# OS support in ARMv7-A

Matt Horsnell

Senior Research Engineer, ARM Ltd.

# Outline

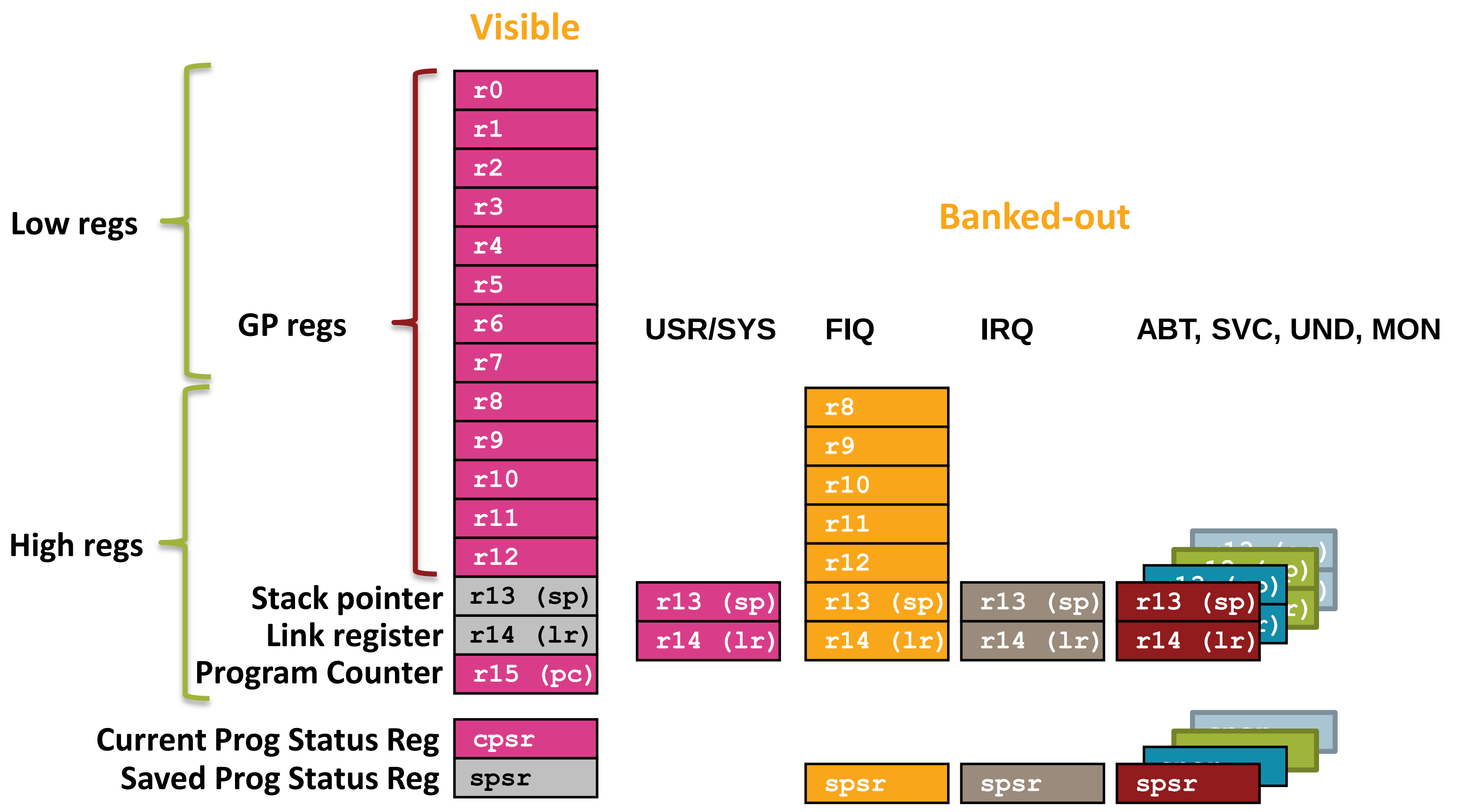
- ARM
- ARM Architecture
- ARM v7A – OS support
  - Memory ordering and MP
  - Exception/Interrupt handling
  - Power management
  - Security
  - Virtualization
  - big.LITTLE
- ARM for OS research

