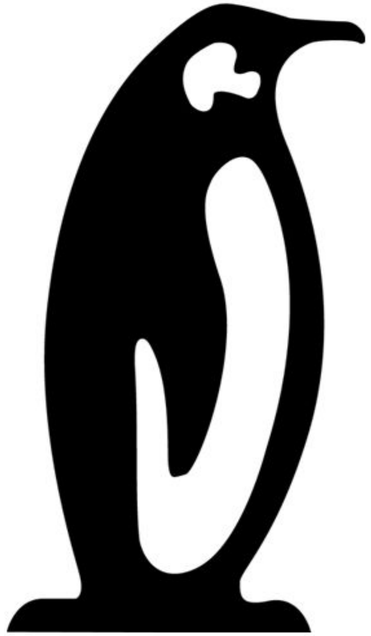
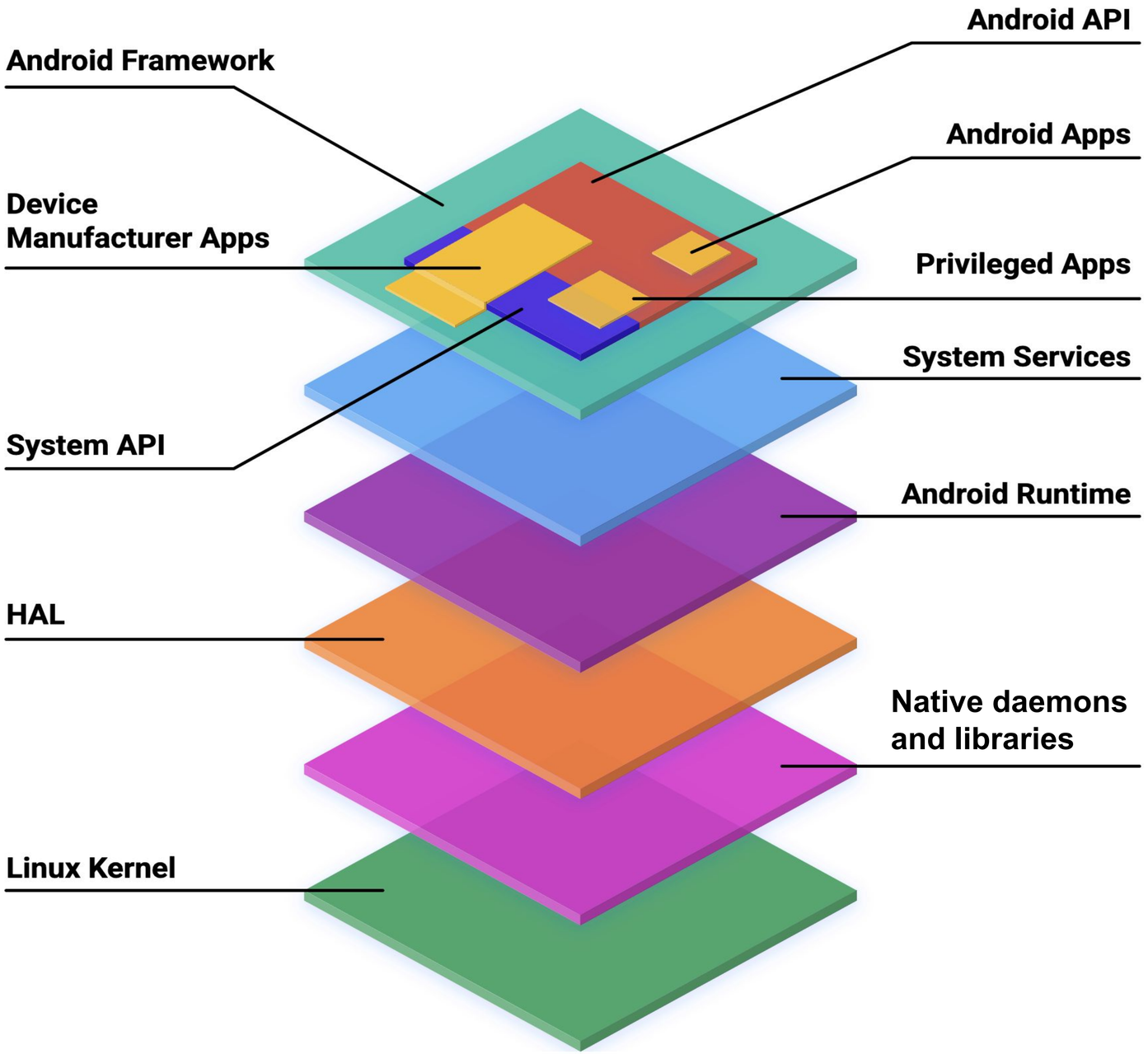


