

## Copyright Notice

These slides are distributed under the **Creative Commons Attribution 3.0 License**

- You are free:
  - to share—to copy, distribute and transmit the work
  - to remix—to adapt the work
- under the following conditions:
  - Attribution:** You must attribute the work (but not in any way that suggests that the author endorses you or your use of the work) as follows:

*"Courtesy of Gernot Heiser, UNSW Sydney"*

The complete license text can be found at <http://creativecommons.org/licenses/by/3.0/legalcode>

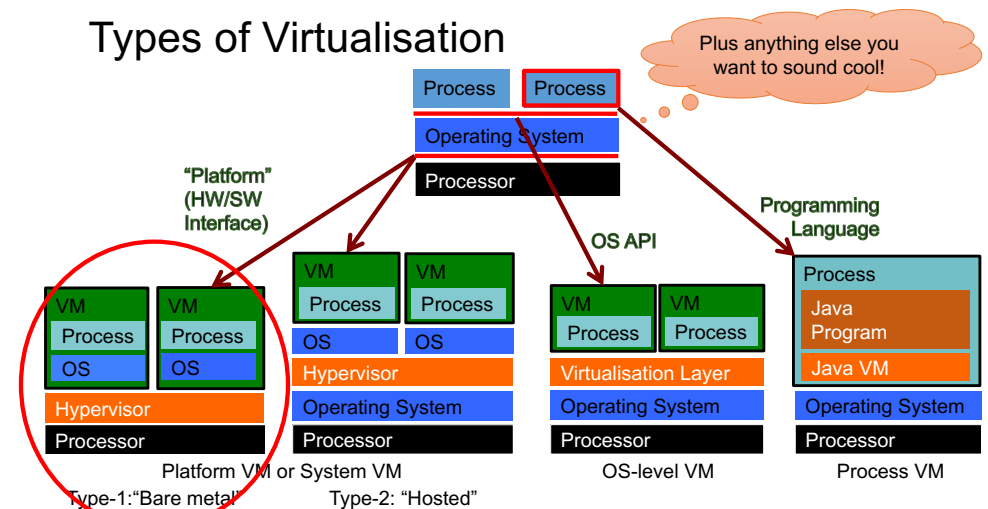
## Virtual Machine (VM)

*"A VM is an efficient, isolated duplicate of a real machine"* [Popek&Goldberg 74]

- Duplicate:** VM should behave identically to the real machine
  - Programs cannot distinguish between real or virtual hardware
  - Except for:
    - Fewer resources (potentially different between executions)
    - Some timing differences (when dealing with devices)
- Isolated:** Several VMs execute without interfering with each other
- Efficient:** VM should execute at speed close to that of real hardware
  - Requires that most instructions are executed directly by real hardware

Hypervisor aka virtual machine monitor (VMM):  
 Software layer implementing the VM

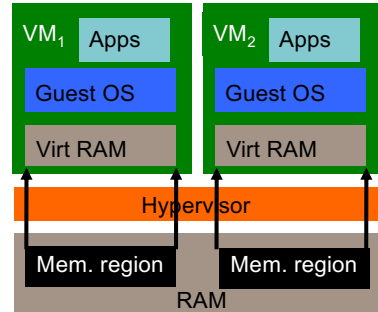
## Types of Virtualisation



# Why Virtual Machines?

- Historically used for easier sharing of expensive mainframes
  - Run several (even different) OSes on same machine
    - called *guest operating system*
  - Each on a subset of physical resources
  - Can run single-user single-tasked OS in time-sharing mode
    - legacy support

