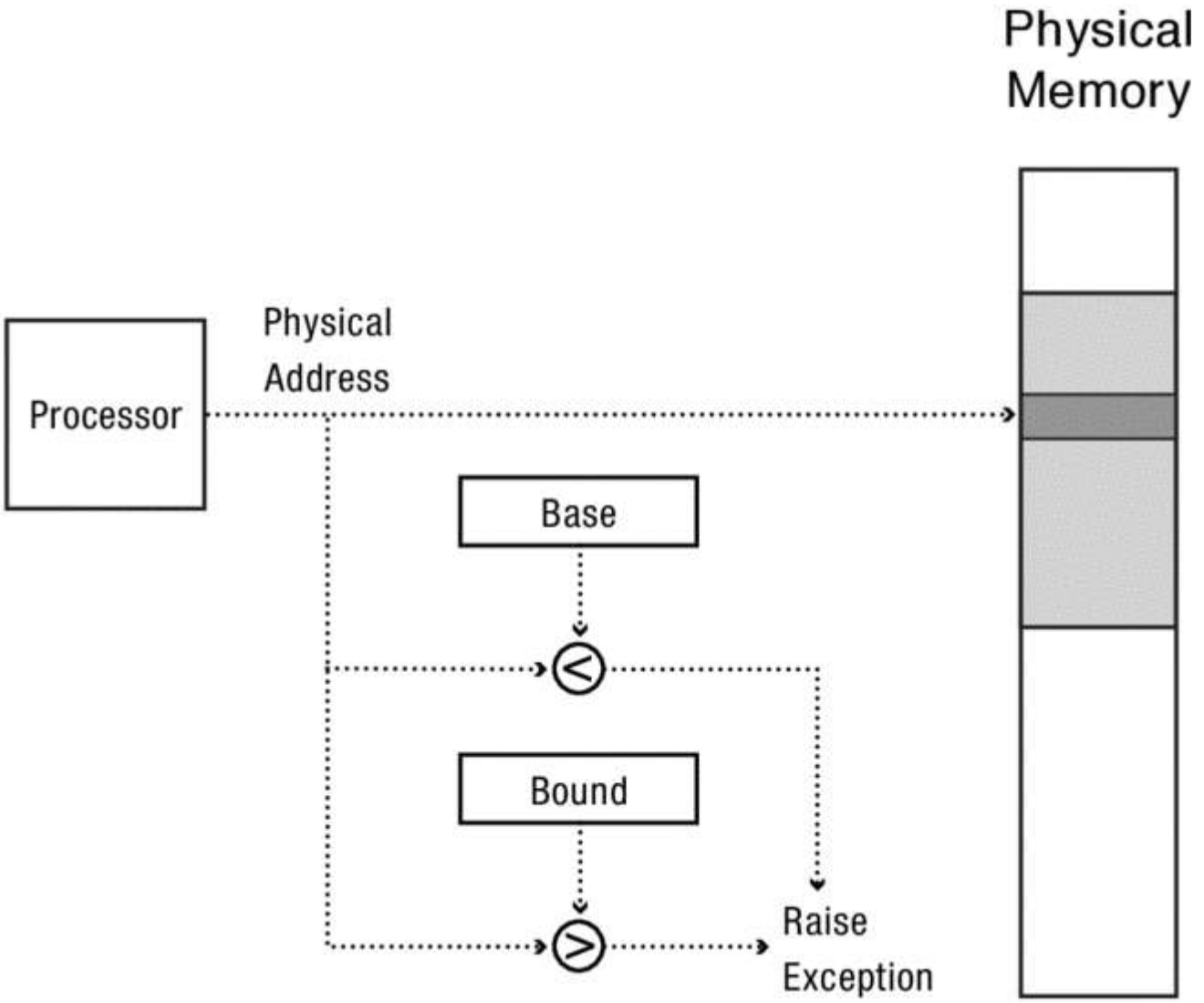
- Fundamental tradeoff between protection and efficiency
  - Communication easier *within* a process
  - Communication harder *between* processes