Bring up devices with 16kb support

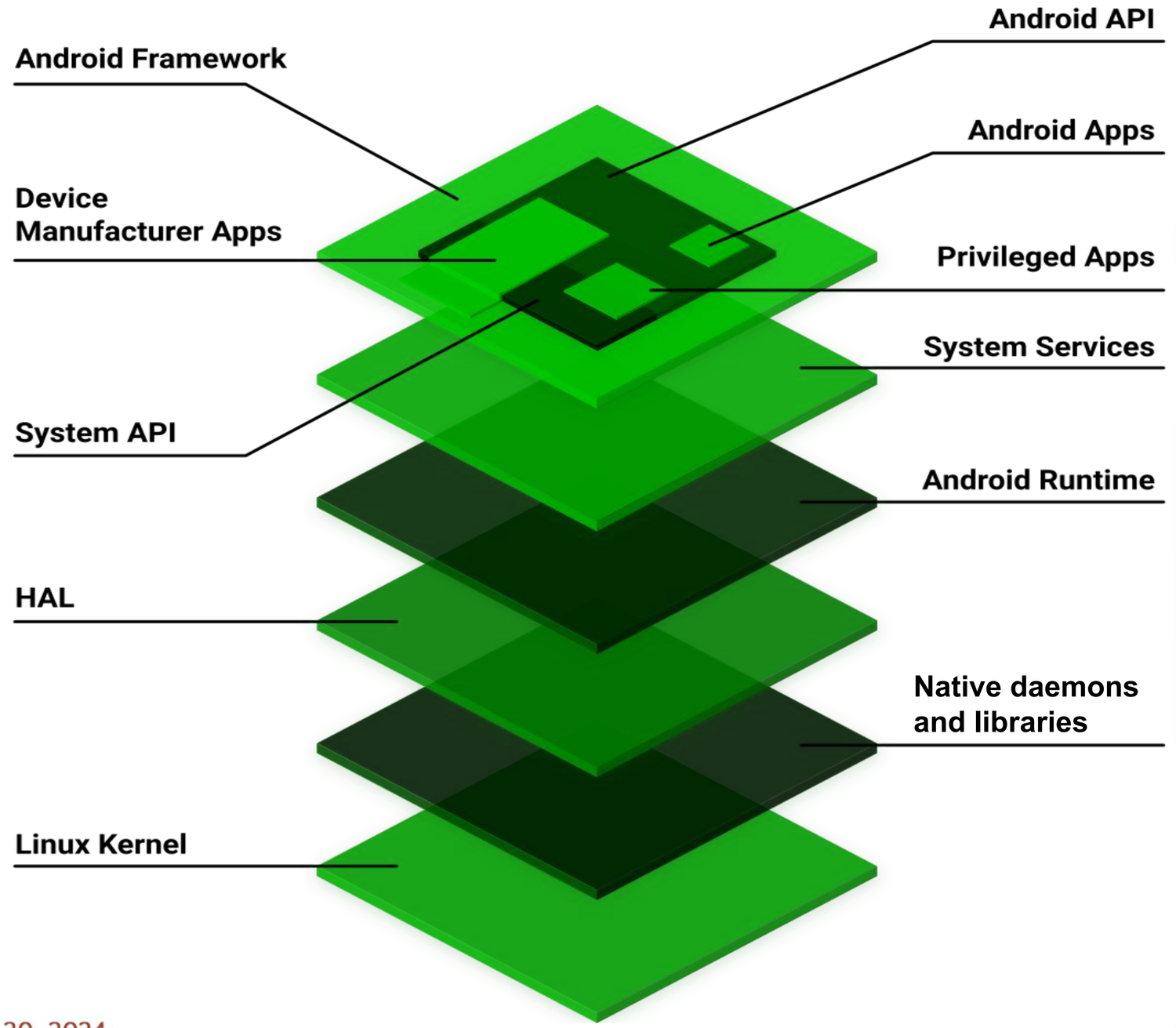
Juan Yescas & Kalesh Singh
Google

LINUX PLUMBERS CONFERENCE Vienna, Austria / Sept. 18-20, 2024

AOSP Architecture



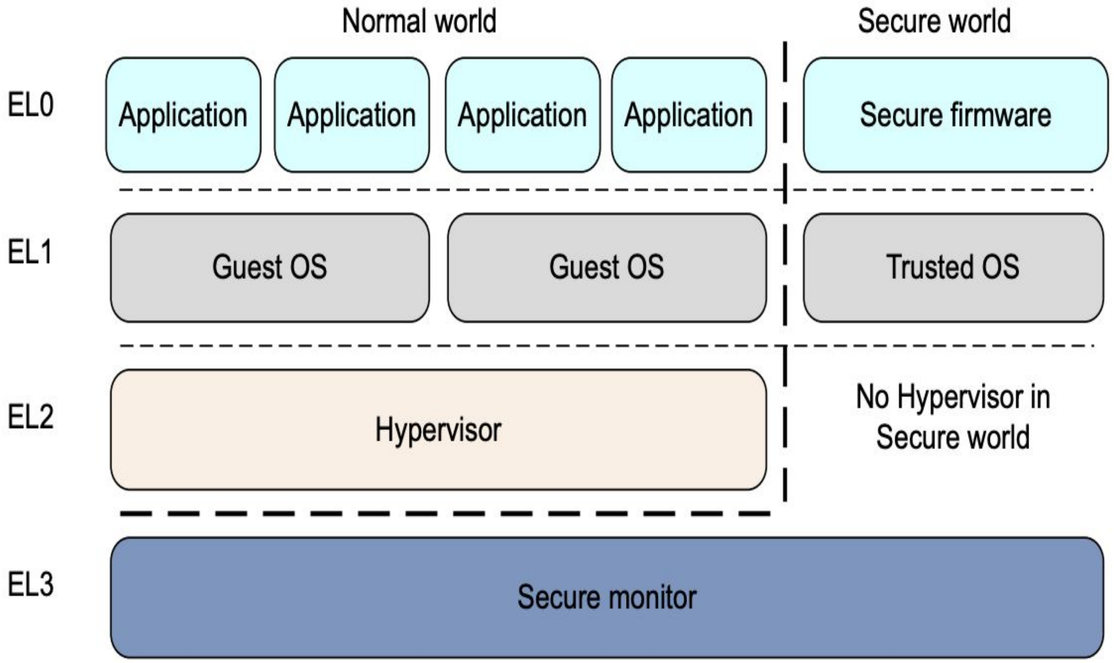
Only the layers
in green
need to change



And Bootloader

Why all the layers are affected?

ARM 64 Architecture Overview



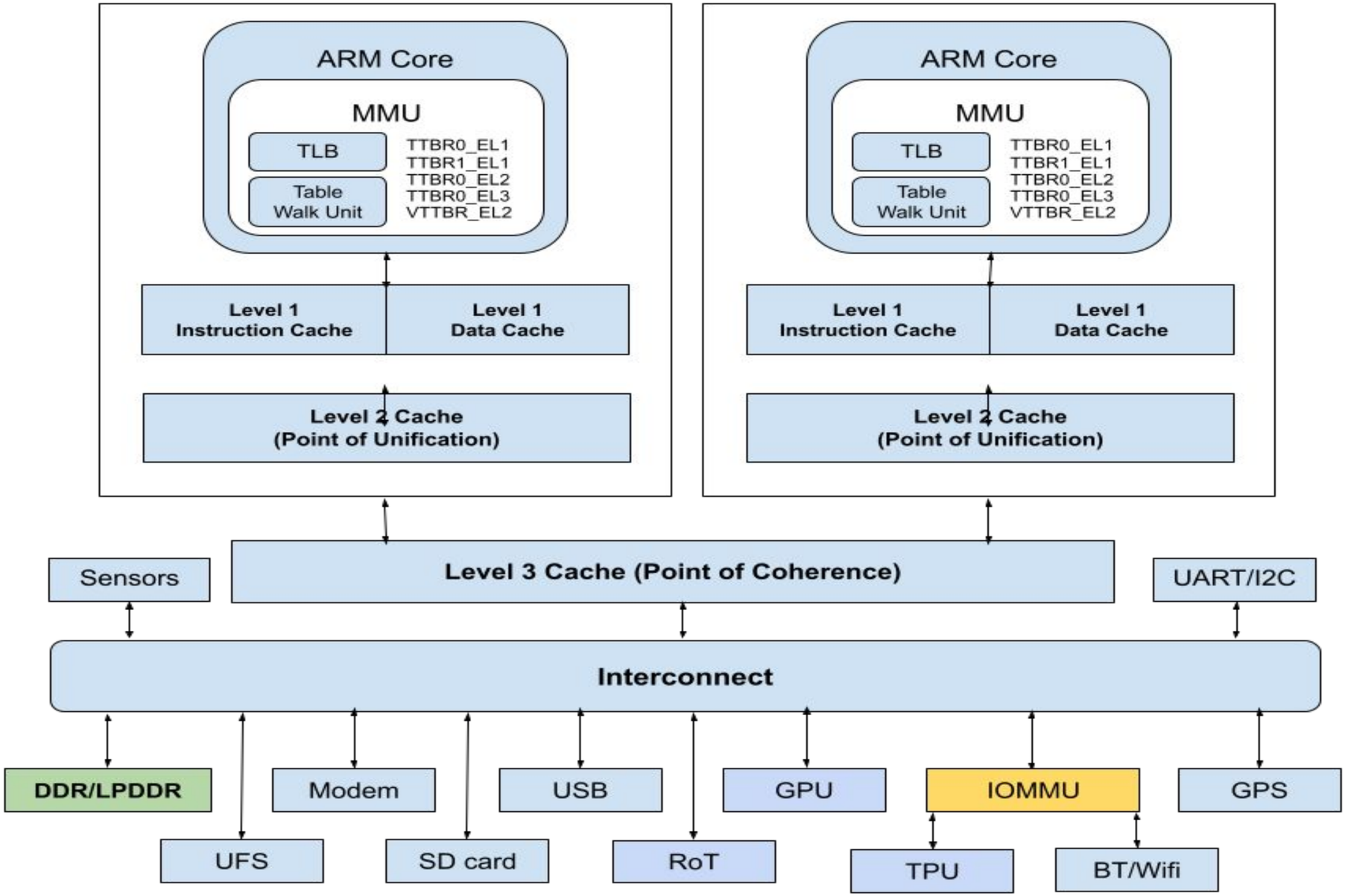
Where is the pointer to the page tables located?

