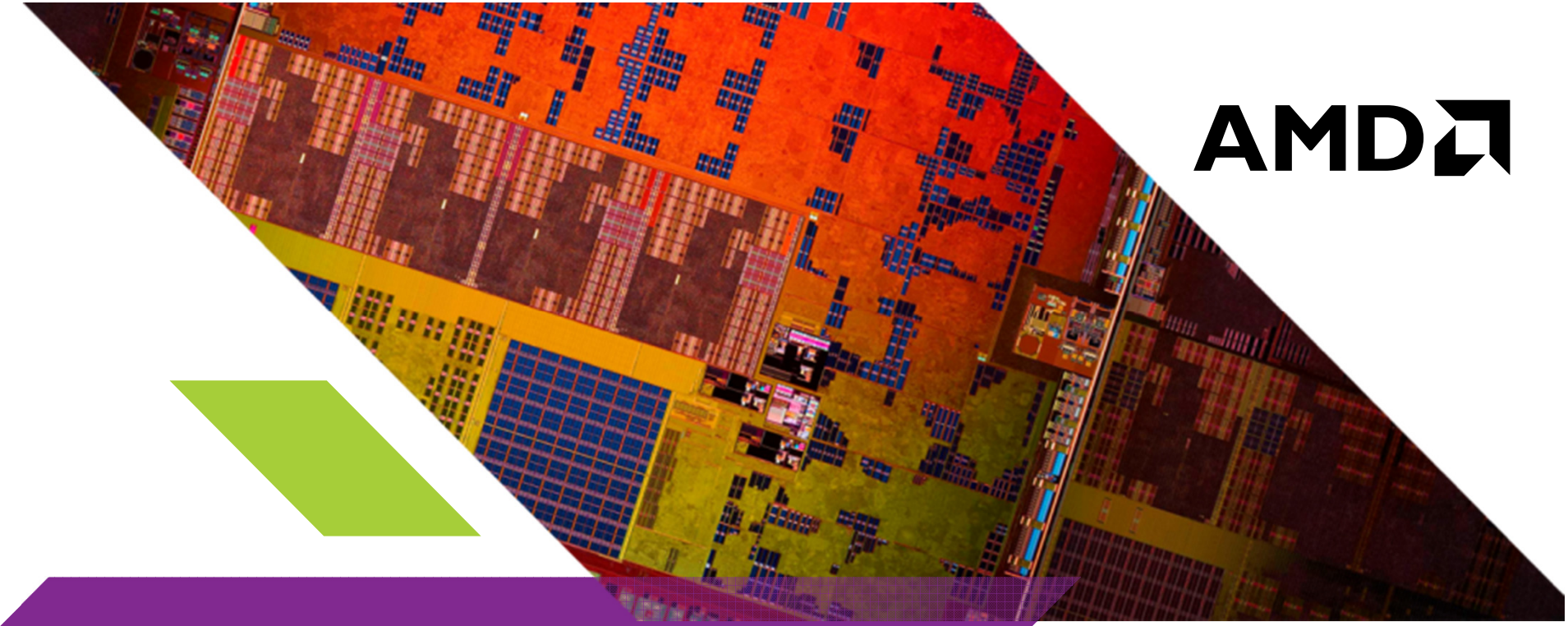


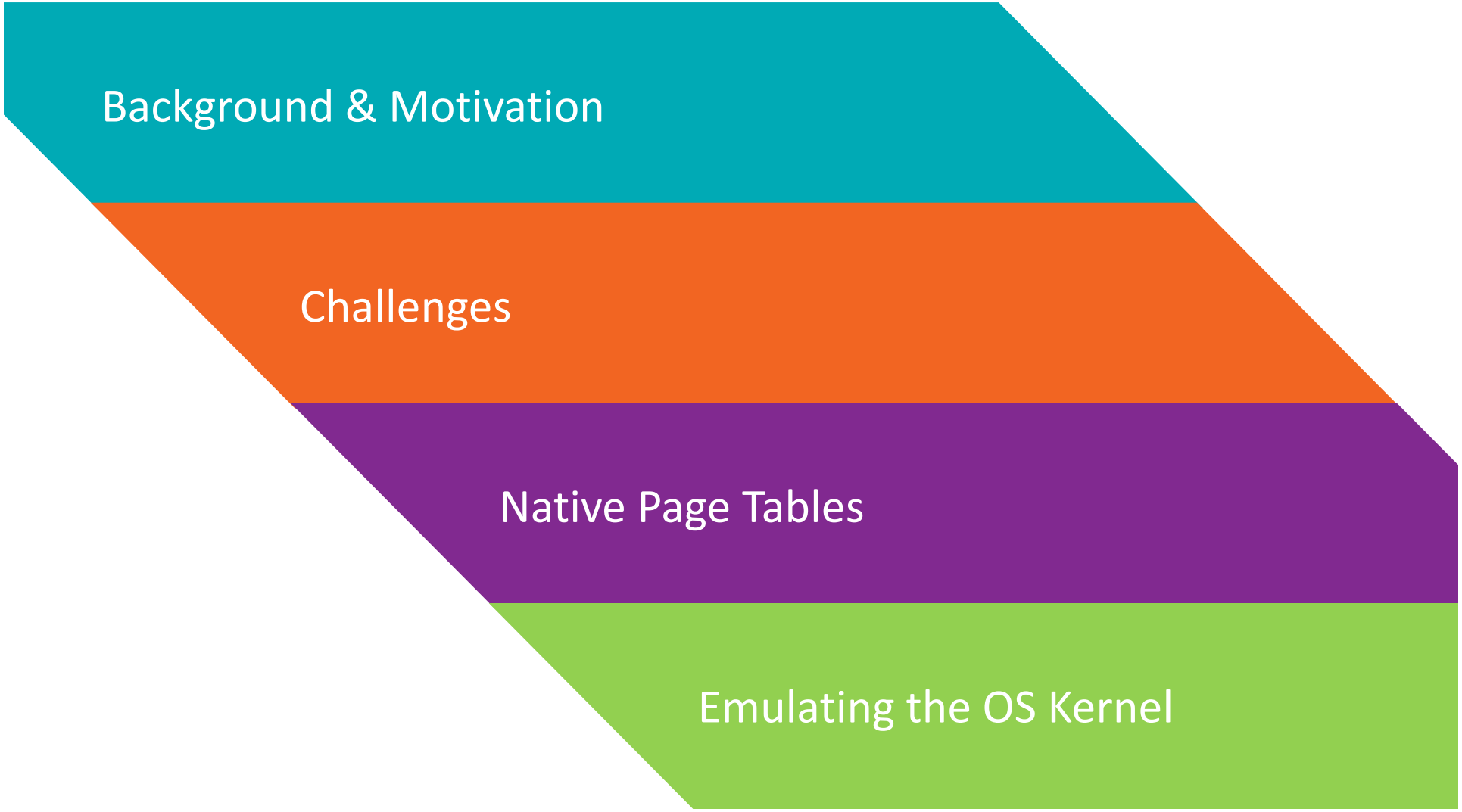
AMD

KVM CPU MODEL IN SYSCALL EMULATION MODE

ALEXANDRU DUTU, JOHN SLICE

JUNE 14, 2015

AGENDA


Background & Motivation
Challenges
Native Page Tables
Emulating the OS Kernel

BACKGROUND

AMD

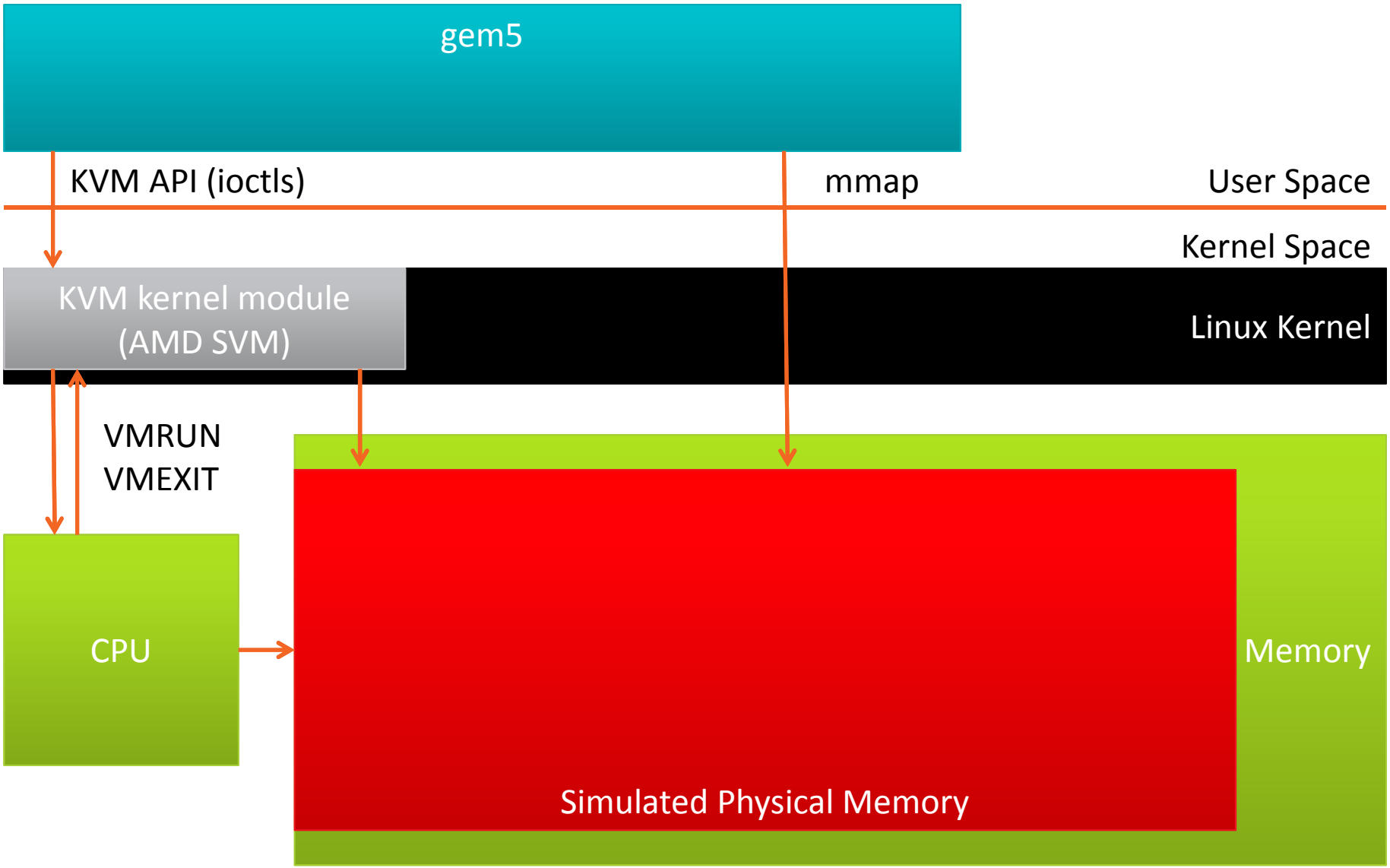
- Kernel-based Virtual Machine (KVM)
  - Linux kernel becomes a hypervisor
  - Driver for user space communication
  - IOCTL user level API
- KVM CPU model
  - Sets up a virtual machine (VM), using KVM
  - Runs code to be simulated inside the VM at hardware speeds
- Syscall Emulation (SE) mode
  - Simulated user space code
  - Emulated kernel space code

WHY KVM IN SE MODE?

AMD

- Fast-forwarding at hardware speeds using KVM is a great capability
  - Currently only available in FS mode
- Syscall Emulation (SE) mode has a number of advantages
  - No need to set up disk images, boot kernel, etc.
  - Debugging & analysis of simulated code is much easier
    - Only your application code is running inside gem5
  - No need to write complete functional device drivers for experimental devices
  - In some environments, the loss of accuracy from not modeling the OS is tolerable
    - Compute-bound applications
    - I/O is done primarily through user-level operations
- Very valuable to be able to combine these

KVM CPU MODEL



KVM CPU MODEL

AMD

- Full-System mode behavior:
  - gem5 loads a kernel image
  - KVM CPU model starts executing the kernel image
  - Simulated OS sets up and manages VM guest page tables within VM
  - Simulated OS handles application exceptions (system calls, page faults) within VM
  - Control returns to gem5 only on MMIO access or to process scheduled event

- Syscall-Emulation mode **desired behavior**:
  - Execute just user space code in a VM
  - Exit the VM when going to kernel space (system calls, page faults)
  - Return to the VM when leaving the kernel space