## ARM

- The ARM architecture is now pervasive in many markets
- ~25% of all electronic products contain at least one ARM
- ARM designs processors, cell libraries, and associated IP
- Cambridge HQ
  - Multiple sites worldwide (San Jose, Austin, Sophia, Bangalore…)
- > 2,000 employees

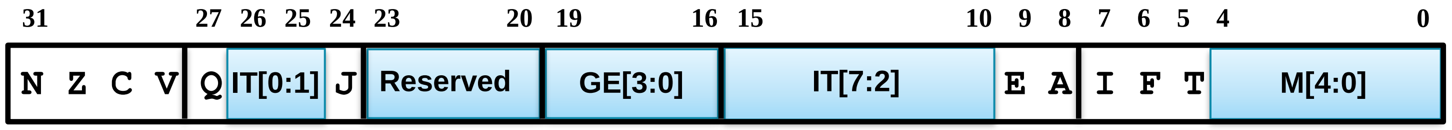
## ARM v7-A

- 32-bit, RISC, load/store architecture
- Thumb2 – 16/32-bit instruction set
- At least 7 modes ... +MON, +HYP



The Architecture for the Digital World®

# ARM v7-A



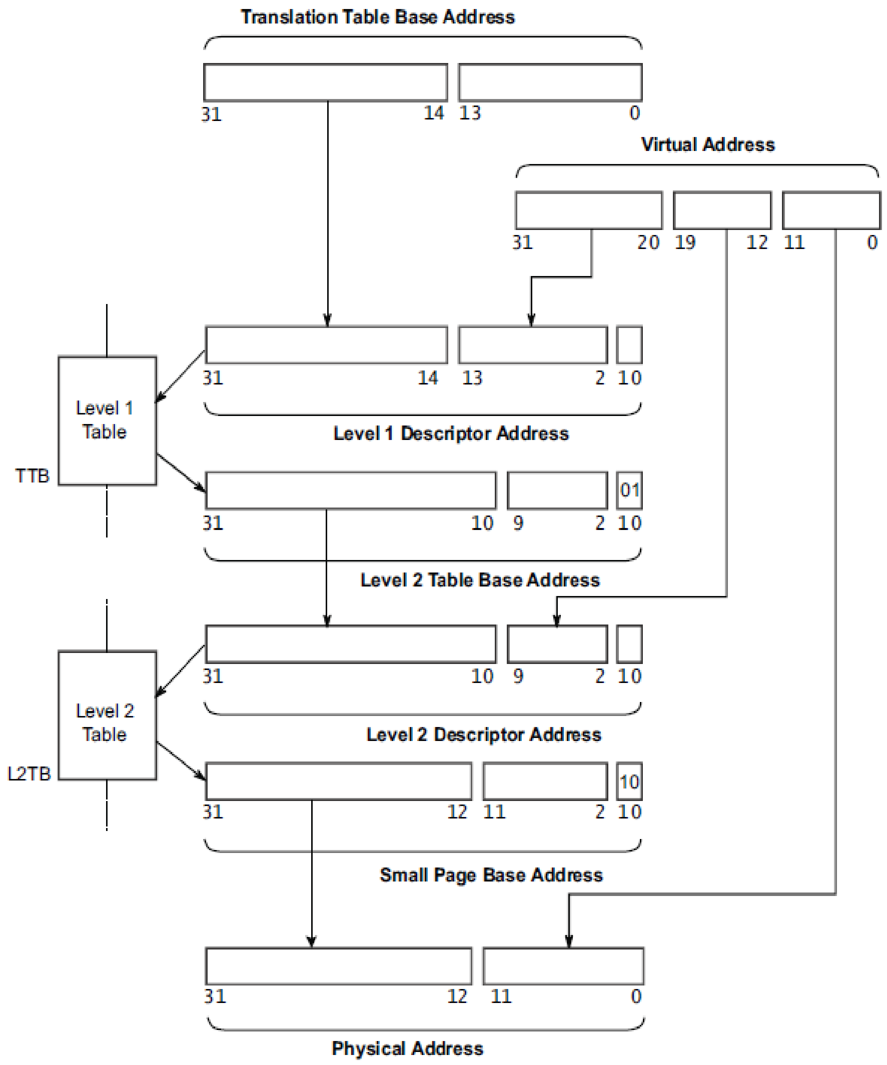
- Program status register
  - Conditions: Negative, Zero, Carry, oVerflow, saturation
  - States: Jazelle, Thumb
  - Modes: USR, SYS, FIQ, IRQ, ABT, SVC, UND, MON
  - Endianess:  load/stores
  - Interrupts: disable bits

## ARM v7-A

- Cortex A (applications) class processors
- VFP and NEON
- Backward compatible
- Full 4GB virtual and physical address space
- Efficient hardware page table walking from V to P
  - VM page sizes (4KB, 64KB, 1MB, 16MB)
  - Cacheability and access permissions per page basis
- Big-endian/Little-endian support
- Unaligned access support
- SMP support on MPCore™ variants
- PIPT data caches

# ARM v7-A – MMU

- 2-level MMU
  - V2P translation
  - cacheability, permissions
- L1 page table
  - 4GB into 1MB (16KB)
  - 4 entry types:
    Fault, L2 PTR, section, super-section
- L2 page table
  - 1MB into 64 or 16KB
    - consumes 1KB
    - 3 entry types:
      Fault, large page, small page



# ARM v7-A - LPAE

- Translation of 32-bit virtual to ≤ 40-bit physical addresses
  - ease pressure on 4GB limit for IO and memory
- Each process can access 4GB, but system wide access to 1TB
  - Multiple large process can remain resident
- Support added
  - Hierarchical permissions
  - Contiguous page hints
  - “Privileged never execute”
  - ASID stored in TTBR
  - Simplified fault encoding

# ARM - memory ordering

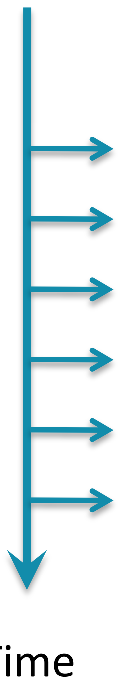
- Weakly ordered

Program Order

```
1:   STR R12,  [R1]

2:   LDR  R0,  [SP], #4

3:   LDR  R2,  [R3,#8]
```



Execution Timeline

- Access 1 goes to write buffer
- Access 2 causes a cache lookup which misses
- Access 3 causes a cache lookup which hits
- Access 3 returns data into ARM register
- Cache line-fill triggered by Access 2 returns data
- Memory store triggered by Access 1 is performed

## ARM - memory ordering

- barriers - instructions that apply ordering constraints

- **DSB -** Data synchronization barrier
  - processor waits for all pending explicit data accesses to complete before any further instructions are executed.

- **DMB –** Data memory barrier
  - all memory access in program order before the barrier must complete before any memory access after the barrier.

- **ISB –** Instruction synchronization barrier
  - flushes the pipeline and pre-fetch buffer so that all instructions following the ISB are fetched from cache or memory after the instruction has completed.

# ARM - memory ordering

- Processor A:

```
STR R0, [Addr1]
LDR R1, [Addr2]
```

  Processor B:

```
STR R2, [Addr2]
LDR R3, [Addr1]
```

- No ordering constraints
  - Assume no hardware implied ordering
  - No software barriers

- Valid outcomes?

# ARM - memory ordering

- Processor A:

```
STR R0, [Msg]           @ write new data into postbox
DMB                     @ ensure data in postbox
STR R1, [Flag]          @ new data is ready to read
STR R1, [Flag]          @ new data is ready to read
```

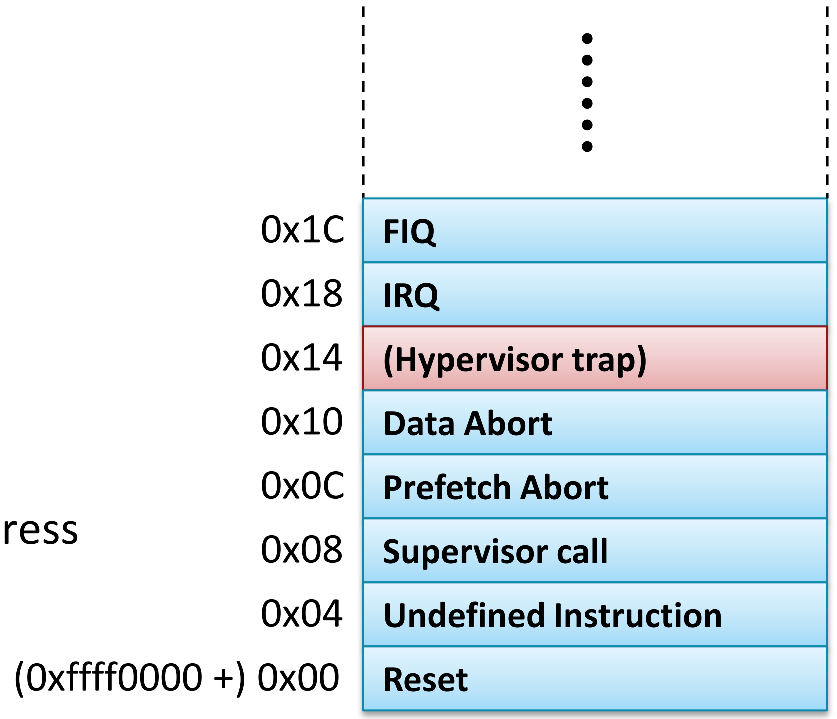
Processor B:

```
Poll_loop:
        LDR R1, [Flag]
        CMP R1,#0                @ is the flag set yet?
        BEQ Poll_loop
        DMB                      @ ensure flag set
        LDR R0, [Msg]            @ read new data.
        LDR R0, [Msg]            @ read new data.
```

- barriers are expensive
- incoherent instruction caches

# ARM – exception handling

- exception
  - any condition that halts normal execution
  - requires handler routine
- exception entry
  - preserves addr of next instruction in LR
  - copy CPSR into SPSR
  - modifies CPSR bits
  - forces the PC to the exceptions vector address
- exception exit
  - restore CPSR from SPSR
  - set the PC using the LR



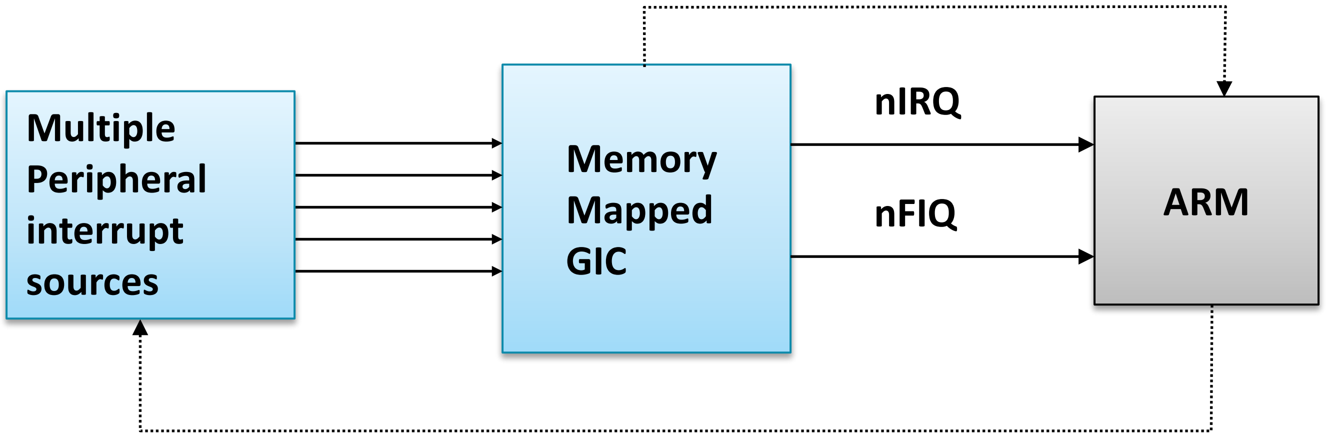
kernel/arch/arm/init.c

**Vector Table**

kernel/include/arch/arm/exceptions.h

# ARM – interrupt handling

- SoCs have a wide range of external source interrupts
  - these are mapped onto interrupts which generate exceptions
- FIQ/IRQ
- Generic Interrupt Controller (GIC)
  - memory mapped registers
  - manages the delivery of interrupts to the ARM

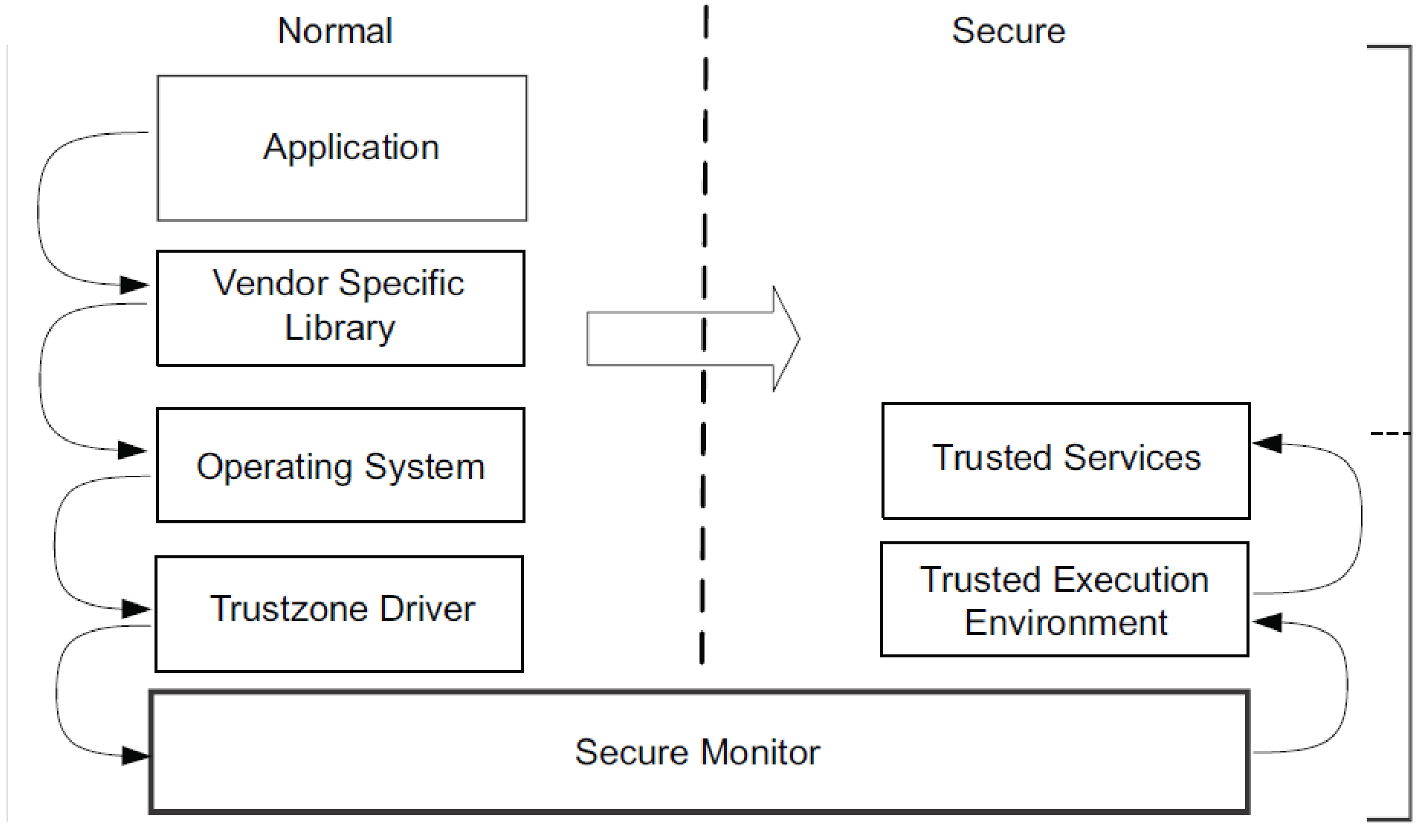


## ARM – power management

- Many ARM systems are mobile devices
  - optimization of power usage is a key design constraint
- Programmers can code for low-power
  - TCO, cooling and environment issues
- ARM power levels
  - Run, Standby, Dormant, Shutdown
- Standby
  - WFI/WFE hints to stall processor and gates the clock
  - enter/exit 2-cycles
- Dormant
  - processor state saved to memory, clock gated and logic switched off
  - enter/exit ~15K cycles
- DVFS

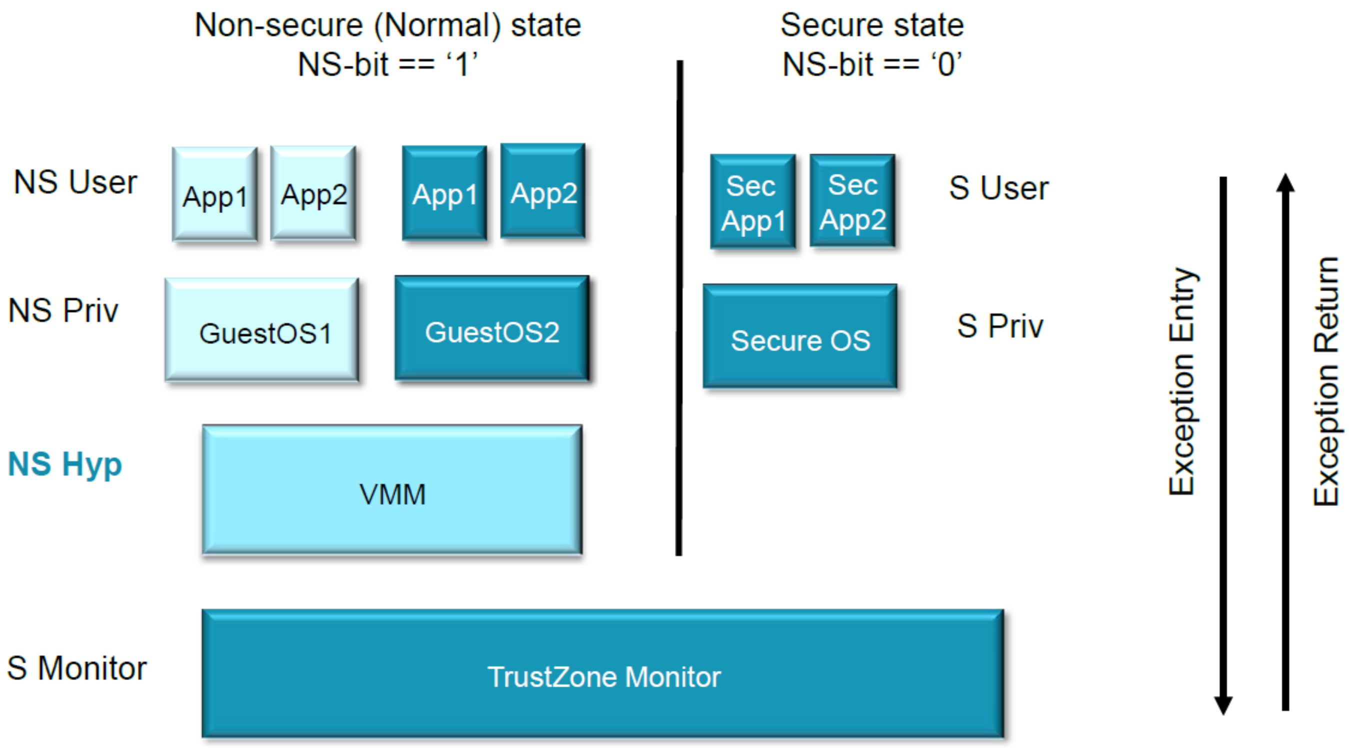
## ARM - Security

- TrustZone
  - division of hw and sw resources
  - restricted access to secure services through MON mode

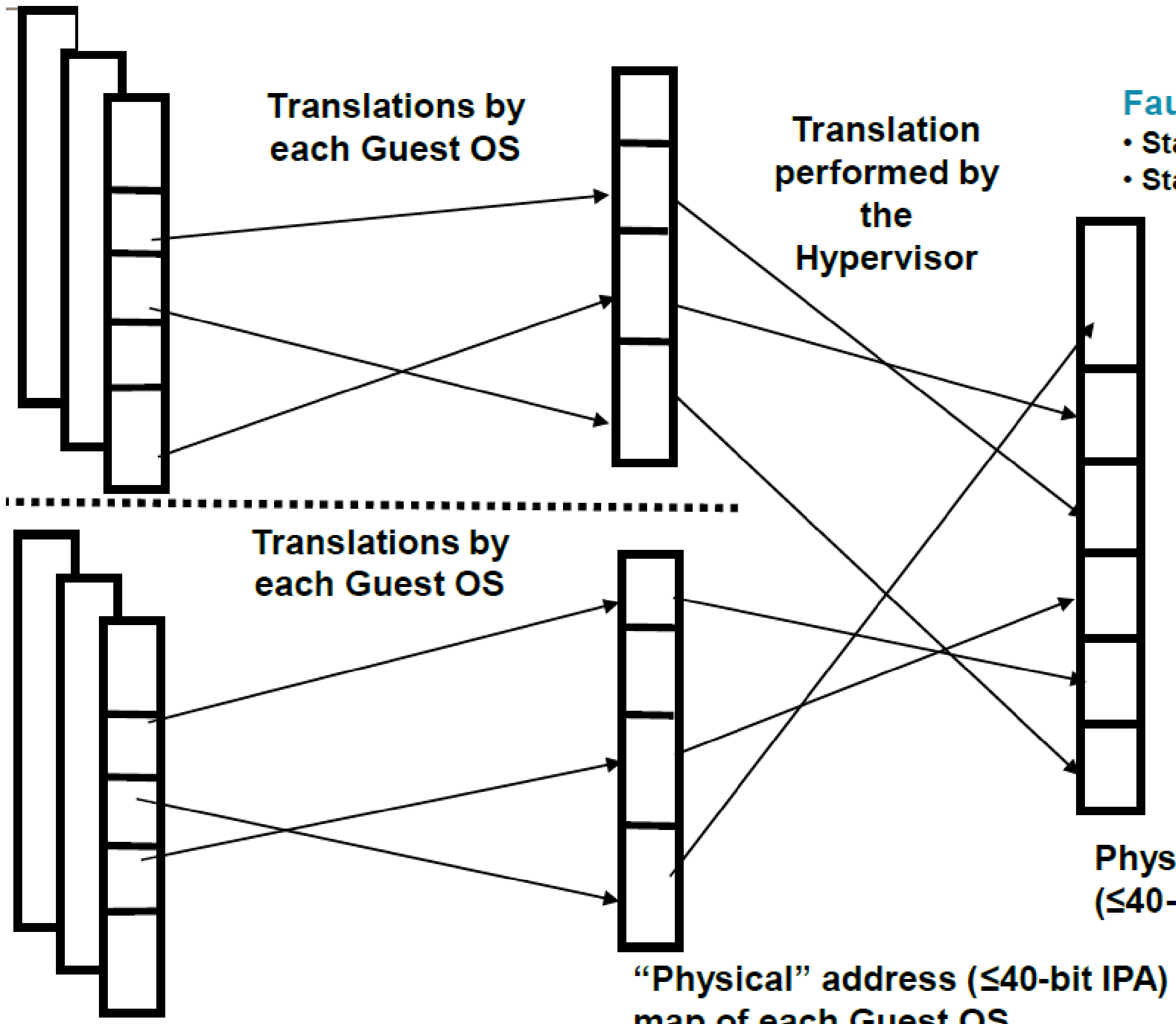


# ARM - Virtualization

- Virtualization extension to v7-A
  - new privilege level for the hypervisor (HYP)
  - 2-stage address translation – OS and hypervisor levels
  - complements the security extensions



# ARM - Virtualization


Translations by each Guest OS
Translation performed by the Hypervisor
Translations by each Guest OS

**Faults disambiguated**
- Stage 1 => OS fault handling
- Stage 2 => Hyp mode

Physical address (≤40-bit PA) map

"Physical" address (≤40-bit IPA) map of each Guest OS

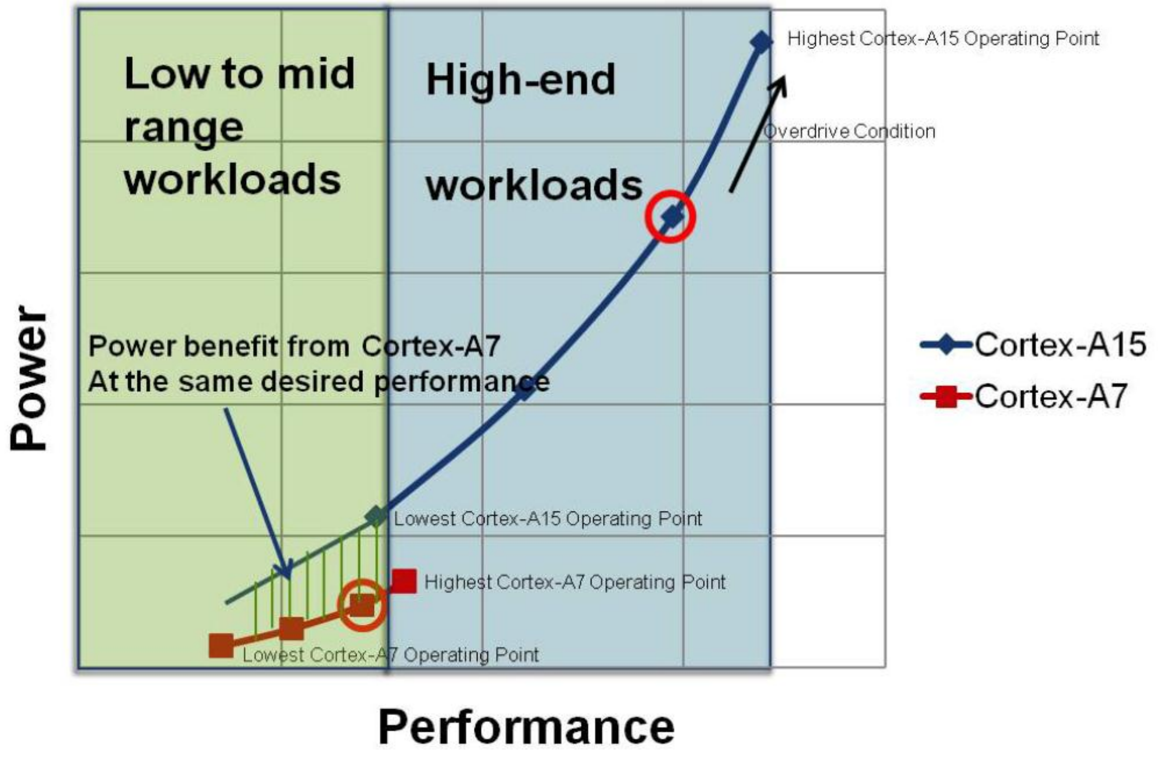
Virtual address (32-bit VA) map of each application on each GuestOS

Implementation note: TLBs can merge Stage1 and Stage 2 translation information (VA => PA)

# ARM – big.LITTLE

- heterogeneous "switched" MP
  - ultra low-power core (Cortex-A7)
  - low-power high performance core (Cortex-A15)

  } **architecturally Identical ***
- switch between cores, maximising energy efficiency



## ARM – OS research

- ARM v7-A is a mature full-featured architecture
- many cheap dev boards
  - don't need to risk bricking your phone/NAS/pda anymore!
  - pandaboard, samsung origen board, ST snowball, Freescale iMx
  - Linux/Android builds available & supported by linaro.org
  - ~$150-200 - within the reach of academic research
- Cortex-A Programmer's Guide
  - great free guide explaining ARM v7-A and OS programming
  - http://bit.ly/CortexAProg
  - (requires email registration)

## End