In the Translation Table Base Register (TTBR)

How many **TTBRs** there could be in the **ARMv8** implementations per core?

```
TTBR0_EL1 -> User Space
TTBR1_EL1 -> Linux Kernel
TTBR0_EL2 -> Hypervisor
TTBR0_EL3 -> Secure World
VTTBR_EL2 -> Second Stage Translation
             to support virtualization
```

System on Chip



Bootloader: Supporting 16kb page sizes

Issue

Every exception level (EL0-EL3) can have a different page size configuration.

How do we share memory between exception levels?

Solution

As per the ARM Firmware Framework for Arm A-profile (section 4.6 Memory granularity and alignment), these constraints have to be met:

- *If X is the larger translation granule size used by the two translation regimes, then the size of the memory region must be a multiple of X.*
- *The base address of the memory region must be aligned to X.*
- *Size of a* memory region must be a multiple of 16KiB and expressed as a count of 4KiB pages.

Bootloader: Example

Example

- EL1 wants to share 32768 (**`BUFFER_SIZE`**) with EL3.
- EL3 has 4096 granule size (**`EL3_PAGE_SIZE`**)
- EL1 has 16394 granule size (**`EL1_PAGE_SIZE`**)
- BUFFER_SIZE has to be aligned to **`EL1_PAGE_SIZE`** (32768 % 16394 == 0)
- **`MEMORY_REGION_UNIT_SIZE`** = 4096

The number of “page counts” to share with EL3 is given by the formula:

```
EL3_PAGE_COUNT = (BUFFER_SIZE / MEMORY_REGION_UNIT_SIZE)
               = (32768/4096) = 8
```

The “page counts” to share with EL3 is 8. This number can pass through a SMC call to EL3.

Driver issues: IOMMU & Contiguous Memory Allocator (CMA)

Issue - IOMMU

When the IOMMU was set up, PAGE_SIZE was used. This caused the devices connected to the IOMMU didn't work, or in the worst case, the kernel crashes.

Solution - IOMMU

Define a constant for the IOMMU page size. For example:

```
#define IOMMU_PAGE_SIZE 4096
#define IOMMU_BASE_SHIFT 12
```

Issue - CMA allocations failing

Allocations from reserved memory "shared-dma-pool" were failing.

Solution - Contiguous Memory Allocator

Align memory in "shared-dma-pool" to multiple of HUGE PAGE SIZE. In this case 32MiB. In 4KiB base-page-size system 4MiB is required.

Driver issues: Getting free memory

Issue - Allocation extra memory due wrong assumptions

When there is the assumption that PAGE_SIZE = 4096, we can allocate more memory from what it is needed.

```
// Allocate 256KiB
void *buff = (void *)__get_free_pages(GFP_KERNEL | __GFP_ZERO, 6);
```

Solution - Allocation extra memory due wrong assumptions

Use get_order.

```
const int sz_256kb = 1 << 18;
void *buff = (void *)__get_free_pages(GFP_KERNEL | __GFP_ZERO,  get_order(sz_256kb));
```

