Linux Support for ARM LPAE

Catalin Marinas

ELC Europe 2011

Agenda

- Introduction
- Classic ARM MMU
- Classic ARM MMU Limitations
- ARM LPAE Features
- ARM LPAE and Virtualisation
- Linux and ARM LPAE
- Current Status and Future Developments

Introduction

- Early ARM systems required only a few MBs of RAM
- More and more complex smart-phones requiring 100s MB or even GBs of RAM
  - 32-bit physical addresses impose a 4GB hard limit
- It’s not all about RAM
  - Peripherals
  - Flash memory
  - System ROM
- (Mobile) virtualisation requires even more RAM
  - But not necessarily at the Guest OS level
- ARM LPAE support for Linux developed within ARM Ltd.
  - First patches posted on LKML – October 2010

Classic ARM MMU

- 32-bit physical address space
- 2-level translation tables
  - Pointed to by TTBR0 (user mappings) and TTBR1 (kernel mappings but with restrictions to the user/kernel memory split)
  - 32-bit page table entries
- 1st level contains 4096 entries (4 pages for PGD)
  - 1MB section per entry or
  - Pointer to a 2nd level table
  - Implementation-defined 16MB supersections
- 2nd level contains 256 entries pointing to 4KB page each
  - 1KB per 2nd level page table
- ARMv6/v7 introduced TEX remapping
  - Memory type becomes a 3-bit index

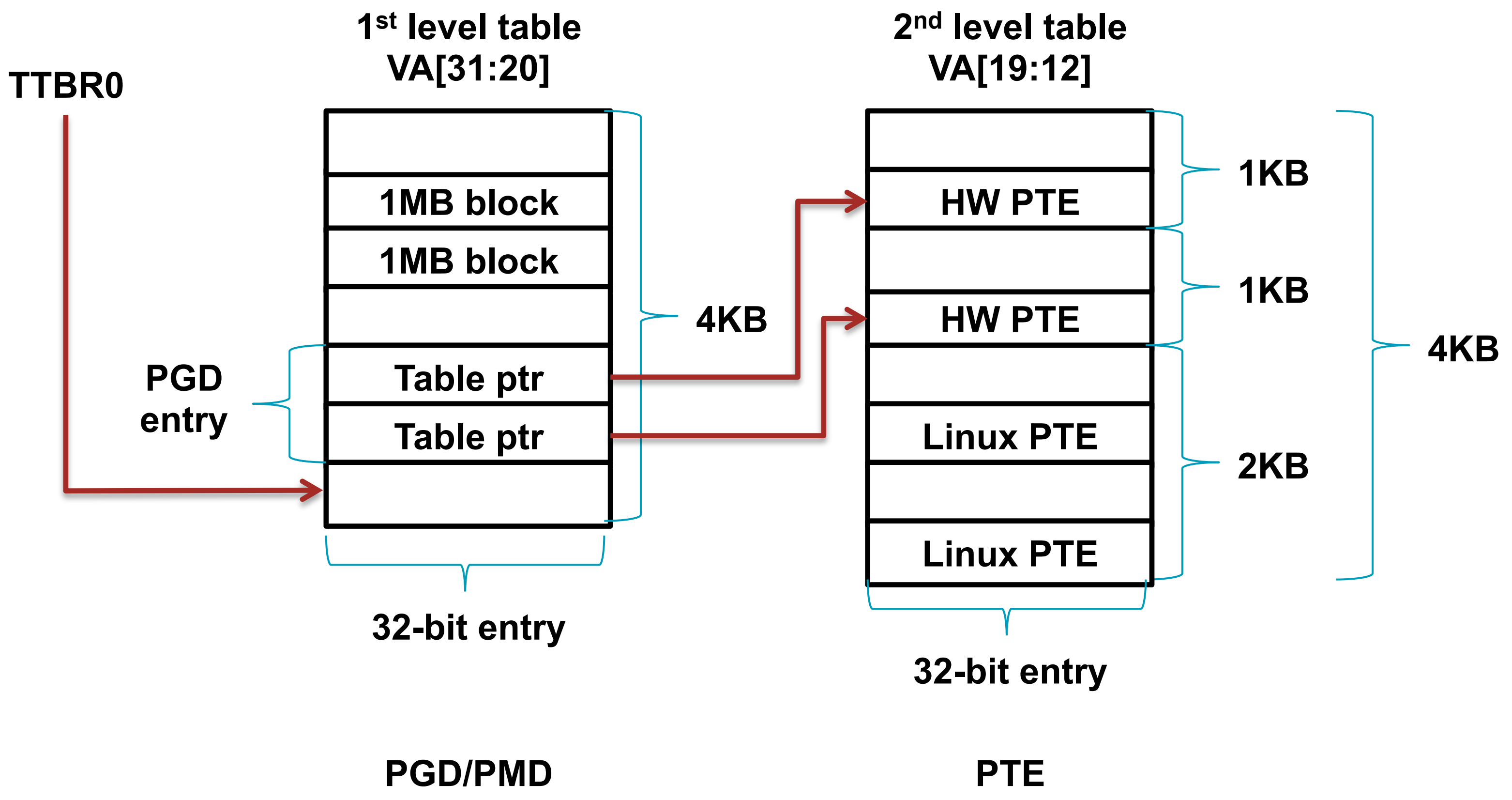
Classic ARM MMU (cont'd)