Single and Multithreaded Processes

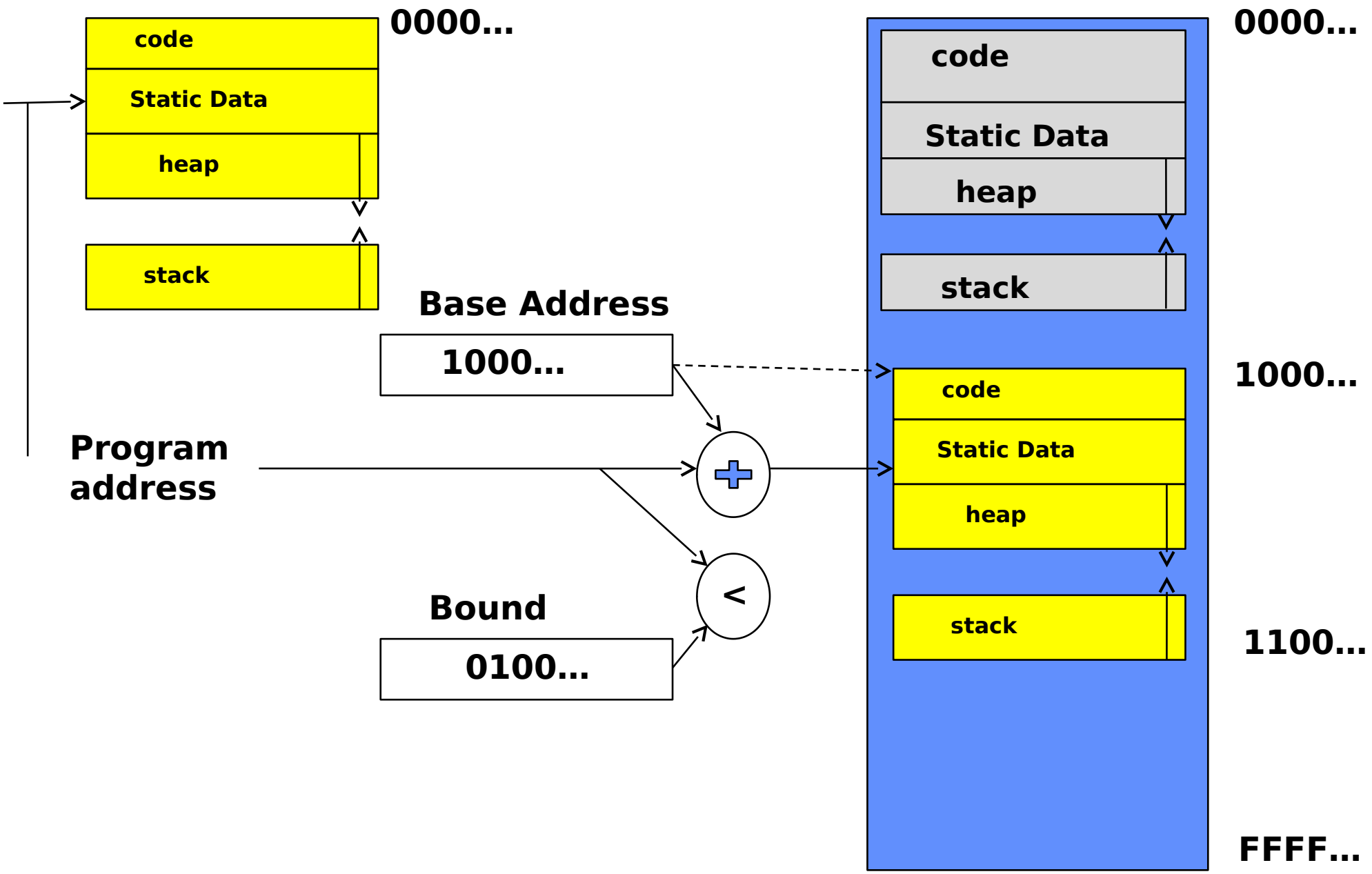


- Threads encapsulate concurrency
- Address spaces encapsulate protection
- Why have multiple threads per address space?
  - Parallelism: take advantage of actual hardware parallelism (e.g. multicore)
  - Concurrency: ease of handling I/O and other simultaneous events