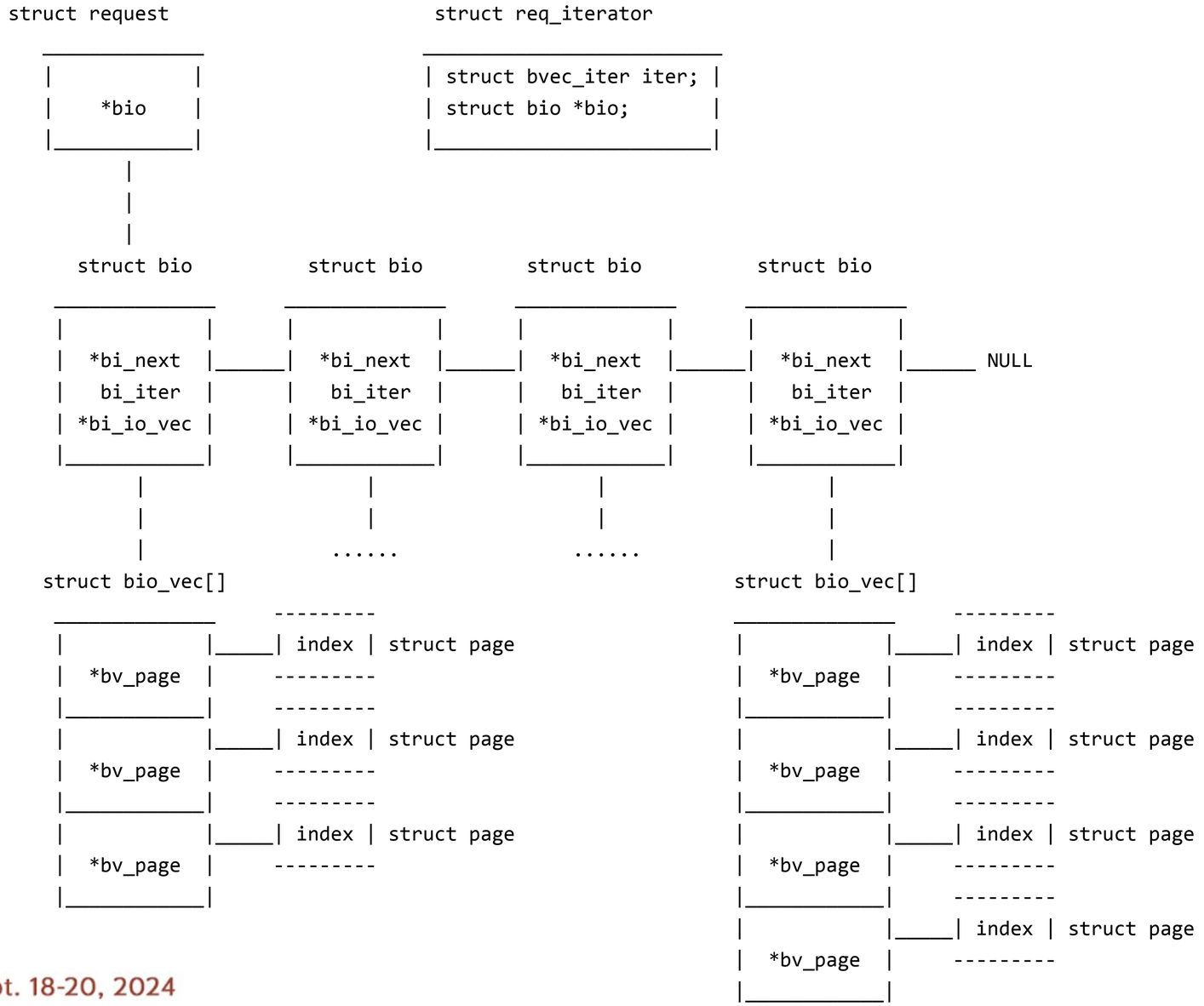
Linux Block Layer: Block I/O Request

Sector: Basic unit of the block devices. It is generally 512 bytes.

Block size: Basic unit of the file system. In Linux:

block size <= PAGE_SIZE

Segment: It is a page or portion of a page that contains data of adjacent sectors in disk.



Linux Block Layer: Small Segment Issues

Issue - segment smaller than PAGE_SIZE

Some UFS host controllers don’t follow the Host Controller Interface (HCI).
Some UFS host controllers don’t support 16384 segments.

Solution

Add support for segments smaller than PAGE_SIZE in the block layer.

https://r.android.com/q/topic:%22android15-6.6-ufs%22
https://r.android.com/q/topic:%22android15-6.1-ufs%22
https://r.android.com/q/topic:%22ufs-5-15-patches%22

Linux File System: Block Size <= PAGE_SIZE

In Android, there are 3 popular filesystems that support 16kb page sizes:

- ext4
- f2fs
- erofs

Sub-page blocks support

This means that the block size can be smaller or equal to the page size. For example, ext4 and erofs support sub-page blocks.

For f2fs the block size assumed 4096, support was added to relax this to block-size == page-size

ext4	4KiB Block Size	16KiB Block Size
4KiB PAGE_SIZE	Supported	Not supported
16KiB PAGE_SIZE	Supported	Supported

f2fs	4KiB Block Size	16KiB Block Size
4KiB PAGE_SIZE	Supported	Not supported
16KiB PAGE_SIZE	Not supported	Supported

erofs	4KiB Block Size	16KiB Block Size
4KiB PAGE_SIZE	Supported	Not supported
16KiB PAGE_SIZE	Supported	Supported

Hardware that does not support 16kb page size

Issue - HW does not support 16kb PAGE_SIZE

The hardware does not support 16kb page size.

Solution

Disable the hw and provide an alternative option if available.

init.compression.rc

```
on init && property:ro.boot.hardware.cpu.pagesize=4096
   write /sys/block/zram0/hw enable
   write /sys/block/zram0/cpu disable

on init && property:ro.boot.hardware.cpu.pagesize=16384
   write /sys/block/zram0/hw disable
   write /sys/block/zram0/cpu enable
```

Issue - CSR Registers are 4kb aligned

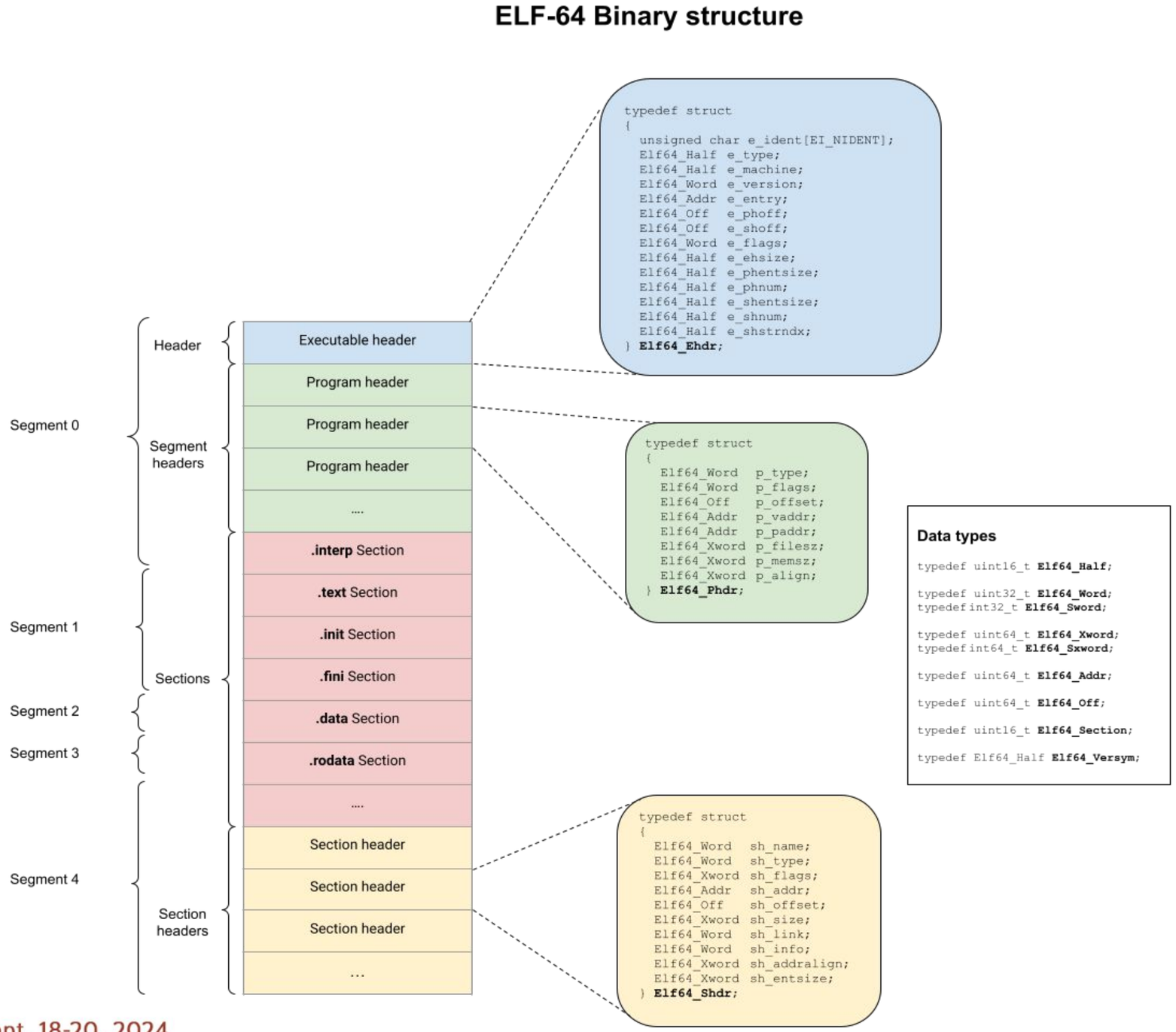
The Control Status Registers (CSR) are 4kb aligned and the mailboxes can not be configured when the address is not 16kb multiple.