CHALLENGES

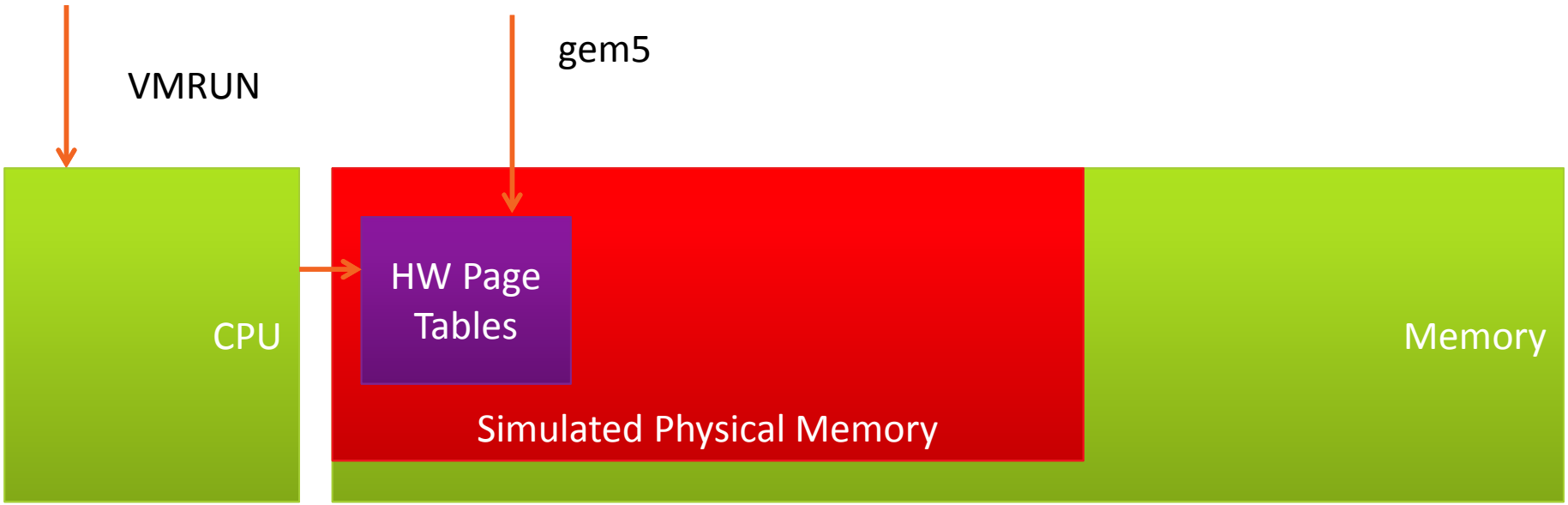
AMD

- Hardware virtualization requires native page tables for guest->host mapping
  - SE layer currently **uses a simple std::map**
  - Need to enable gem5 to build & manipulate native page tables in SE mode
- Need to set up CPU state (control regs etc.) for user-level execution
  - KVM expects **(non-existent) guest OS to set this state**
- Need to redirect application page faults into SE layer
  - Needed to grow stack on demand
  - KVM wants to **redirect these to (non-existent) guest OS**
- Need to redirect system calls into SE layer
  - Again, KVM **lets the (non-existent) guest OS handle these**

NATIVE PAGE TABLES

AMD

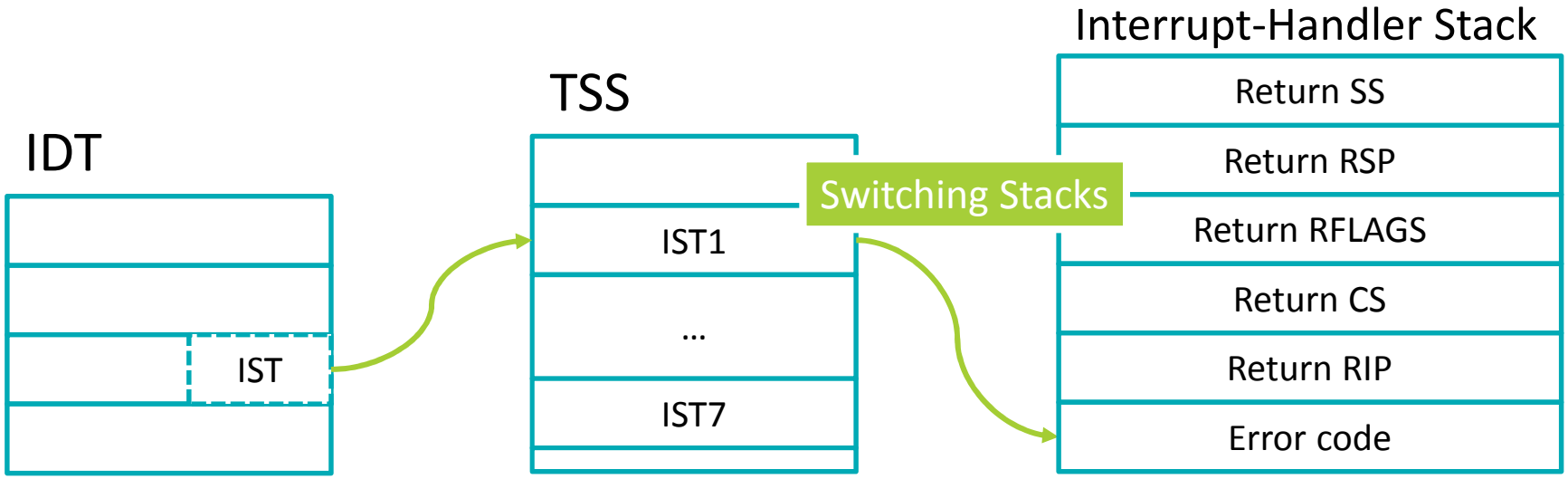
- Reside in simulated physical memory
  - walkable by the VM, during app simulation
  - and by gem5 during the emulation of syscalls
- Compliant with ISA specifications
  - refactored SE page table class to allow optional native page table implementation
  - built template class to handle multi-level page tables, including x86 specialization