- Other features
  - XN (eXecute Never) bit
  - Different memory types: Normal (cacheable and non-cacheable), Device, Strongly Ordered
  - Shareability attributes for SMP systems
- ASID-tagged TLB (ARMv6 onwards)
  - Avoids TLB flushing at context switch
  - 8-bit ASID value assigned to an `mm_struct`
  - Dynamically allocated (there can be more than 256 processes)

Classic ARM MMU Limitations

- Only 32-bit physical address space
  - Growing market requiring more than 4GB physical address space (both RAM and peripherals)
  - Supersections can be used to allow up to 40-bit addresses using 16MB sections (implementation-defined feature)
- Prior to ARMv6, not a direct link between access permissions and Linux PTE bits
  - Simplified permission model introduced with ARMv6 but not used by Linux
- 2nd level page table does not fill a full 4K page
- ARM Linux workarounds
  - Separate array for the Linux PTE bits
  - 1st level entry consists of two 32-bit locations pointing to 2KB 2nd level page table entries

Classic ARM MMU Limitations (cont'd)



ARM LPAE Features

- 40-bit physical addresses (1TB)
- 40-bit intermediate physical addresses (guest physical space)
- 3-level translation tables
  - Pointed to by TTBR0 (user mappings) and TTBR1 (kernel mappings)
    - Not as restrictive on user/kernel memory split (can use 3:1)
    - With 1GB kernel mapping, the 1st level is skipped
  - 64-bit entries in each level
- 1st level contains 4 entries (stage 1 translation)
  - 1GB section or
  - Pointer to 2nd level table
- 2nd level contains 512 entries (4KB in total)
  - 2MB section or
  - Pointer to 3rd level