Solution

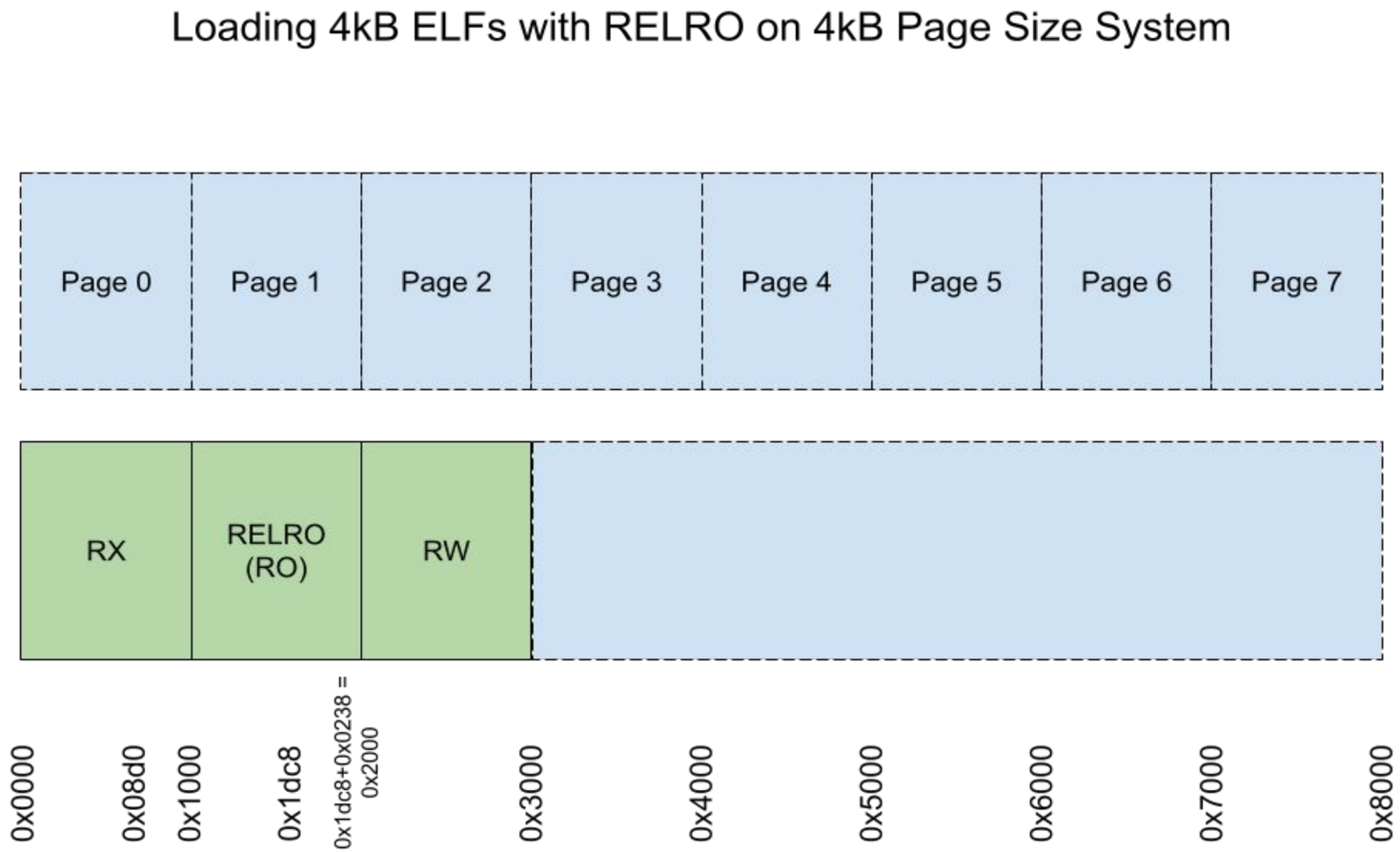
Only use the mailboxes that are 16kb aligned.

User Space

Executable and Linkable Format (ELF)