EMULATING THE OS KERNEL: SETUP

AMD

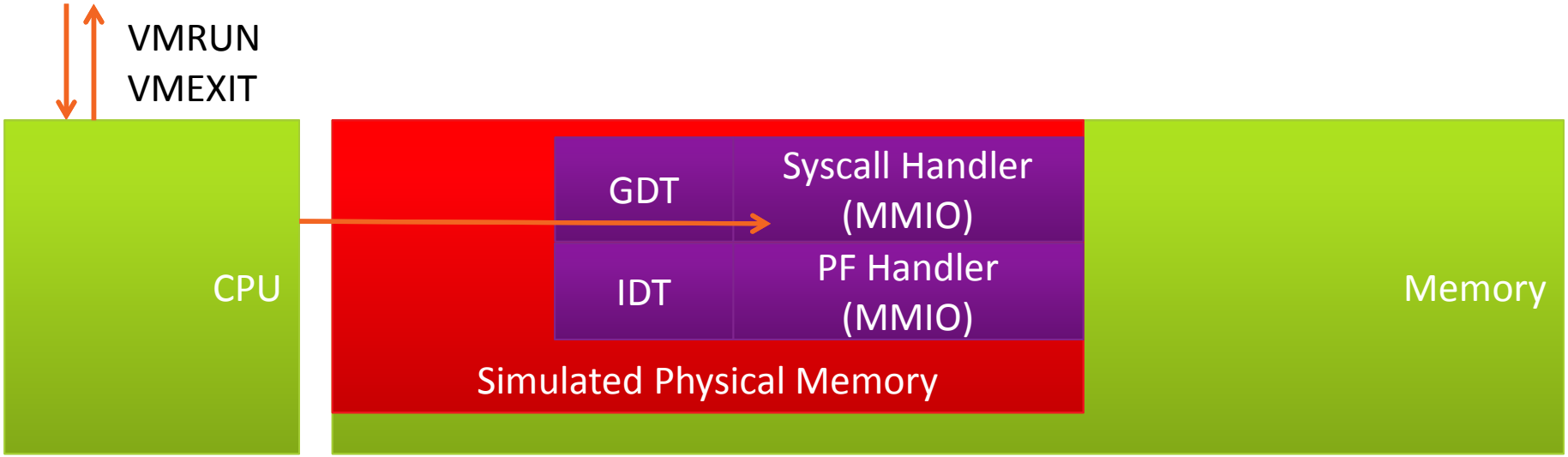
- Getting the VM in a state which can execute user space code:
  - added additional code and data segments for ring 0
  - modified the GDT accordingly
- Enabling exception handling
  - added Interrupt Description Table (IDT)
  - changed the Task State Segment (TSS)
  - added Interrupt Stack Table (IST)