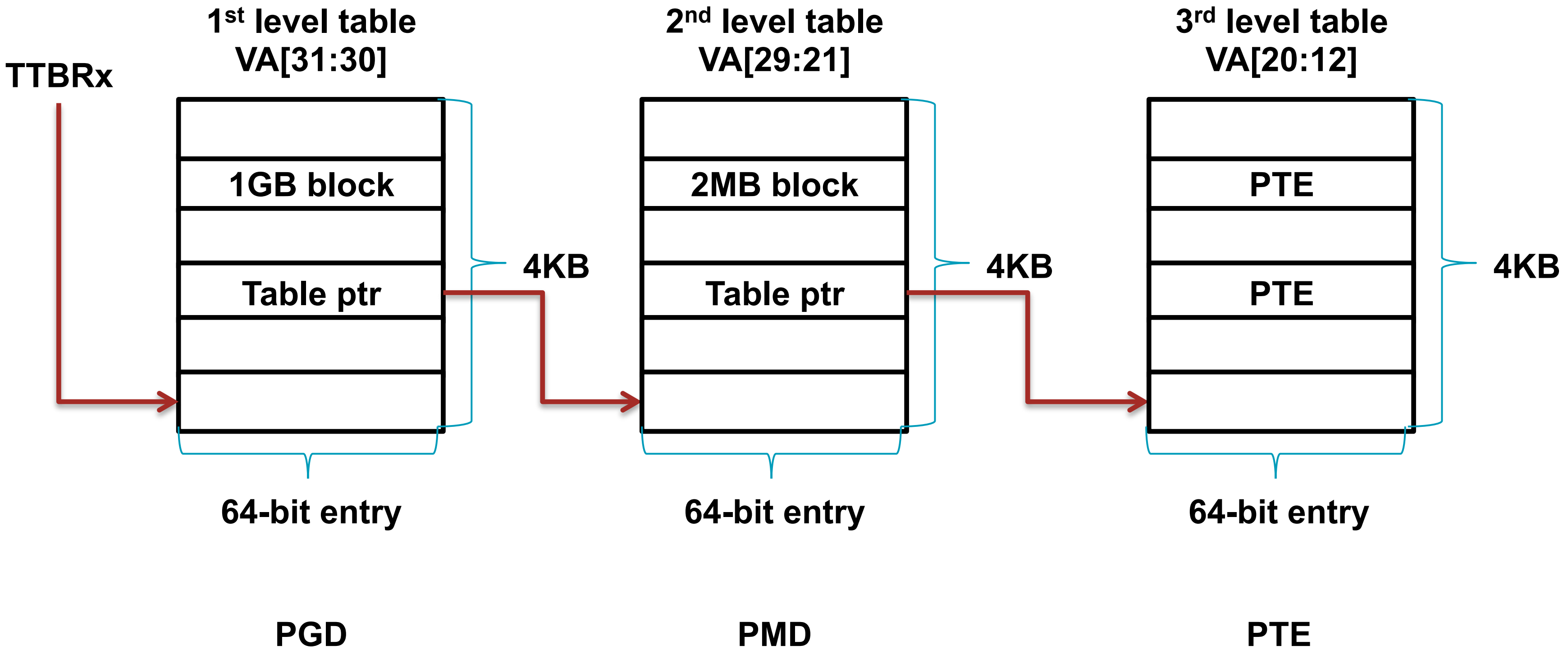
ARM LPAE Features (cont'd)

- 3rd level contains 512 entries (4KB)
  - Each addressing a 4KB range
  - Possibility to set a contiguity flag for 16 consecutive pages
- LDRD/STRD (64-bit load/store) instructions are atomic on ARM processors supporting LPAE
- Only the simplified page permission model is supported
  - No kernel RW and user RO combination
- Domains are no longer present (they have already been removed in ARMv7 Linux)
- Additional spare bits to be used by the OS
- Dedicated bits for user, read-only and access flag (young) settings

ARM LPAE Features (cont'd)

- ASID is part of the TTBR0 register
  - Simpler context switching code (no need to deal with speculative TLB fetching with the wrong ASID)
  - The Context ID register can be used solely for debug/trace
- Additional permission control
  - PXN – Privileged eXecute Never
  - SCTLR.WXN, SCTLR.UWXN – prevent execution from writable locations (the latter only for user accesses)
  - APTable – restrict permissions in subsequent page table levels
  - XNTable, PXNTable – override XN and PXN bits in subsequent page table levels
- New registers for the memory region attributes
  - MAIR0, MAIR1 – 32-bit Memory Attribute Indirection Registers
  - 8 memory types can be configured at a time

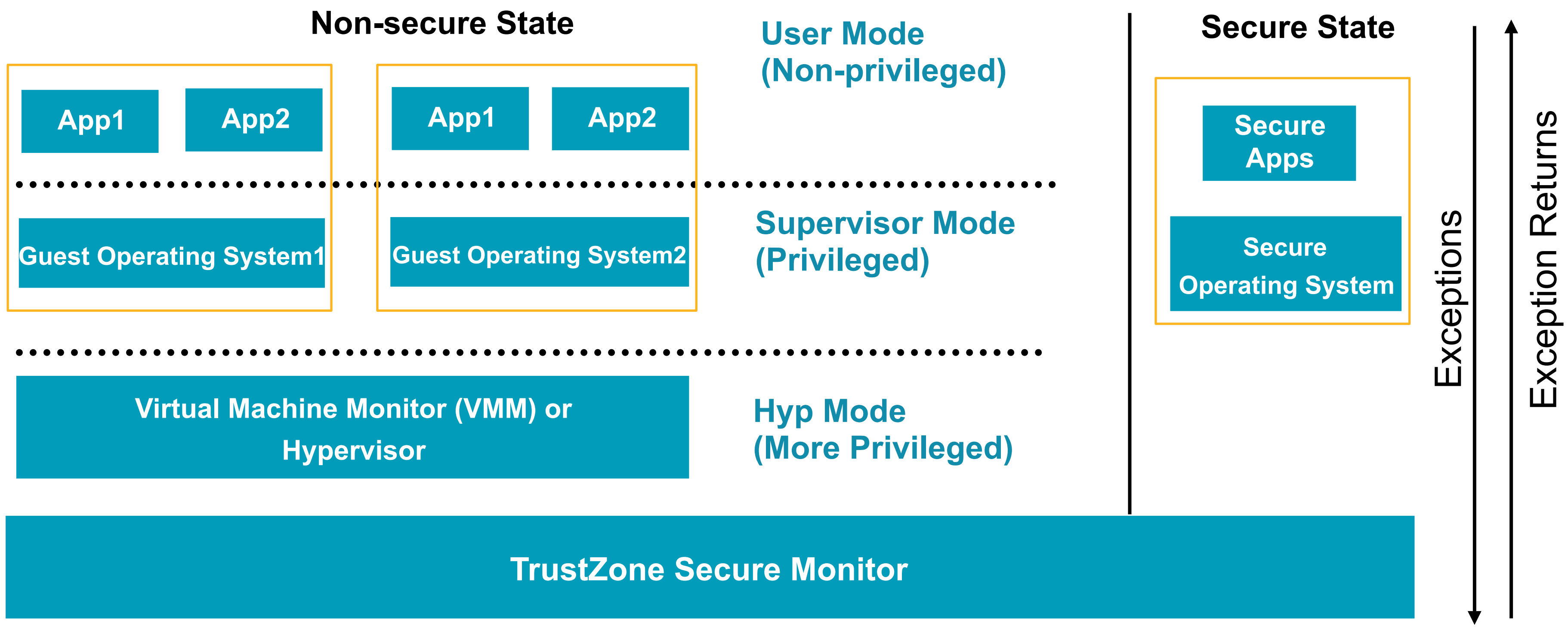
ARM LPAE Features (cont'd)



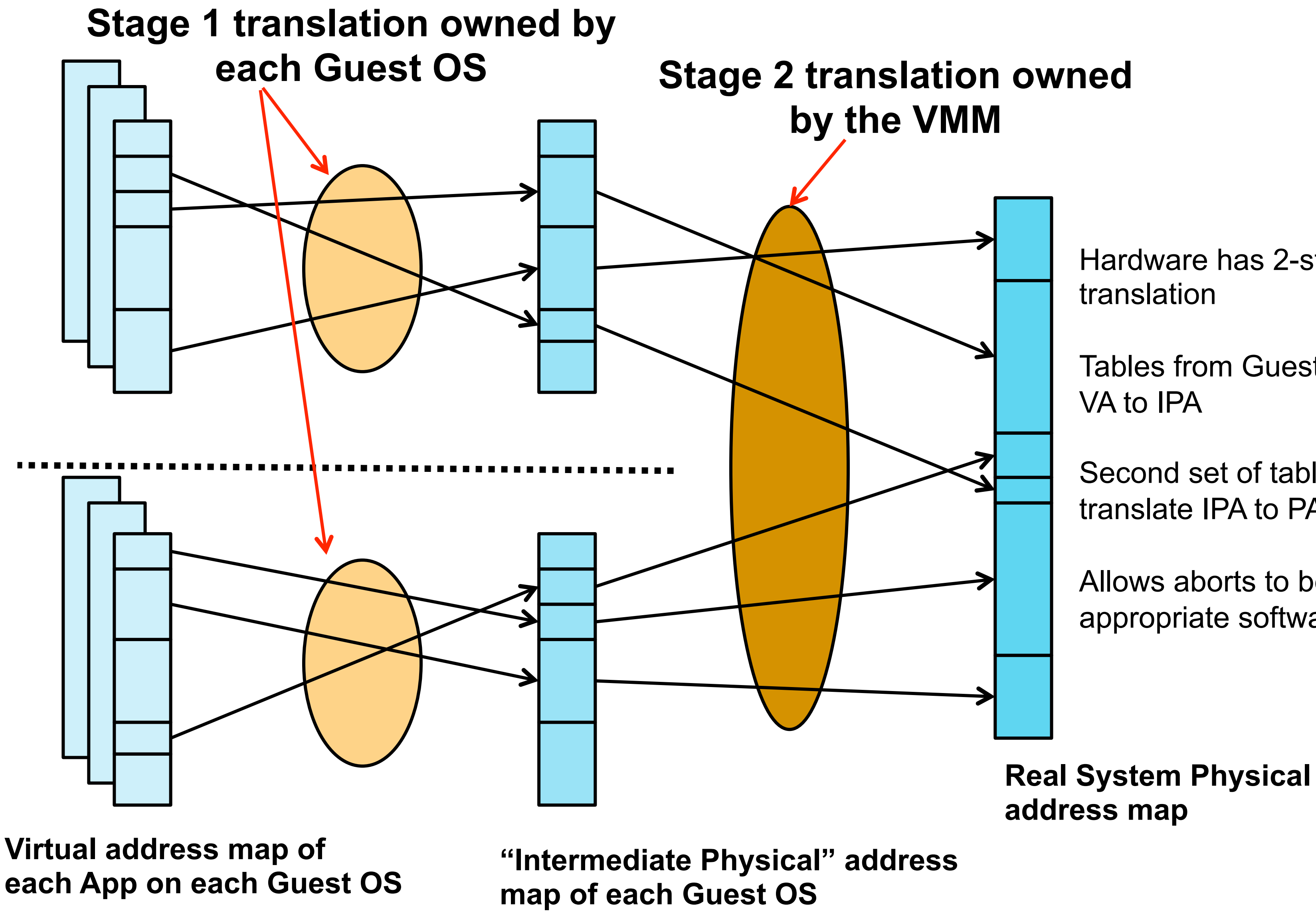
ARM LPAE and Virtualisation

- Guest OS running at the same privilege as on earlier processors
- New higher privileged Hypervisor mode
  - Controls a wide range of OS accesses to the hardware: memory, timers, interrupt controller
- The same page table format can be used as stage 2 translations
  - Converts intermediate physical address (IPA) to the physical address
- Guest memory attributes can be overridden by the Hypervisor
- Stage 2 translation requires setup in the Hypervisor mode

ARM LPAE and Virtualisation (cont'd)



ARM LPAE and Virtualisation (cont'd)


Stage 1 translation owned by each Guest OS
Stage 2 translation owned by the VMM
Real System Physical address map
Virtual address map of each App on each Guest OS
"Intermediate Physical" address map of each Guest OS

Hardware has 2-stage memory translation

Tables from Guest OS translate VA to IPA

Second set of tables from VMM translate IPA to PA

Allows aborts to be routed to appropriate software layer

Linux and ARM LPAE

- Linux + ARM LPAE has the same memory layout as the classic MMU implementation
  - Described in Documentation/arm/memory.txt
  - `0..TASK_SIZE` – user space
  - `PAGE_OFFSET-16M..PAGE_OFFSET-2M` – module space
  - `PAGE_OFFSET-2M..PAGE_OFFSET` – highmem mappings
- Highmem is already supported by the classic MMU
  - Memory beyond 4G is only accessible via highmem
  - Page mapping functions use `pfn` (32-bit variable, `PAGE_SHIFT == 12`, maximum 44-bit physical addresses)
- Original ARM kernel port assumed 32-bit physical addresses
  - LPAE redefines `phys_addr_t`, `dma_addr_t` as `u64` (generic typedefs)
  - New `ATAG_MEM64` defined for specifying the 64-bit memory layout

Linux and ARM LPAE (cont'd)

- Hard-coded assumptions about 2 levels of page tables
  - `PGDIR_(SHIFT|SIZE|MASK)` references converted to `PMD_*`
- `swapper_pg_dir` extended to cover both 1st and 2nd levels of page tables
  - 1 page for PGD (only 4 entries used for stage 1 translations)
  - 4 pages for PMD
  - `init_mm.pgd` points to `swapper_pg_dir`
- TTBR0 used for user mappings and always points to PGD
- TTBR1 used for kernel mapping:
  - 3:1 split – TTBR1 points to 4th page of PMD (2 levels only, 1GB)
    - Classic MMU does not allow the use of TTBR1 for the 3:1 split
  - 2:2 split – TTBR1 points to 3rd PGD entry
  - 1:3 split – TTBR1 points to 1st PGD entry

Linux and ARM LPAE (cont'd)

- The page table definitions have been separated into pgtable*-2level.h and pgtable*-3level.h files
  - Few PTE bits shared between classic and LPAE definitions
  - Negated definitions of `L_PTE_EXEC` and `L_PTE_WRITE` to match the corresponding hardware bits
  - Memory types are the same and they represent an index in the TEX remapping registers (PRRR/NMRR or MAIR0/MAIR1)
- The proc-v7.S file has been duplicated into proc-v7lpae.S
  - Different register setup for TTBRx and TEX remapping (MAIRx)
  - Simpler `cpu_v7_set_pte_ext` (1:1 mapping between hardware and software PTE bits)
  - Simpler `cpu_v7_switch_mm` (ASID switched with TTBR0)
- Current ARM code converted to pgtable-nopud.h
  - Not using 'nopmd' with classic MMU

Linux and ARM LPAE (cont'd)

- Lowmem is mapped using 2MB sections in the 2nd level table
  - PGD and PMD only allocated from lowmem
  - PTE tables can be allocated from highmem
- Exception handling
  - Different IFSR/DFSR registers structure and exception numbering – arch/arm/mm/fault.c modified accordingly
  - Error reporting includes PMD information as well
- PGD allocation/freeing
  - Kernel PGD entries copied to the new PGD during `pgd_alloc()`
  - Modules and pkmap entries added to the PMD during fault handling
- Identity mapping (PA == VA)
  - Required for enabling or disabling the MMU – secondary CPU booting, CPU hotplug, power management, kexec

Linux and ARM LPAE (cont'd)

- Uses `pgd_alloc()` and `pgd_free()`
- When `PHYS_OFFSET > PAGE_OFFSET`, kernel PGD entries may be overridden
- `swapper_pg_dir` entries marked with an additional bit – `L_PGD_SWAPPER`
  - `pgd_free()` skips such entries during clean-up

Current Status and Future

- Initial development done on software models
  - Tested on real hardware (FPGA and silicon)
- Parts of the LPAE patch set already in mainline
  - Mainly preparatory patches, not core functionality
  - Aiming for full support in Linux 3.3
- Hardware supporting LPAE
  - ARM Cortex-A15, Cortex-A7 processors
  - TI OMAP5 (dual-core ARM Cortex-A15)
- Other developments
  - KVM support for Cortex-A15 – implemented by Christoffer Dall at Columbia University
    - Uses the Virtualisation extensions together with the LPAE stage 2 translations

Reference

- ARM Architecture Reference Manual rev C
  - Currently beta, not publicly available yet
- Specifications publicly available on ARM Infocenter
  - http://infocenter.arm.com/
  - ARM architecture -> Reference Manuals -> ARMv7-AR LPA Virtualisation Extensions
- Linux patches – ARM architecture development tree
  - Hosts the latest ARM architecture developments before patches are merged into the mainline kernel
  - git://github.com/cmarinas/linux.git
  - When the kernel.org accounts are back
    - git://git.kernel.org/pub/scm/linux/cmarinas/linux-arm-arch.git

Questions