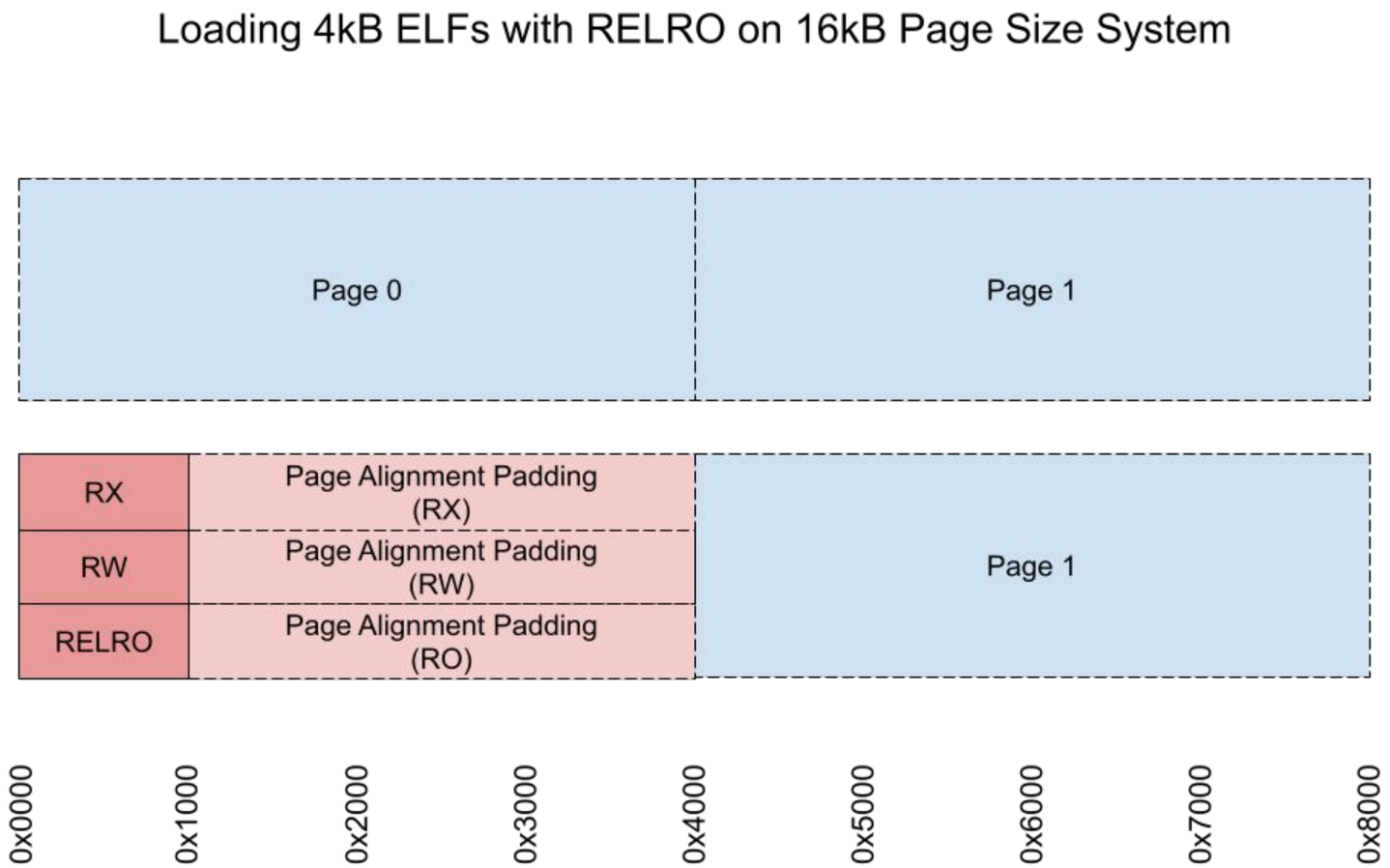


Why the shared libraries need to be 16kb elf aligned?



The top row shows page boundaries in memory and the bottom row shows how segment boundaries (permissions boundaries) would be mapped onto the underlying pages.

Why the shared libraries need to be 16kb elf aligned?



The top row shows page boundaries in memory and the bottom row shows how segment boundaries (permissions boundaries) would be mapped onto the underlying pages.

Android userspace: shared libraries and binaries

Android shared libraries and binaries

In **android targets**, these build variable have to be set:

PRODUCT_NO_BIONIC_PAGE_SIZE_MACRO := true

PRODUCT_MAX_PAGE_SIZE_SUPPORTED := 16384

https://source.android.com/docs/core/architecture/16kb-page-size/16kb

Vendor shared libraries (prebuilts)

The vendors have to provide the shared libraries compiled with

-Wl,-z,max-page-size=16384

https://source.android.com/docs/core/architecture/16kb-page-size/16kb#build-lib-16kb-alignment

Best practices

Use **getpagesize()**

Memory map regions multiple of **getpagesize()**

ART and User space memory Allocators

Remove 4096 assumptions from allocators -- scudo, jemalloc, ...

Update ART gerentation of OAT(ELF) images to 16KiB align the ELF segments.

Best practices

Use **getpagesize()**

Tools generating ELFs for a different target machine/architecture, use the max page size of the supported architectures for maximum portability.

APKs and zip align

Android APKs can be packaged so that uncompressed ELFs are located at page aligned (4096) boundaries in the zipped apk.

This is for security and space saving purposes.

```
zipalign -p -f -v 4 infile.apk outfile.apk
```

This was done so that the uncompressed ELFs can be mapped directly for the offset in the zipped APK.

For 16KiB age size the APKs need to use a zip-alignment of 16KiB

```
zipalign -P 16 -f -v 4 infile.apk outfile.apk
```

https://developer.android.com/tools/zipalign#usage

Best practices

If using **`zipalign`** to align apks, specify the **`-P 16`** option.

Questions

Resources

Resources

Platform developers resources

https://source.android.com/docs/core/architecture/16kb-page-size/16kb

Use ARM 64 emulator for 16kb page sizes
https://source.android.com/docs/core/architecture/16kb-page-size/getting-started-cf-arm64-pgagnostic

Use x86-64 emulator for 16kb page sizes
https://source.android.com/docs/core/architecture/16kb-page-size/getting-started-cf-x86-64-pgagnostic

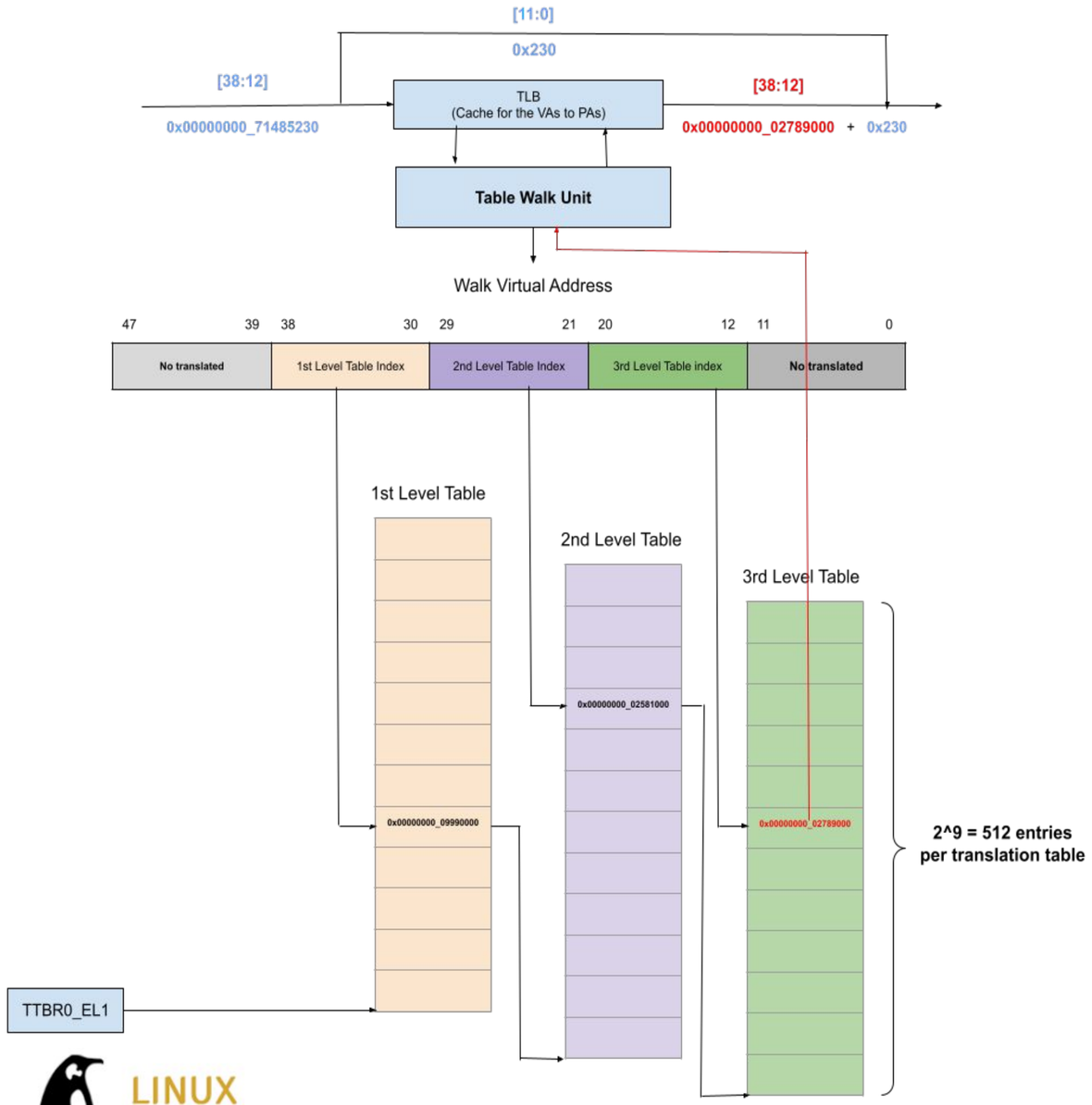
Enable 16kb toggle (switch between 4kb and 16kb kernels)
https://source.android.com/docs/core/architecture/16kb-page-size/16kb-developer-option

Application developers

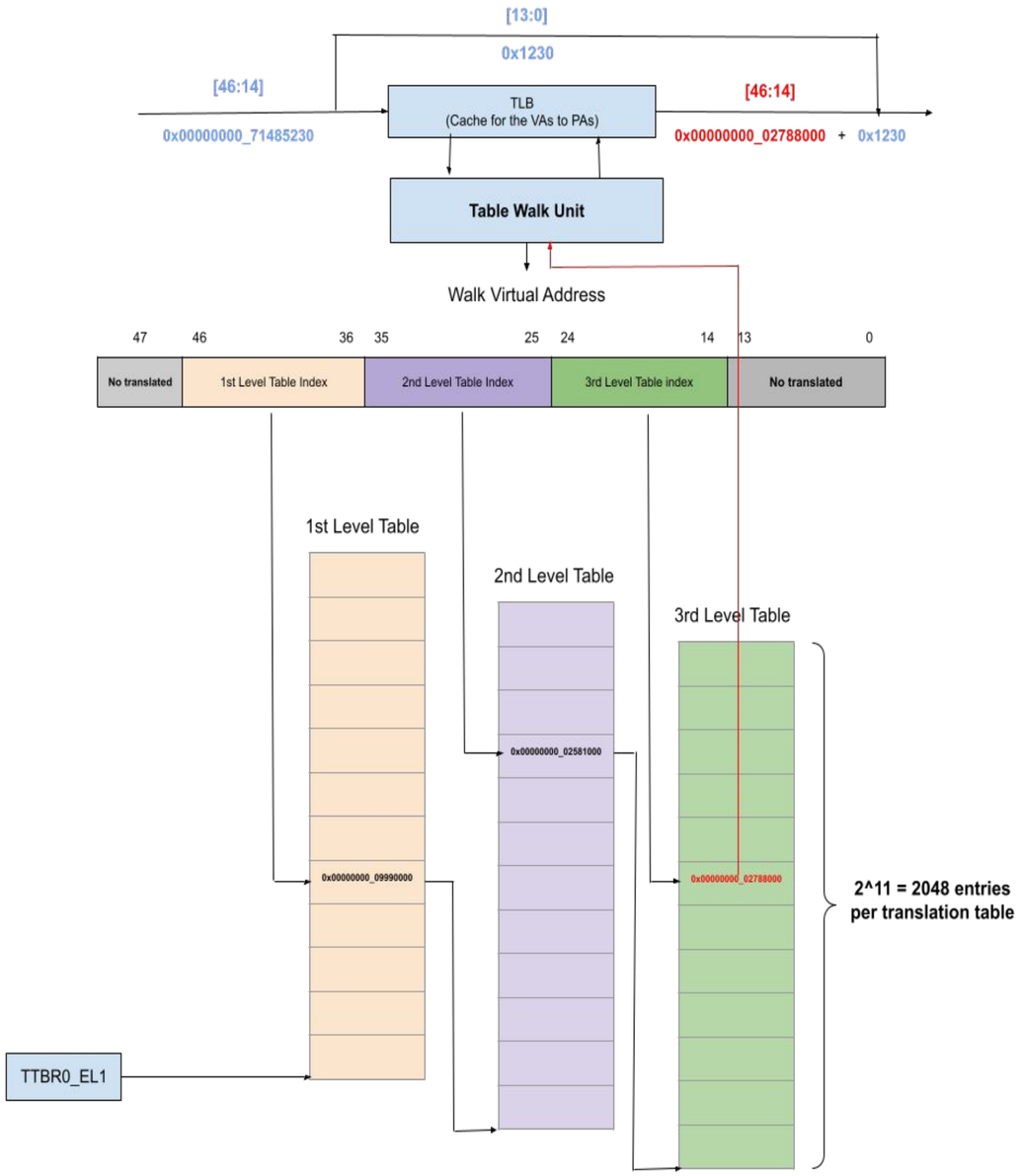
https://developer.android.com/guide/practices/page-sizes