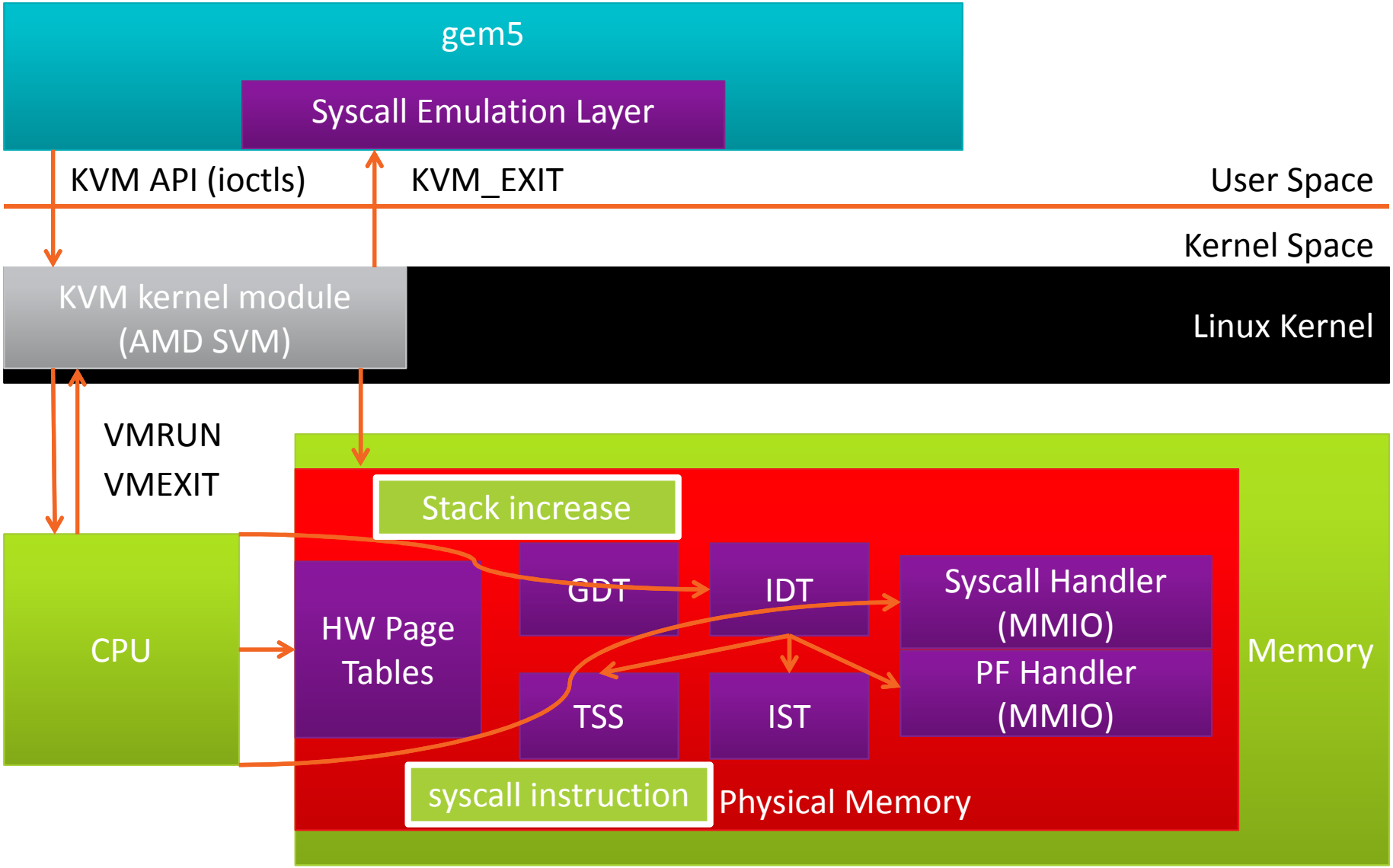
EMULATING THE OS KERNEL: HANDLING EXCEPTIONS

AMD

- Redirecting syscalls to SE layer
  - created small native syscall handler passed to VM
  - the handler triggers KVM exit through an MMIO operation
  - gem5 emulates the syscall
- Redirecting page faults to SE layer
  - created small native page-fault handler passed to VM
  - the handler triggers KVM exit through an MMIO operation
  - gem5 fixes the stack (the only demand paging scenario)



RECAP


gem5
Syscall Emulation Layer
KVM API (ioctls)
KVM_EXIT
User Space
Kernel Space
KVM kernel module
(AMD SVM)
Linux Kernel
VMRUN
VMEXIT
Stack increase
GDT
IDT
Syscall Handler
(MMIO)
CPU
HW Page
Tables
PF Handler
(MMIO)
TSS
IST
Memory
syscall instruction
Physical Memory

CONCLUSION

IS IT POSSIBLE FOR A VM TO EXECUTE JUST USER SPACE CODE?

AMD

Indeed it is, now that gem5…

- uses ISA native page tables shared with the VM
- sets the VM state as if an OS kernel just booted
- supports ISA native syscall and exception handling

DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION