LINUXCONFAU
ONLINE 2021

Large Virtual Address support (52-bit) in ARM64 kernel

SPEAKER

Bhupesh Sharma

Red Hat

<bhupesh.linux@gmail.com>

Twitter: @bhupesh_sharma

$whoami

- Part of Red Hat kernel team.
- Been hacking on boot loaders & kernel since past 15 years.
- Contribute to:
  - Linux,
  - EFI/u-boot bootloader, and
  - User-space utilities like:
    - kexec-tools, and
    - makedumpfile.
- Co-maintain crash-utility tool

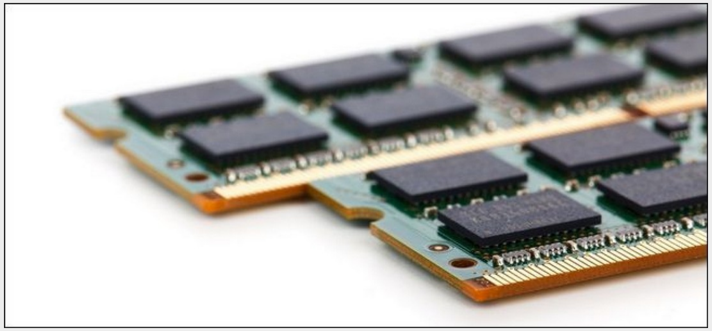
Outline

- Large VA support for arm64
  - What?
  - How?
- 52-bit VA kernel support - arm64
- Flipping the arm64 kernel memory layout
- Impact on user-space applications
- 52-bit userspace VAs
- Next Steps

Large VA support for arm64 – What?

- 64-bit hardware ⇨ can address **very large** address space.
  - Upto 16 EiBs (16 × 10246 = 264 = 18,446,744,073,709,551,616 bytes)
  - Approx 18.4 exabytes of memory.

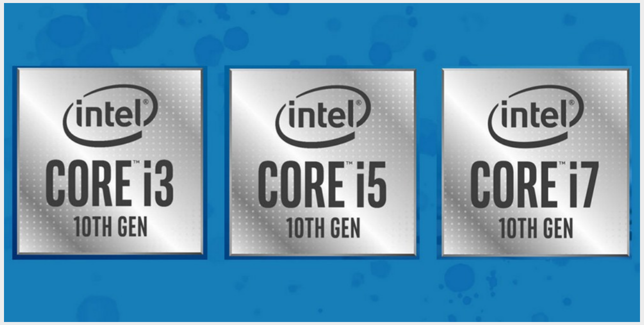
- Servers available with ⇨64 TiB (& upwards) of memory.
  - ↗ Use-cases ⇨ requiring⇨ addressing spaces > 2^48 bytes.
- Limitations
  - Not all instruction sets, and
  - Not all processors,
    - support a full 64-bit virtual or physical address space.



Large VA support for arm64 – What?

- x86_64
  - Supports 5-level page tables in both Hardware & Software.
  - Allows addressing address space = 2^57 bytes.
  - Bumps limits to
    - → 128 PiB of virtual address space,
    - → 4 PiB of physical address space.
- arm64
  - Introduces 2 new architecture extensions
    - 52-bit addressing extensions
      - ARMv8.2 LVA, and
      - ARMv8.2 LPA
  - Allows addressing
    - → 4 PiB of virtual address space,
    - → 4 PiB of physical address space.

Reference:



Large VA support for arm64 – How?

Hardware Support

- ARMv8.2 LVA
  - Supports larger VA space
  - Each translation table base register of up to 52 bits
    - when using the 64KB translation granule.
- ARMv8.2 LPA
  - Allows larger intermediate physical address (IPA), and
  - PA space of up to 52 bits when using the 64KB translation granule.
  - Allows a level 1 block size where the block covers a 4TB address range for the 64KB translation granule if the implementation support 52 bits of PA.

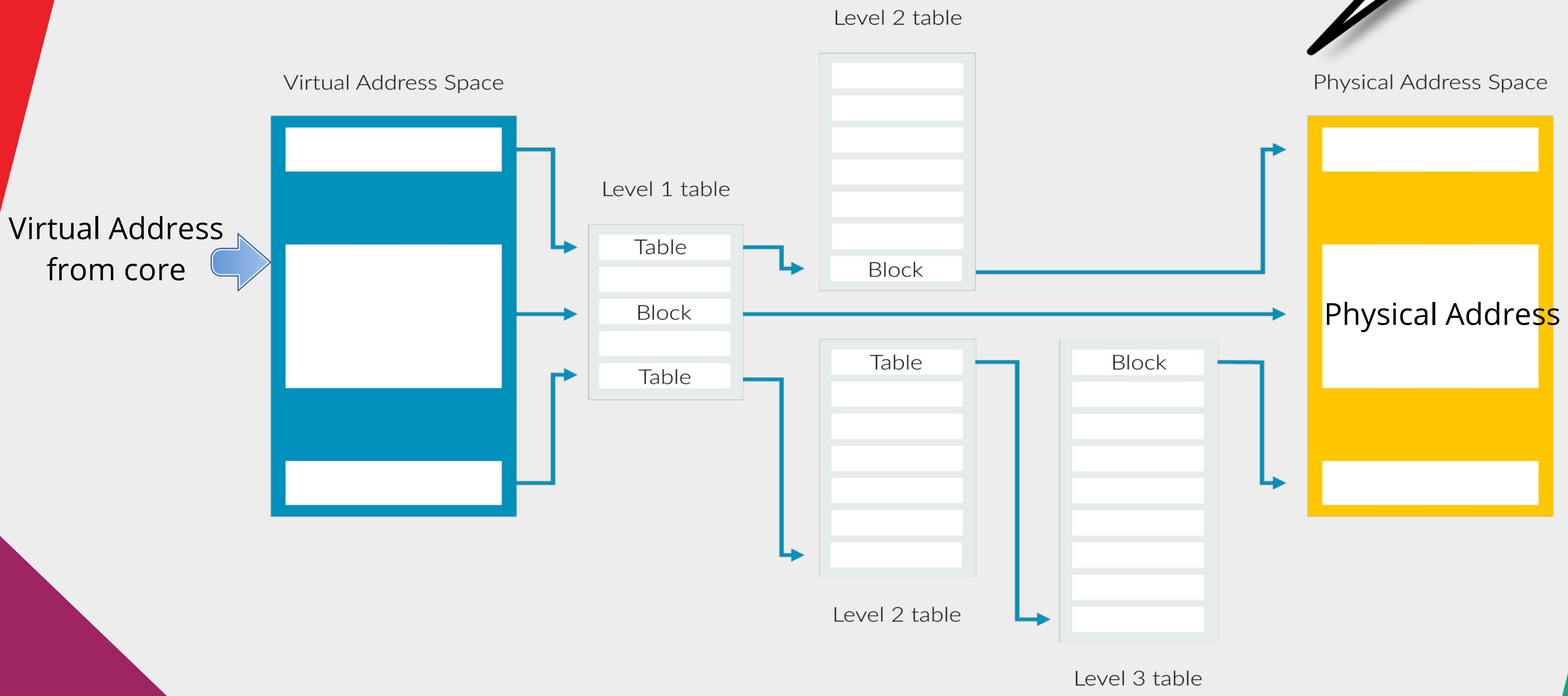
  NOTE: These features are supported in AArch64 state only.
- Cortex-A processors with ARMv8.2 extension support:
  - Cortex A55
  - Cortex A75
  - Cortex A76

Reference: ARMv-8 Architecture Reference Manual from ARM

Large VA support for arm64 – How?

- Translating a virtual address to a physical address

Hardware Support



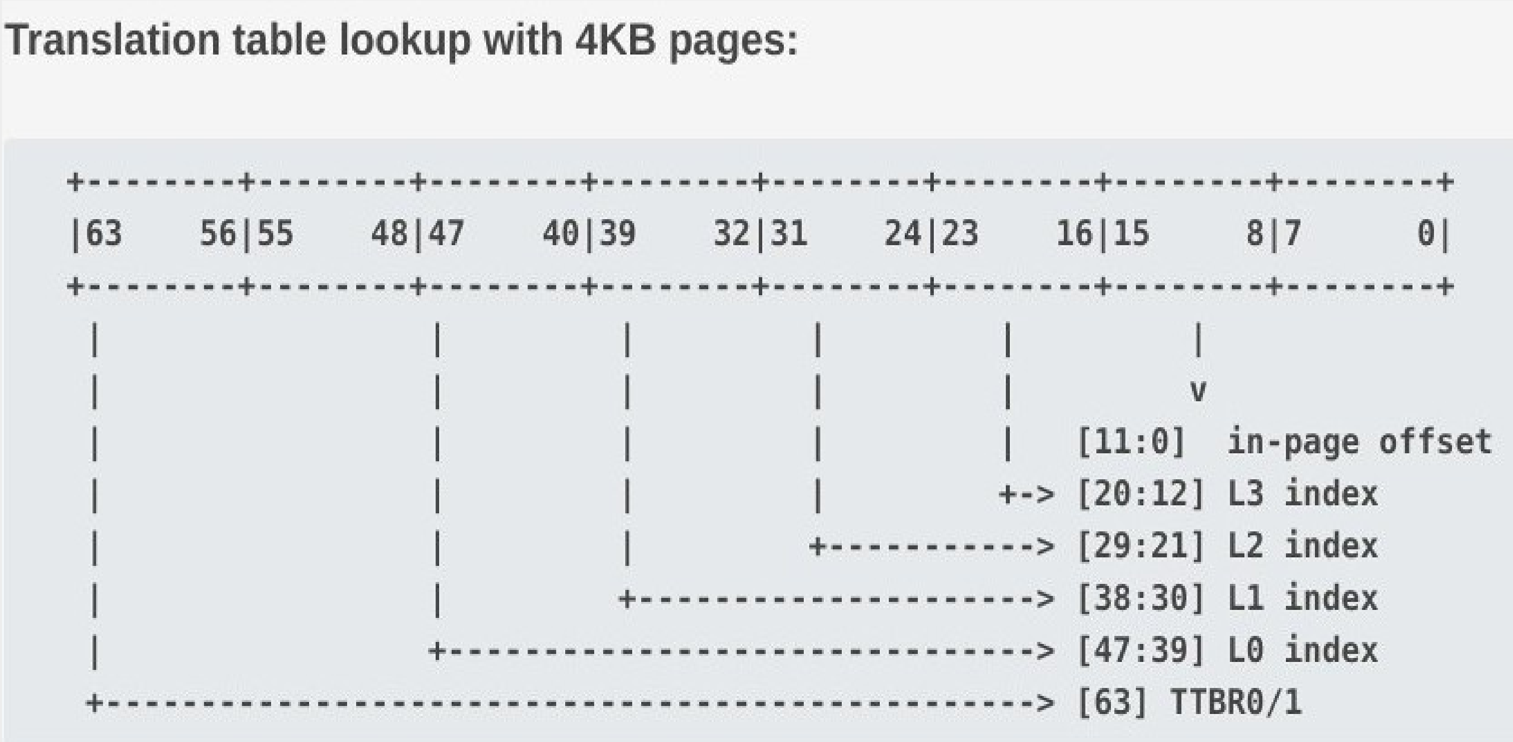
Reference: Memory management guide from ARM

Large VA support for arm64 – How?

- arm64 Linux uses:
  - 4KB page size
    - 39-bit (512GB) virtual addresses ⇨ 3 translation tables levels
    - 48-bit (256TB) virtual addresses ⇨ 4 translation tables levels.

Software Support

Reference:
Documentation/arm64/memory.rst



Translation table lookup with 4KB pages:

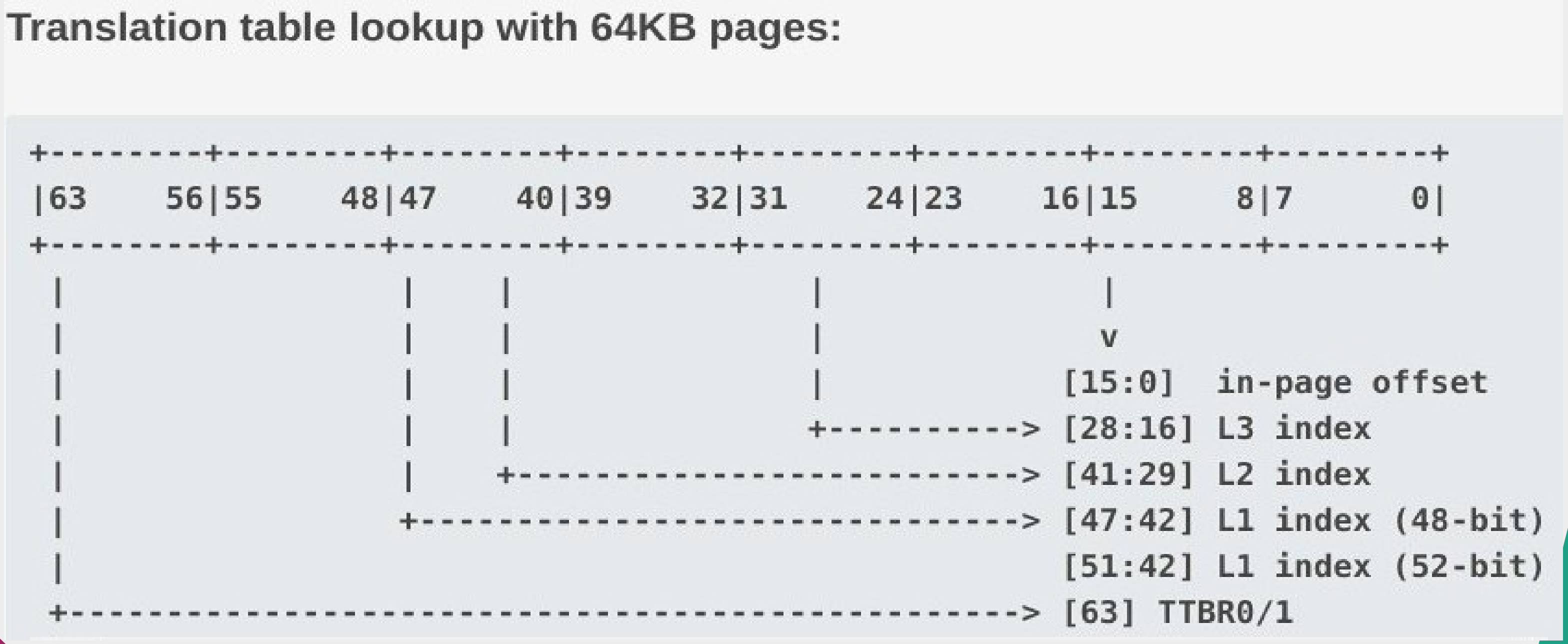
```
+--------+--------+--------+--------+--------+--------+--------+--------+
|63    56|55    48|47    40|39    32|31    24|23    16|15     8|7      0|
+--------+--------+--------+--------+--------+--------+--------+--------+
 |                 |         |         |         |         |
 |                 |         |         |         |         v
 |                 |         |         |         |   [11:0]  in-page offset
 |                 |         |         |         +-> [20:12] L3 index
 |                 |         |         +-----------> [29:21] L2 index
 |                 |         +---------------------> [38:30] L1 index
 |                 +-------------------------------> [47:39] L0 index
 +-------------------------------------------------> [63] TTBR0/1
```

Large VA support for arm64 – How?

- arm64 Linux uses:
  - 64KB page size
    - 42-bit (4TB) virtual addresses ⇨ 2 translation tables levels,
      - but the memory layout is the same.
    - 48-bit (256TB) virtual addresses ⇨ 3 translation tables levels
    - 52-bit (4PiB) virtual addresses ⇨ 3 translation tables levels with ARMv8.2 extension
      - expands number of descriptors in the first level of translation.

Software Support

Reference: Documentation/arm64/memory.rst

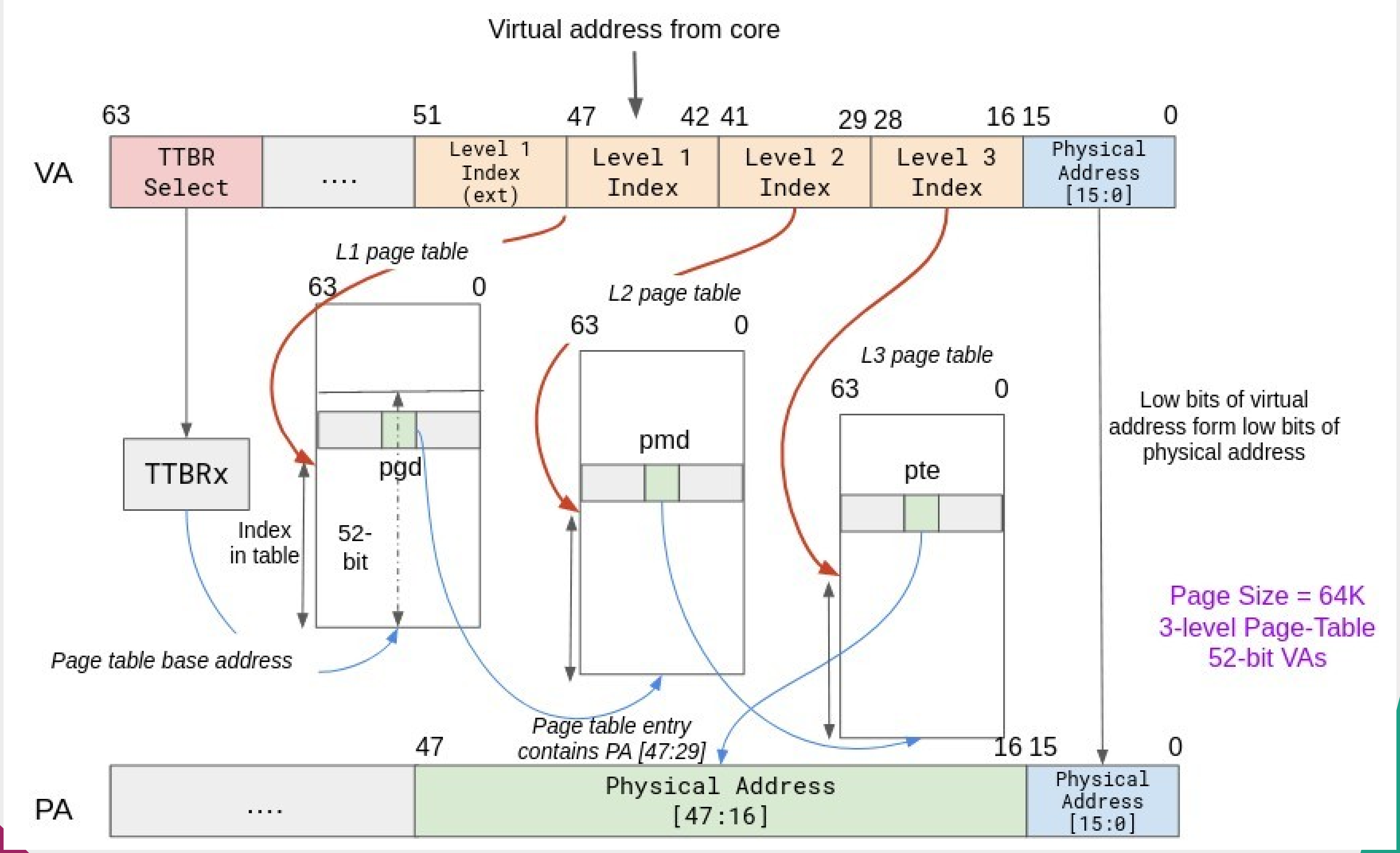


Translation table lookup with 64KB pages:

```
+--------+--------+--------+--------+--------+--------+--------+--------+
|63    56|55    48|47    40|39    32|31    24|23    16|15     8|7      0|
+--------+--------+--------+--------+--------+--------+--------+--------+
 |                 |    |               |              |
 |                 |    |               |              v
 |                 |    |               |            [15:0]  in-page offset
 |                 |    |               +----------> [28:16] L3 index
 |                 |    +--------------------------> [41:29] L2 index
 |                 +-------------------------------> [47:42] L1 index (48-bit)
 |                                                   [51:42] L1 index (52-bit)
 +-------------------------------------------------> [63] TTBR0/1
```

Large VA support for arm64 – How?

- A sample arm64 translation table walk



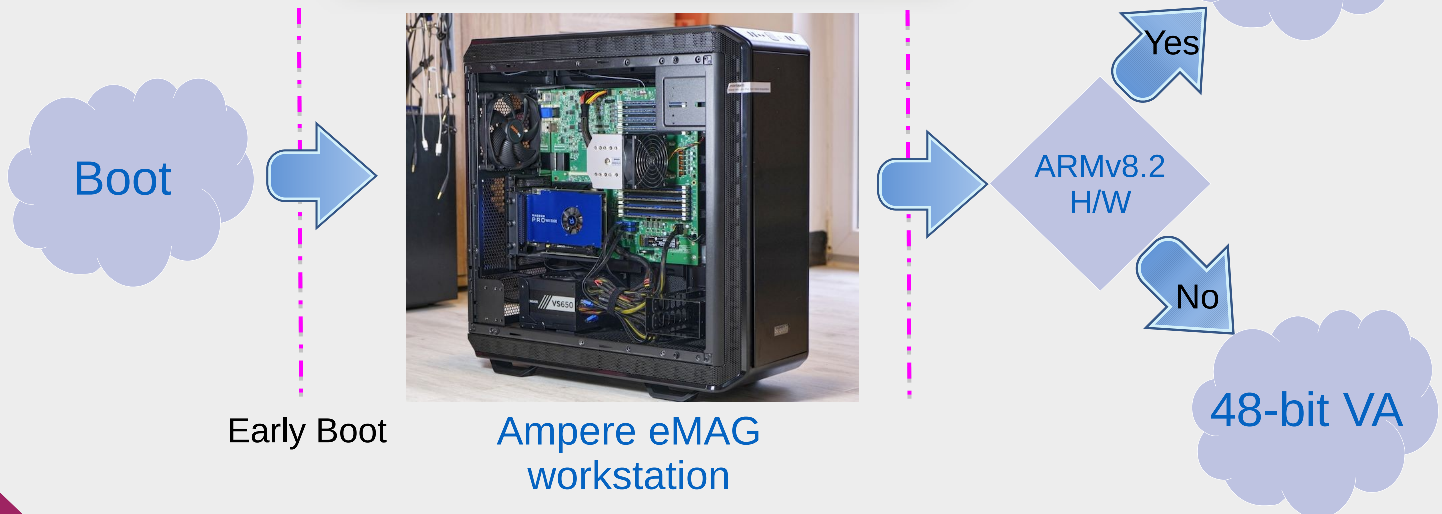
Page Size = 64K
3-level Page-Table
52-bit VAs

Reference: ARMv8-A Address Translation from ARM

52-bit VA kernel support - arm64

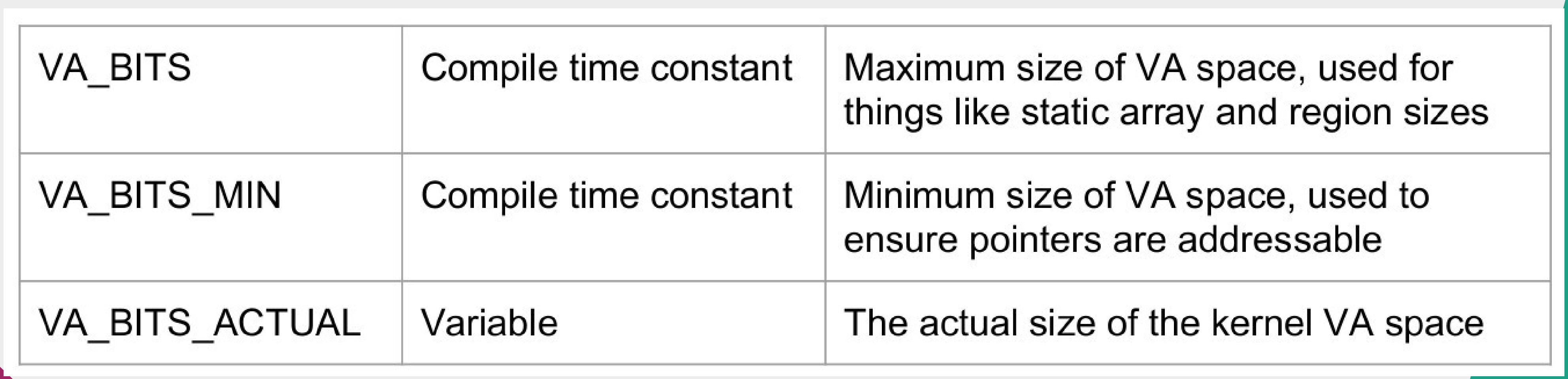
- Design problem from a software support p-o-v
  - Older arm64 CPUs which don’t support ARMv8.2 extensions.
  - New / Upcoming arm64 CPUs which support ARMv8.2 extensions.
- Selected design approach
  - Have a single kernel binary
    - At *early boot time* check if the ARMv8.2 hardware feature is present or not.