Appendix

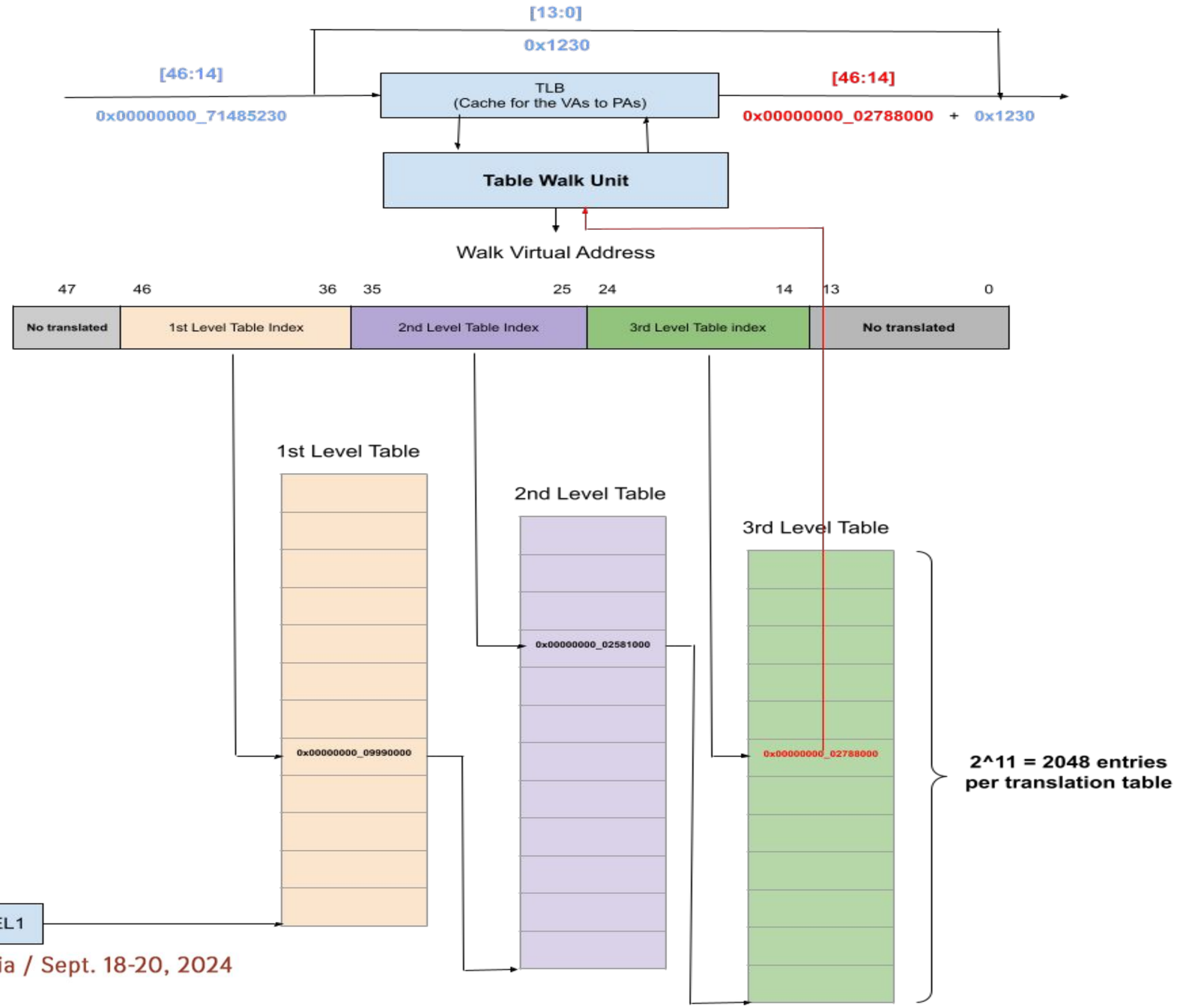
Page Table Walks - 4k Granule - 39-bits VA



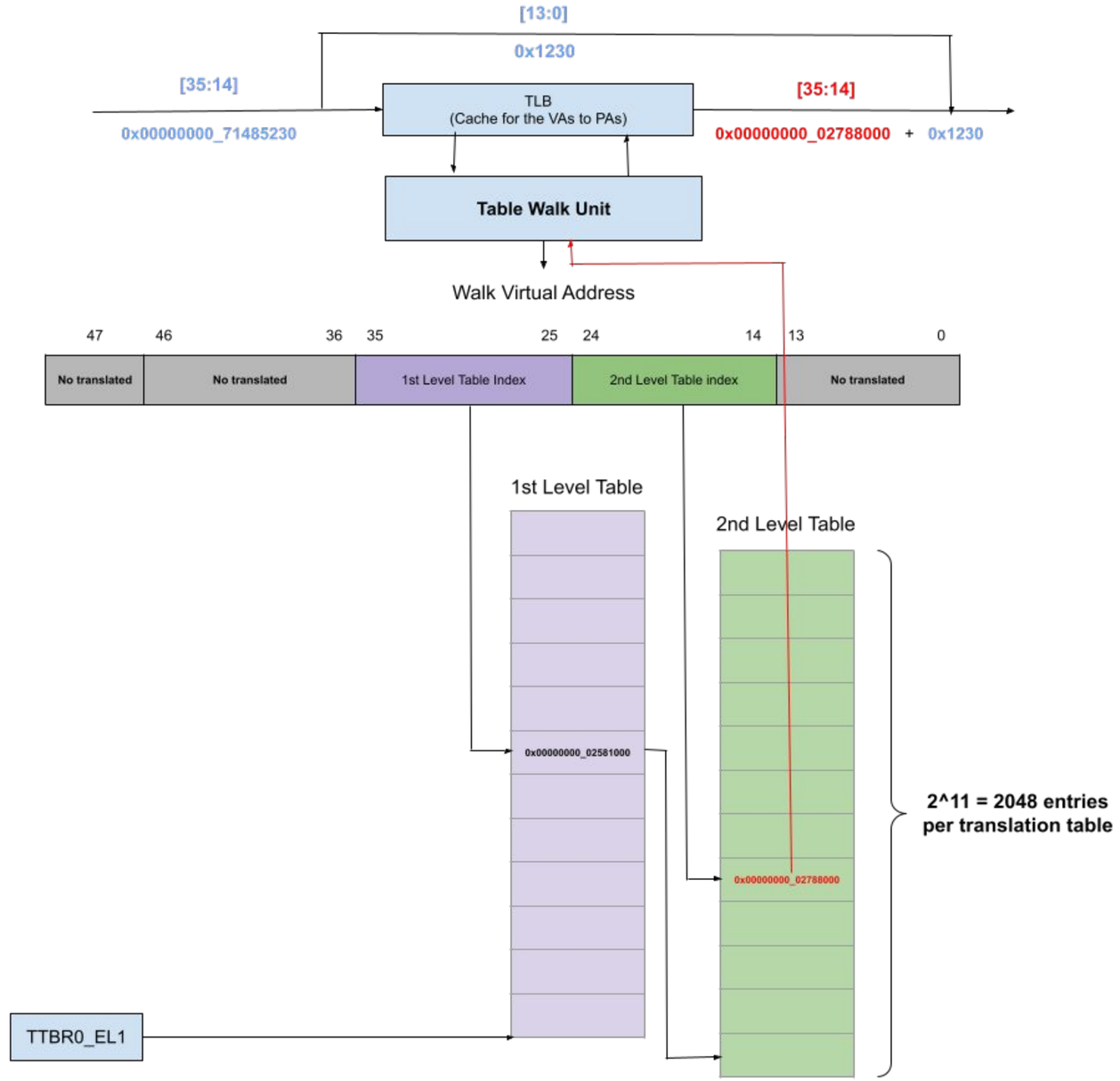
Page Table Walks - 16k Granule - 47-bits VA



Page Table Walks - 16k Granule - 47-bits VA



Page Table Walks - 16k Granule - 36-bits VA



Page Table Walks - 16k Granule - 48-bits VA