Base and Bound register



A simple address translation with Base and Bound



Limitations

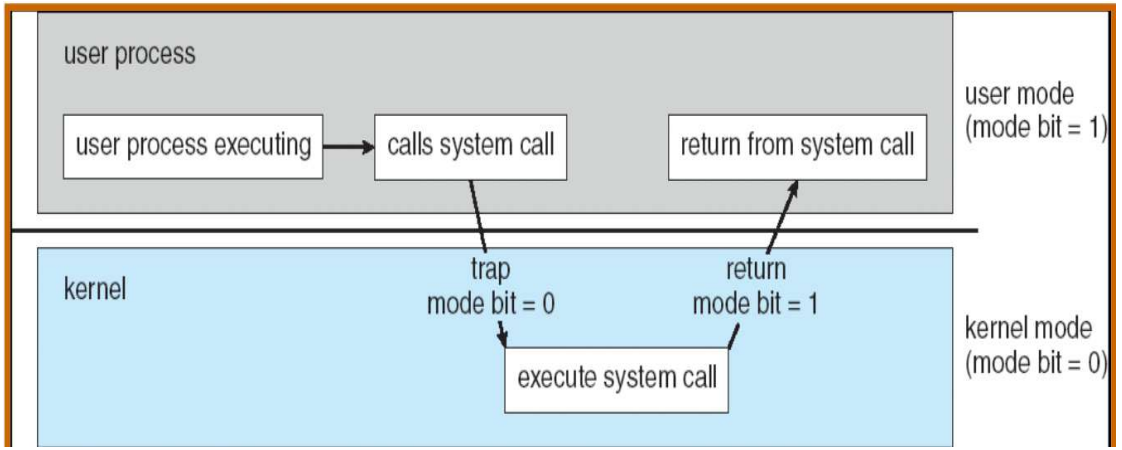
- **Expandable heap and stack**
- **Memory sharing**
- **Memory fragmentation**

Fourth OS Concept: Dual Mode Operation

- Hardware provides at least two modes (at least 1 mode bit):
  1. Kernel Mode (or "supervisor" mode)
  2. User Mode