Reference: Documentation/arm64/memory.rst



52-bit VA kernel support - arm64

- Single kernel binary for both 48-bit and 52-bit VA spaces.
- VMEMMAP constraints
  - must be sized large enough for 52-bit VAs, and
  - must be sized large enough to accommodate a fixed PAGE_OFFSET.
- VA bits related variables used by kernel code:

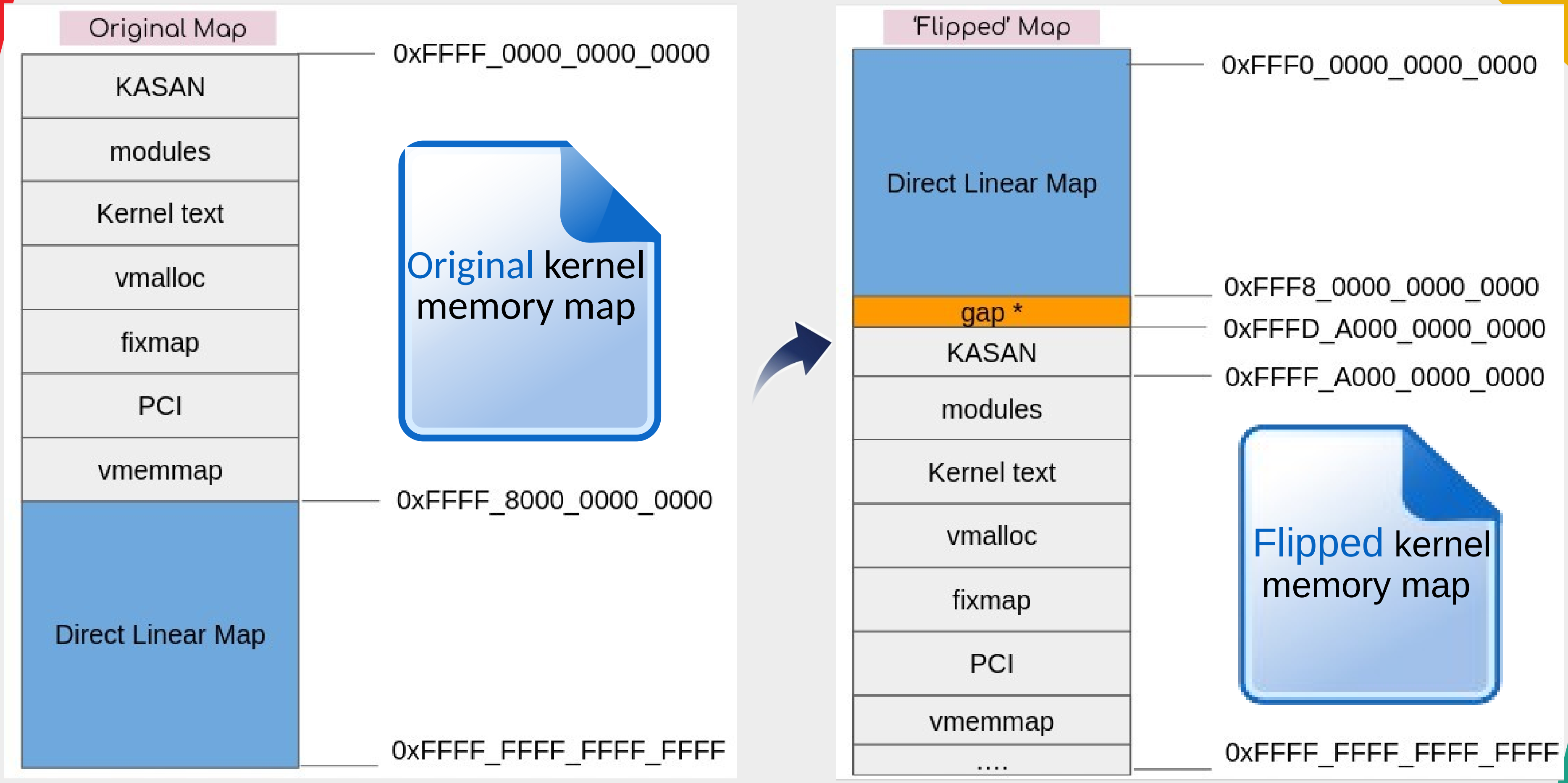


VA_BITS	Compile time constant	Maximum size of VA space, used for things like static array and region sizes
VA_BITS_MIN	Compile time constant	Minimum size of VA space, used to ensure pointers are addressable
VA_BITS_ACTUAL	Variable	The actual size of the kernel VA space

* vabits_actual

Reference: Documentation/arm64/memory.rst

Flipping the kernel memory layout - arm64



Original Map

0xFFFF_0000_0000_0000

KASAN

modules

Kernel text

vmalloc

fixmap

PCI

vmemmap

0xFFFF_8000_0000_0000

Direct Linear Map

0xFFFF_FFFF_FFFF_FFFF

Original kernel memory map

'Flipped' Map

0xFFF0_0000_0000_0000

Direct Linear Map

0xFFF8_0000_0000_0000

gap *

0xFFFD_A000_0000_0000

KASAN

0xFFFF_A000_0000_0000

modules

Kernel text

vmalloc

fixmap

PCI

vmemmap

....

0xFFFF_FFFF_FFFF_FFFF

Flipped kernel memory map

Direct Linear Map is in Higher Half of the VA space.

Reference: Documentation/arm64/memory.rst

- kernel text addresses is kept constant, even for 48 to 52-bits migration.
- We need to flip the VA space.

Impact on user-space applications - arm64

- User-space applications impacted due to flipped kernel memory layout
  - used to debug running / live kernels,
  - to analyze vmcore dumps.
    - for example: kexec-tools, makedumpfile & crash-utility.