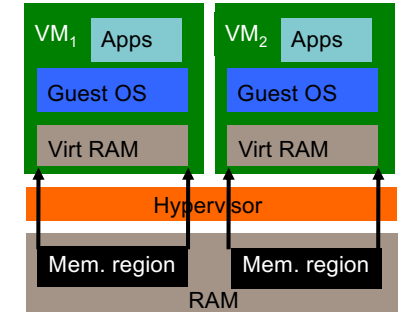
Obsolete by 1980s



# Why Virtual Machines?

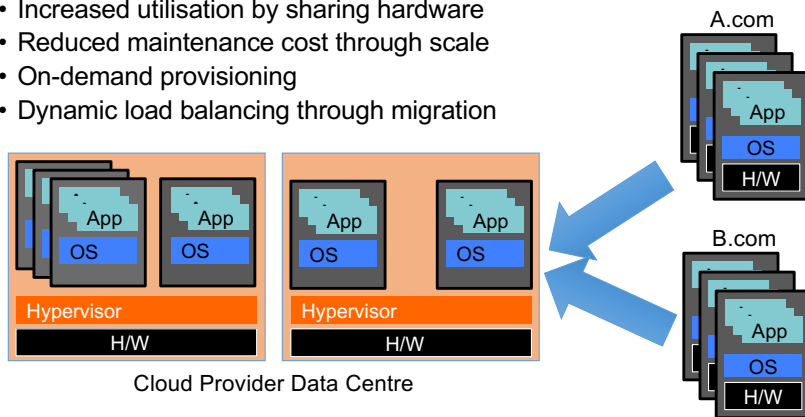
- Heterogenous concurrent guest OSes
  - eg Linux + Windows
- Improved isolation for consolidated servers: QoS & Security
  - total mediation/encapsulation:
    - replication
    - migration/consolidation
    - checkpointing
    - debugging
- Uniform view of hardware

Would not be needed if OSes provided proper security & resource management!



# Why Virtual Machines: Cloud Computing

- Increased utilisation by sharing hardware
- Reduced maintenance cost through scale
- On-demand provisioning
- Dynamic load balancing through migration



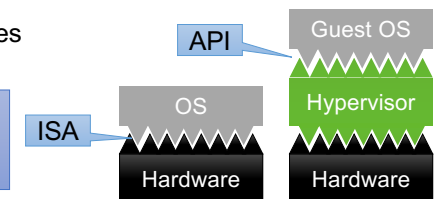
# Hypervisor aka Virtual Machine Monitor

- Software layer that implements virtual machine
- Controls resources
  - Partitions hardware
  - Schedules guests
    - "world switch"
  - Mediates access to shared resources
    - e.g. console, network

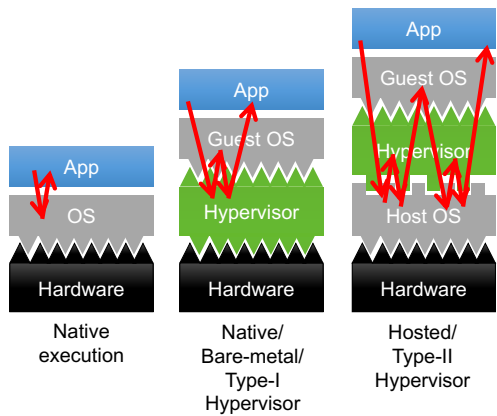
**Implications:**

- Hypervisor executes in *privileged* mode
- Guest software executes in *unprivileged* mode

Privileged guest instructions trap to hypervisor



# Native vs Hosted Hypervisor



- Hosted VMM besides native apps
  - Sandbox untrusted apps
  - Convenient for running alternative OS on desktop
  - leverage host drivers

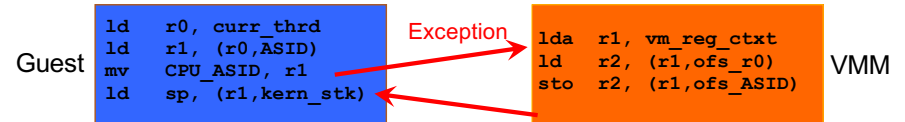
### Overheads:

- Double mode switches
- Double context switches
- Host not optimised for exception forwarding



# Virtualisation Mechanics: Instruction Emulation

- Traditional *trap-and-emulate* (T&E) approach:
  - guest attempts to access physical resource
  - hardware raises exception (trap), invoking HV's exception handler
  - hypervisor emulates result, based on access to virtual resource



Most instructions do not trap

- prerequisite for efficient virtualisation
- requires VM ISA (almost) same as processor ISA



# Trap & Emulate Requirements

- **Privileged instruction:** when executed in user mode will *trap*
- **Privileged state:** determines resource allocation
  - Incl. privilege level, PT ptr, exception vectors...
- **Sensitive instruction:**
  - **control sensitive:** change privileged state
  - **behaviour sensitive:** expose privileged state
    - eg privileged instructions which NO-OP in user state
- **Innocuous instruction:** not sensitive

No-op is insufficient!

- Some inherently sensitive, e.g. set interrupt level
- Some context-dependent, e.g. store to page table

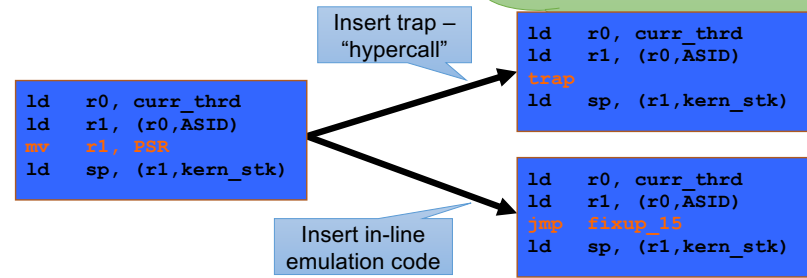
Can run unmodified guest binary

**T&E virtualisable HW:**  
All sensitive instructions are privileged



# "Impure" Virtualisation

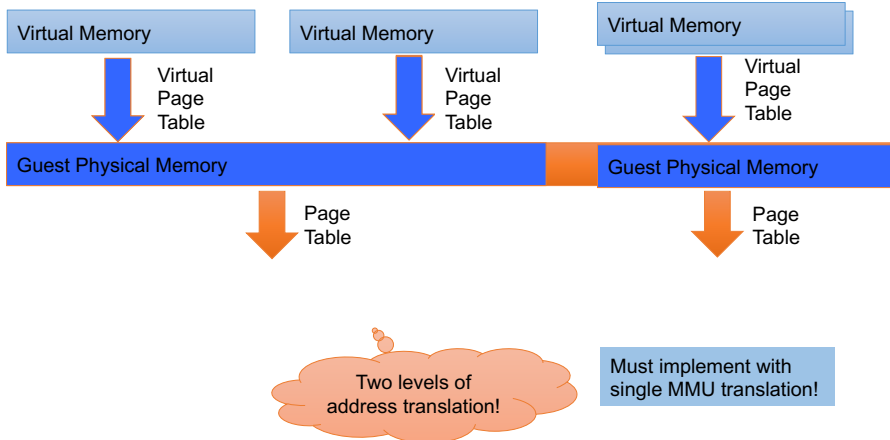
- Support non-T&E hardware
- Improve performance



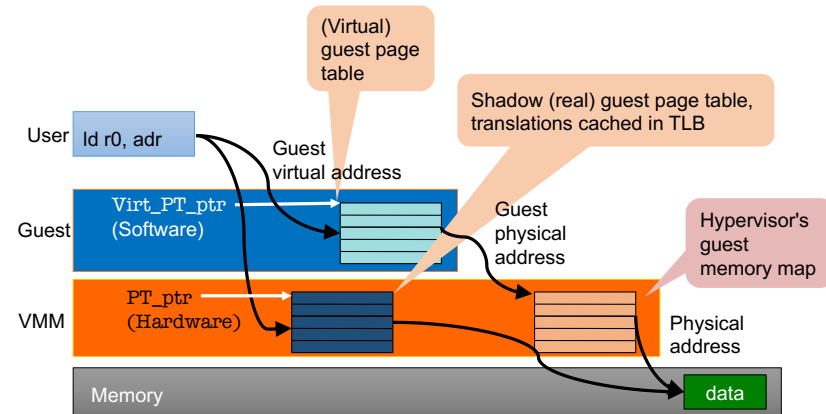
- Modify binary: *binary translation* (VMware)
- Modify hypervisor "ISA": *para-virtualisation*