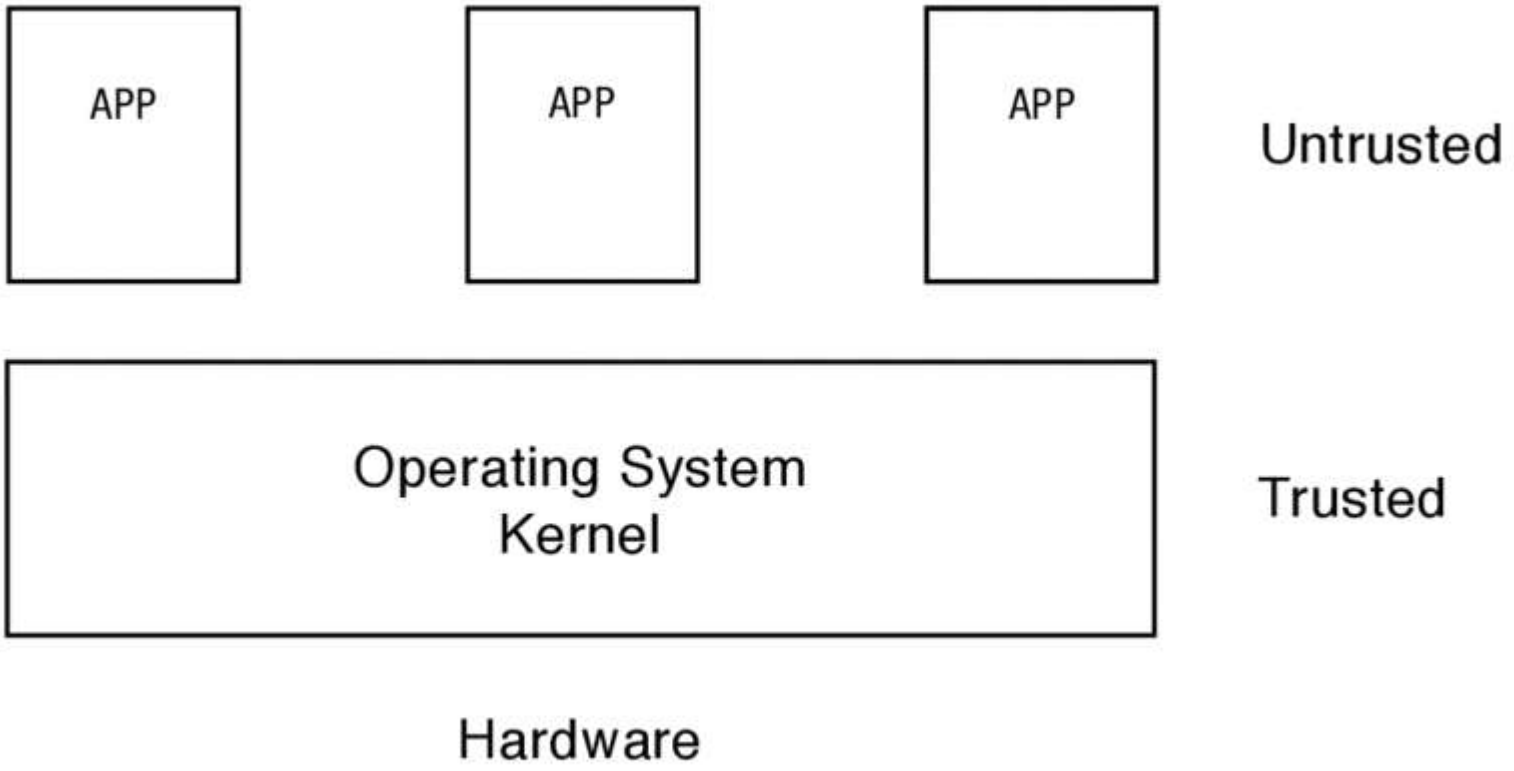
- Certain operations are prohibited when running in user mode
  - e.g. disabling interrupts, interacting directly w/ hardware, writing to kernel memory, etc.

- Carefully controlled transitions between user mode and kernel mode
  - System calls, interrupts, exceptions

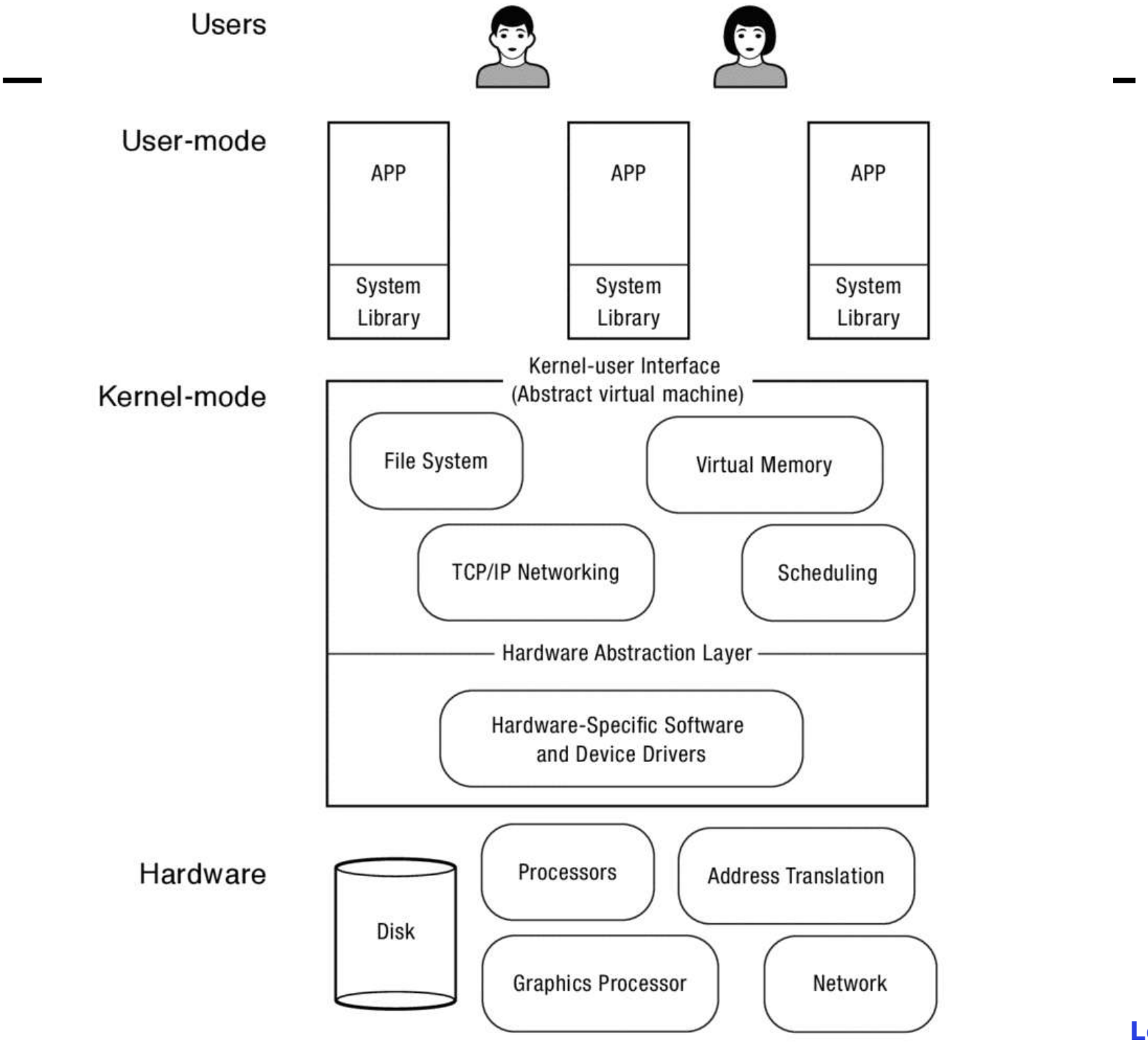


OS Kernel

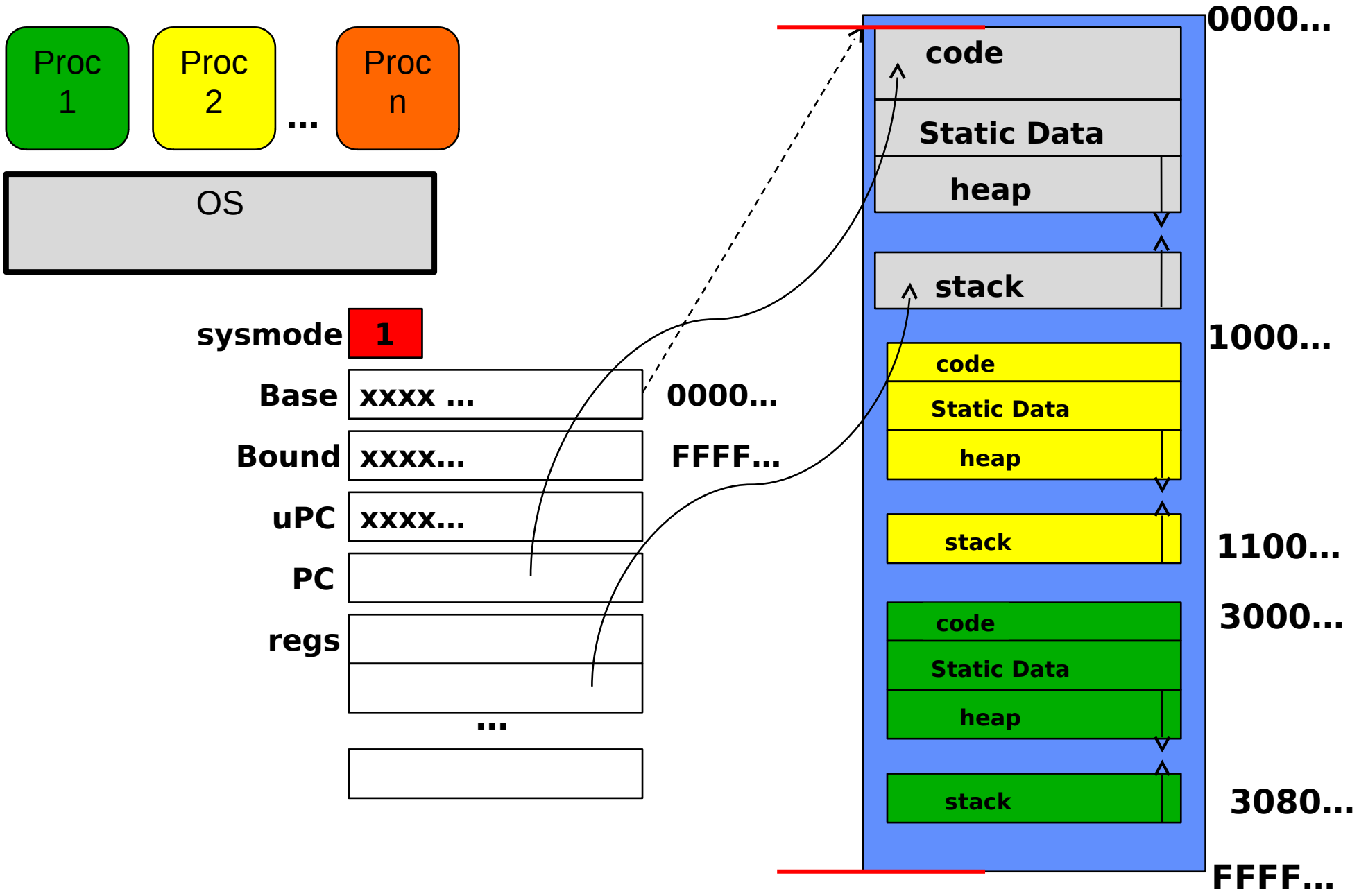
- Protection is implemented by the OS kernel



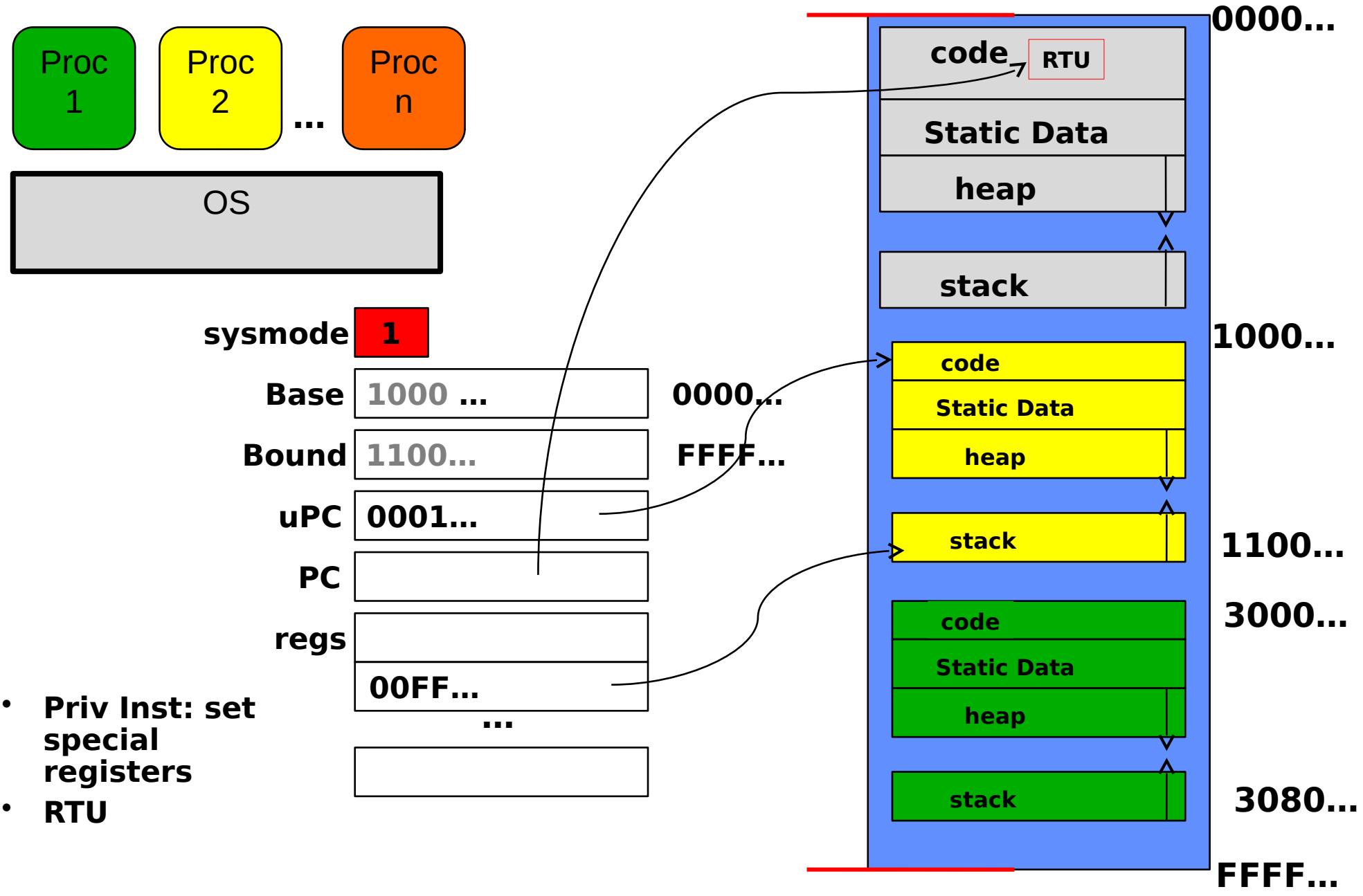
- Both application and OS run on the same machine – same disk, same processor, same memory


Users
User-mode
APP
System Library
APP
System Library
APP
System Library
Kernel-mode
Kernel-user Interface
(Abstract virtual machine)
File System
Virtual Memory
TCP/IP Networking
Scheduling
Hardware Abstraction Layer
Hardware-Specific Software and Device Drivers
Hardware
Disk
Processors
Address Translation
Graphics Processor
Network

Putting pieces together OS loads process

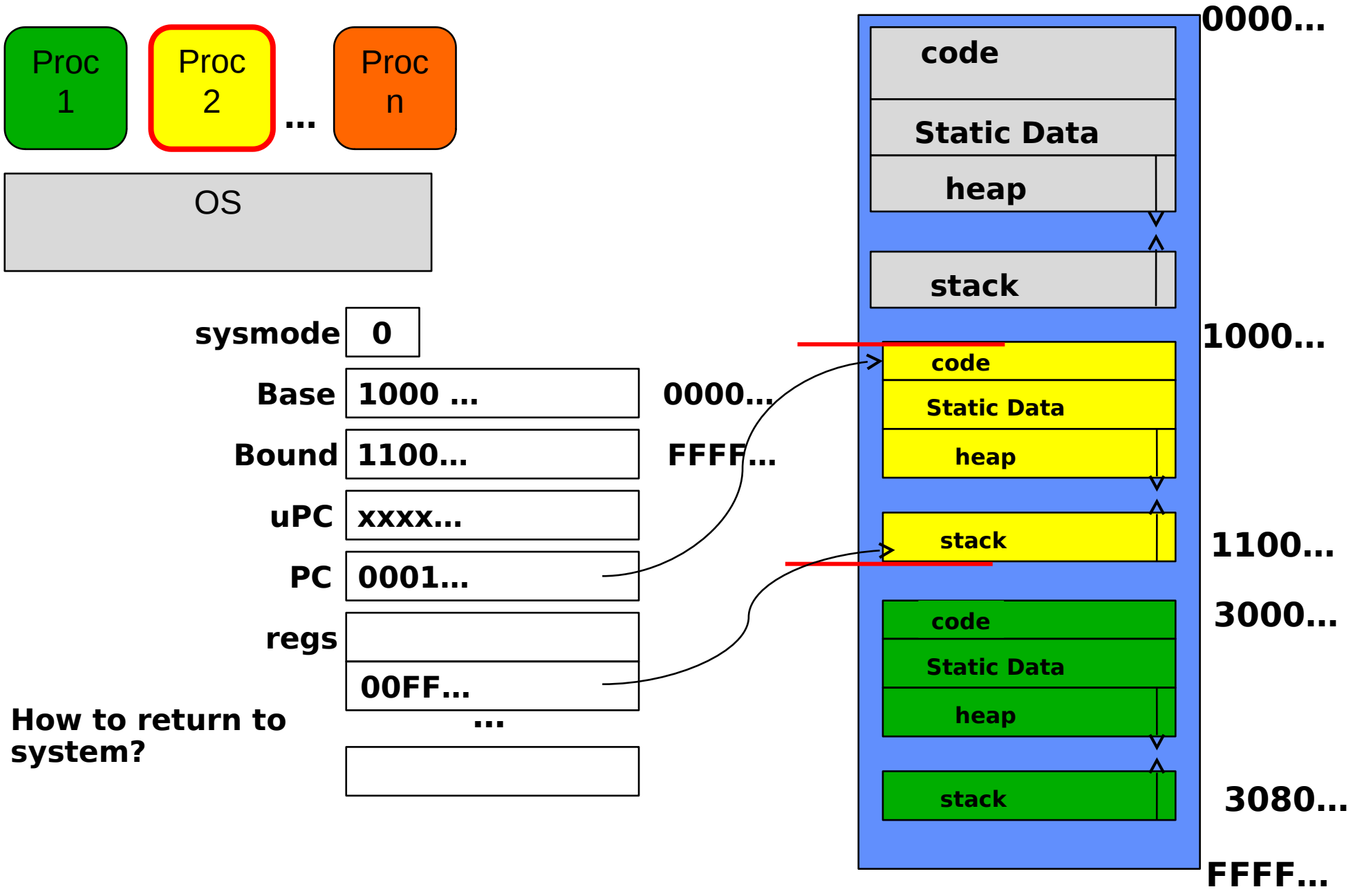


Simple B&B: OS gets ready to switch



- **Priv Inst: set special registers**
- **RTU**

Simple B&B: “Return” to User



- How to return to system?