- Debugging applications need to perform a va_to_pa() conversion
  - Walk the translation table(s) for determining the physical address
- These applications are broken upstream currently.
- Have proposed fixes for affected applications
  - some have been accepted upstream,
  - others are still pending
    - makedumpfile fix, kexec-tools fix

52-bit userspace VA - arm64

-

⇨ What happens to *other* existing applications?
- To maintain backward compatibility
  - kernel will, *by default*, return virtual addresses to userspace from a 48-bit range.
- Opt-in model for willing user-space application(s)
  - Hint parameter is passed to mmap() calls to receive addresses in 52-bit range.

```
maybe_high_address = mmap(~0UL, size, prot, flags,...);
```

- How to build kernel which returns addresses in 52-bit range?
  - Enable CONFIG options:
    - CONFIG_EXPERT && CONFIG_ARM64_FORCE_52BIT

NOTE: Only intended for debugging + should **not** be used in production.

Next Steps

- Fix broken userspace applications – *WIP*.
- Create awareness about the flipped kernel address map.
- JIT and other applications need to made aware about the mmap() hint parameter and how they can receive addresses in 52-bit range.
- While I am recording this talk, Ard Biesheuvel has posted a patchset to extend the Linear range for 52-bit configurations:
  - https://lore.kernel.org/linux-arm-kernel/20201008153602.9467-3-ardb@kernel.org/
- Test upstream kernel on both old CPUs (48-bit VA) and new CPUs with 52-bit VA)
  - In absence of a real 52-bit HW, you can use ARMv8 fast model simulator for some quick checks.

Slides can be found on github

Email: <bhupesh.linux@gmail.com>
Twitter: @bhupesh_sharma

Questions?