# Virtualisation vs Address Translation



# Virtualisation Mechanics: Shadow Page Table



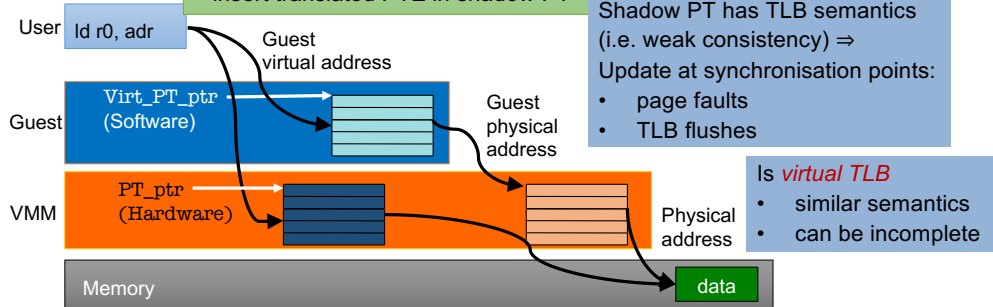
# Mechanics: Shadow Page Table

Used by VMware

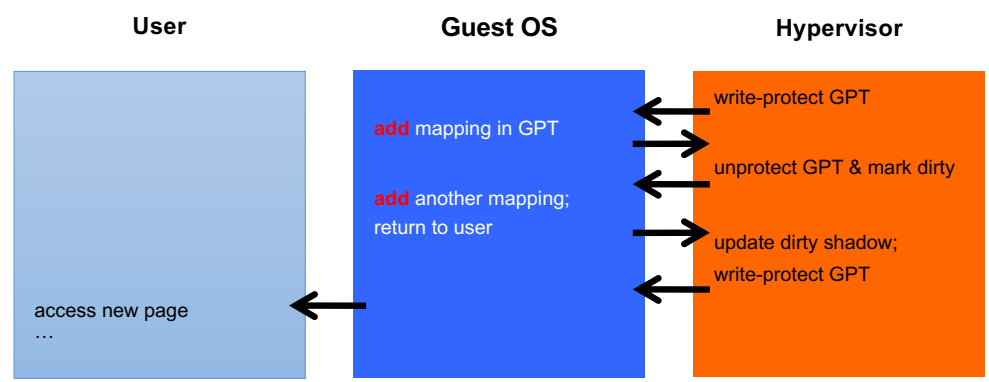
- Hypervisor must shadow (virtualize) PT updates by guest:
- trap guest writes to guest PT
  - translate guest PA in guest (virtual) PTE using memory map
  - insert translated PTE in shadow PT

- Shadow PT has TLB semantics (i.e. weak consistency) ⇒ Update at synchronisation points:
- page faults
  - TLB flushes

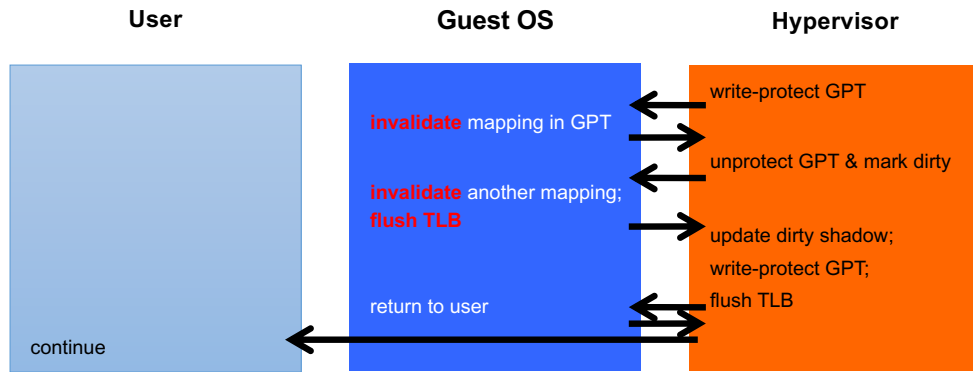
- Is *virtual TLB*
- similar semantics
  - can be incomplete



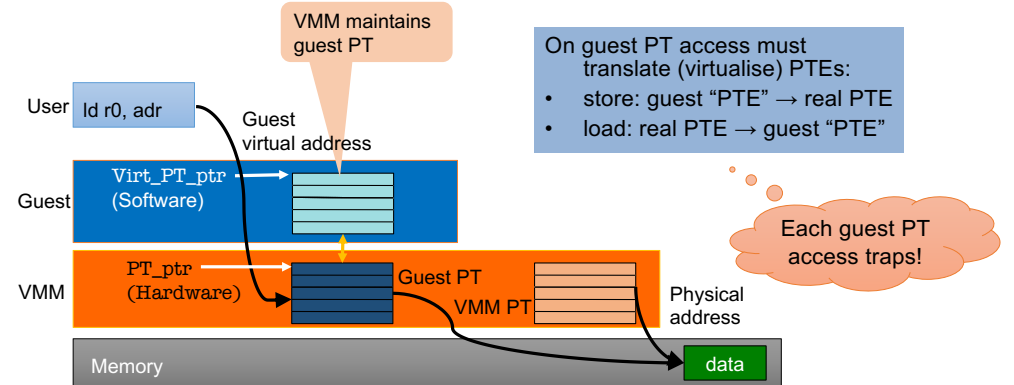
# Mechanics: Lazy Shadow Update



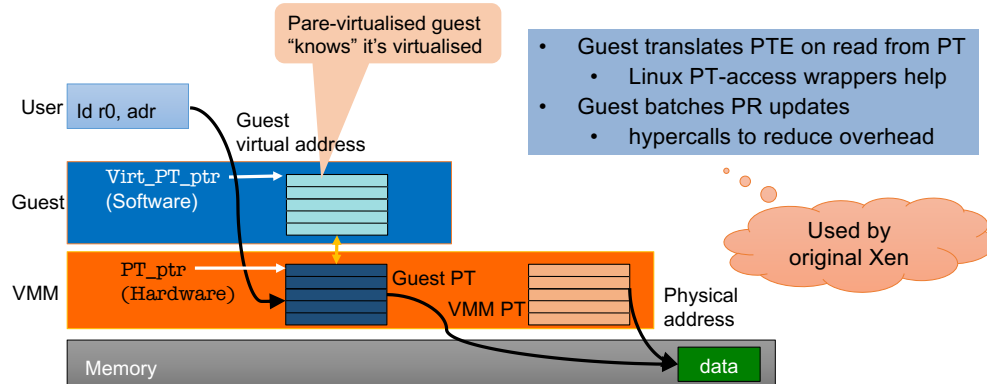
## Mechanics: Lazy Shadow Update



## Mechanics: Real Guest Page Table



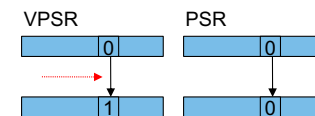
## Mechanics: Optimised Guest Page Table



## Mechanics: Guest Self-Virtualisation

Example: Interrupt-enable in virtual PSR

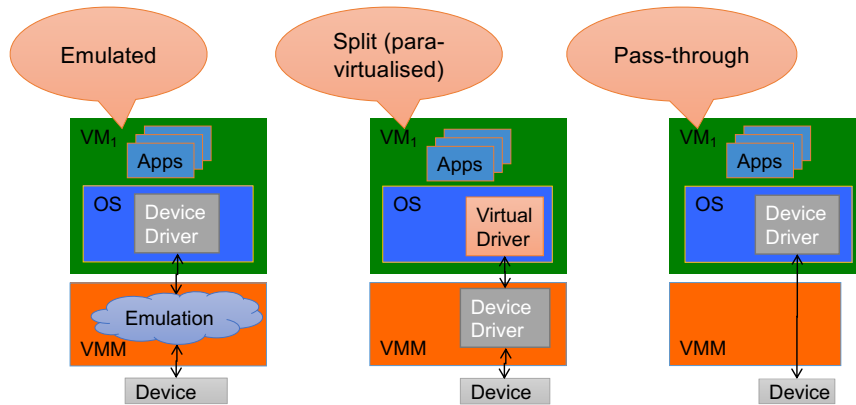
- guest and VMM agree on VPSR location
- VMM queues guest IRQs when disabled in VPSR



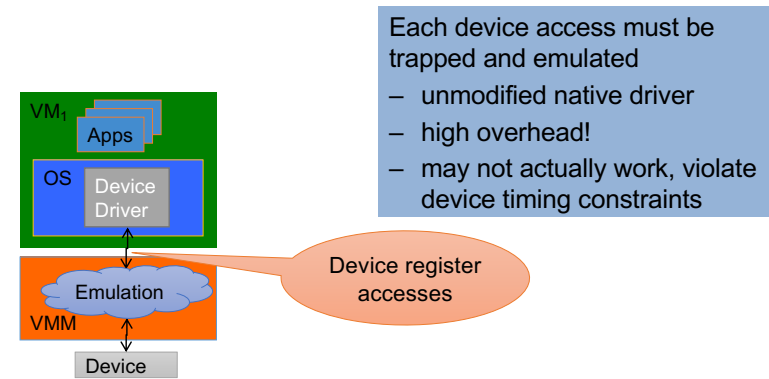
```
mov r1, #VPSR
ldr r0, [r1]
orr r0, r0, #VPSR_ID
sto r0, [r1]
```

Minimise traps by holding some virtual state inside guest

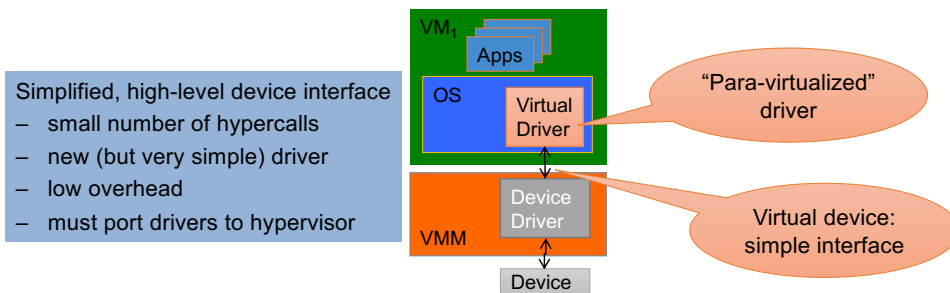
## Mechanics: Device Models



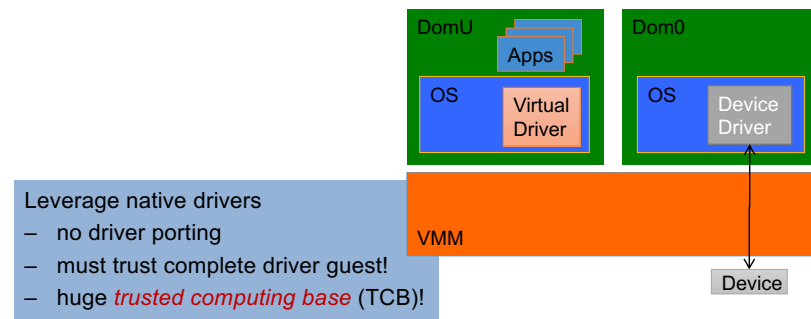
## Mechanics: Emulated Device



## Mechanics: Split Driver



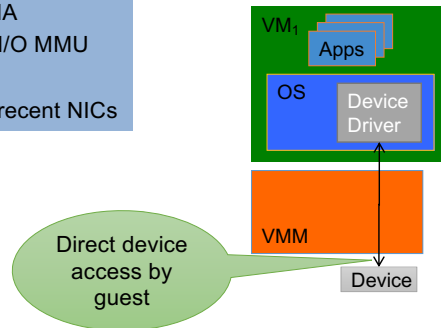
## Mechanics: Driver OS (Xen Dom0)



# Mechanics: Pass-Through Driver

## Unmodified native driver

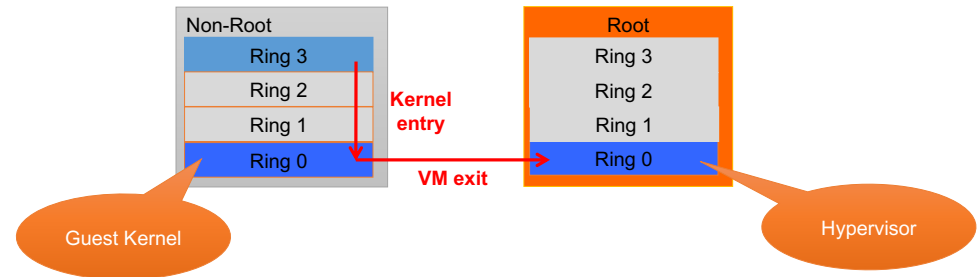
- Must trust driver (and guest) for DMA
  - except with hardware support: I/O MMU
- Can't share device between VMs
  - except with hardware support: recent NICs



# x86 Virtualisation Extensions: VT-x

## New processor mode: VT-x root mode

- orthogonal to protection rings
- entered on virtualisation trap

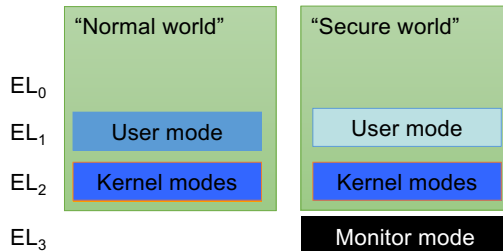


# Arm Virtualisation Extensions (1)

## EL<sub>2</sub> aka "hyp mode"

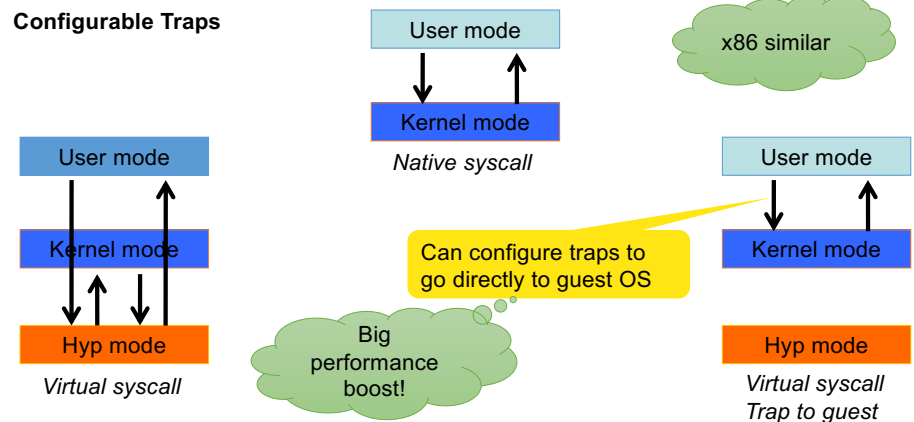
### New privilege level

- Strictly higher than kernel (EL<sub>1</sub>)
- Virtualizes or traps *all* sensitive instructions
- Presently only available in Arm TrustZone "normal world"



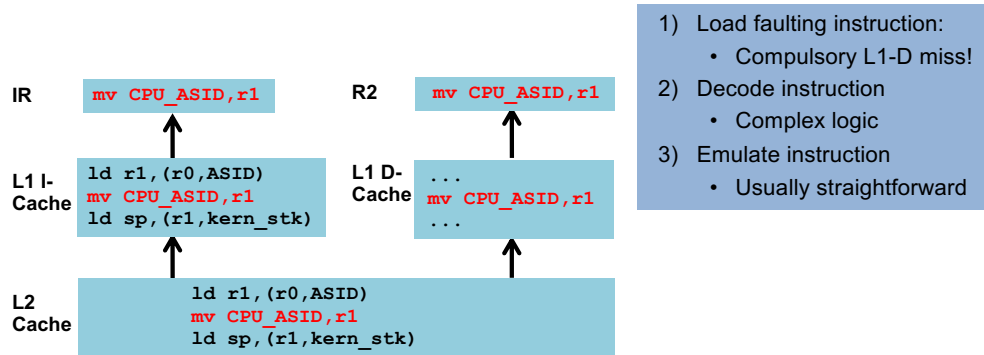
# Arm Virtualisation Extensions (2)

## Configurable Traps



# Arm Virtualisation Extensions (3)

## Emulation

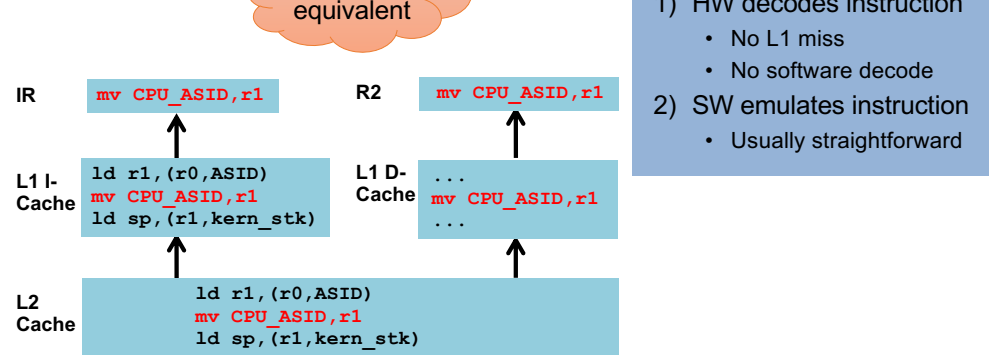


- 1) Load faulting instruction:
  - Compulsory L1-D miss!
- 2) Decode instruction
  - Complex logic
- 3) Emulate instruction
  - Usually straightforward

# Arm Virtualisation Extensions (3)

## Emulation

No x86 equivalent

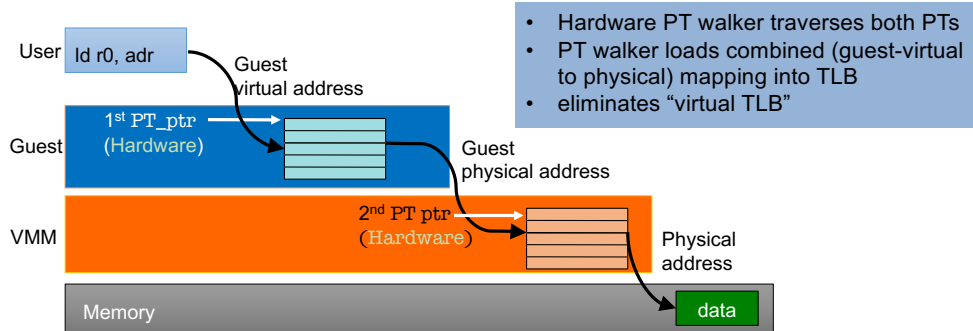


- 1) HW decodes instruction
  - No L1 miss
  - No software decode
- 2) SW emulates instruction
  - Usually straightforward

# Arm Virtualisation Extensions (4)

## 2-stage translation

x86 similar (EPTs)



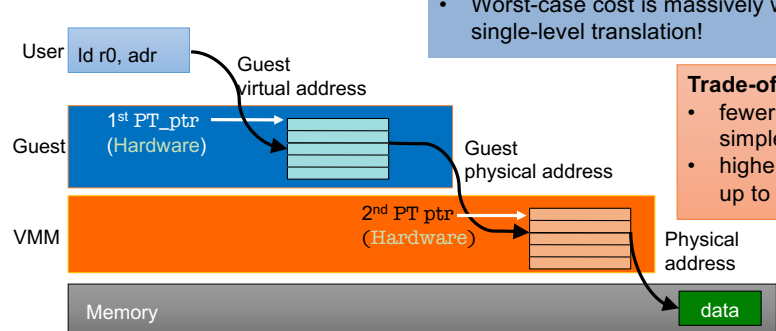
# Arm Virtualisation Extensions (4)

## 2-stage translation cost

- On page fault walk twice number of page tables!
- Can have a page miss on each, requiring PT walk
- $O(n^2)$  misses in worst case for n-level PT
- Worst-case cost is massively worse than for single-level translation!

**Trade-off:**

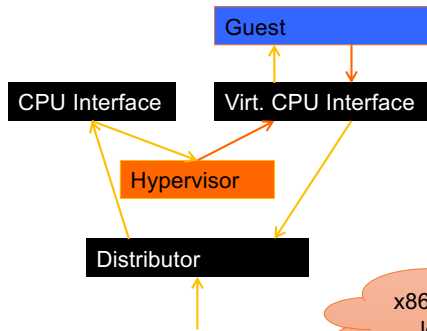
- fewer traps
- simpler implementation
- higher TLB-miss cost up to 50% of run-time!





## Arm Virtualisation Extensions (5)

### Virtual Interrupts

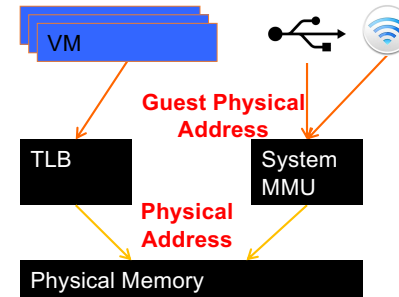


- 2-part IRQ controller
  - global “distributor”
  - per-CPU “interface”
- New H/W “virt. CPU interface”
  - Mapped to guest
  - Used by HV to forward IRQ
  - Used by guest to acknowledge
- Halves hypervisor invocations for interrupt virtualization

x86: issue only for legacy level-triggered IRQs

## Arm Virtualisation Extensions (6)

### System MMU (I/O MMU)



- Devices use virtual addresses
- Translated by *system MMU*
  - elsewhere called I/O MMU
  - translation cache, like TLB
  - reloaded from I/O page table

x86 different (VT-d)

Many ARM SoCs different

- Can do pass-through I/O safely
  - guest accesses device registers
  - no hypervisor invocation

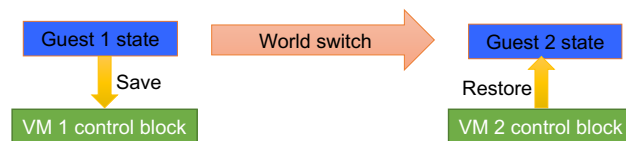
## World Switch

### x86

- VM state is up to 4 KiB
- **Save/restore done by hardware** on VMexit/VMentry
- Fast and simple

### Arm

- VM state is 488 B
- **Save/restore done by hypervisor**
- Selective save/restore
  - Eg traps w/o world switch

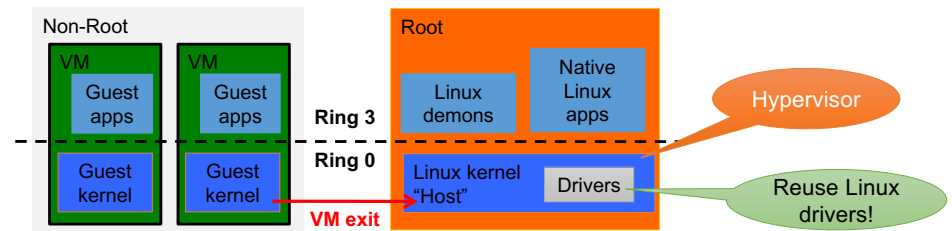


## Hybrid Hypervisor-OSes

Huge TCB, contains full Linux system (kernel and userland)!

Often falsely called a “Type-2” hypervisor

Idea: Turn OS into hypervisor by running in VT-x root mode, pioneered by KVM



## Fun and Games with Hypervisors

... and many more..

- Time-travelling virtual machines [King '05]
  - debug backwards by replaying VM from checkpoint, log state changes
- SecVisor: kernel integrity by virtualisation [Seshadri '07]
  - controls modifications to kernel (guest) memory
- Overshadow: protect apps from OS [Chen '08]
  - make user memory opaque to OS by transparently encrypting
- Turtles: Recursive virtualisation [Ben-Yehuda '10]
  - virtualize VT-x to run hypervisor in VM
- CloudVisor: mini-hypervisor underneath Xen [Zhang '11]
  - isolates co-hosted VMs belonging to different users
  - leverages remote attestation (TPM) and Turtles ideas