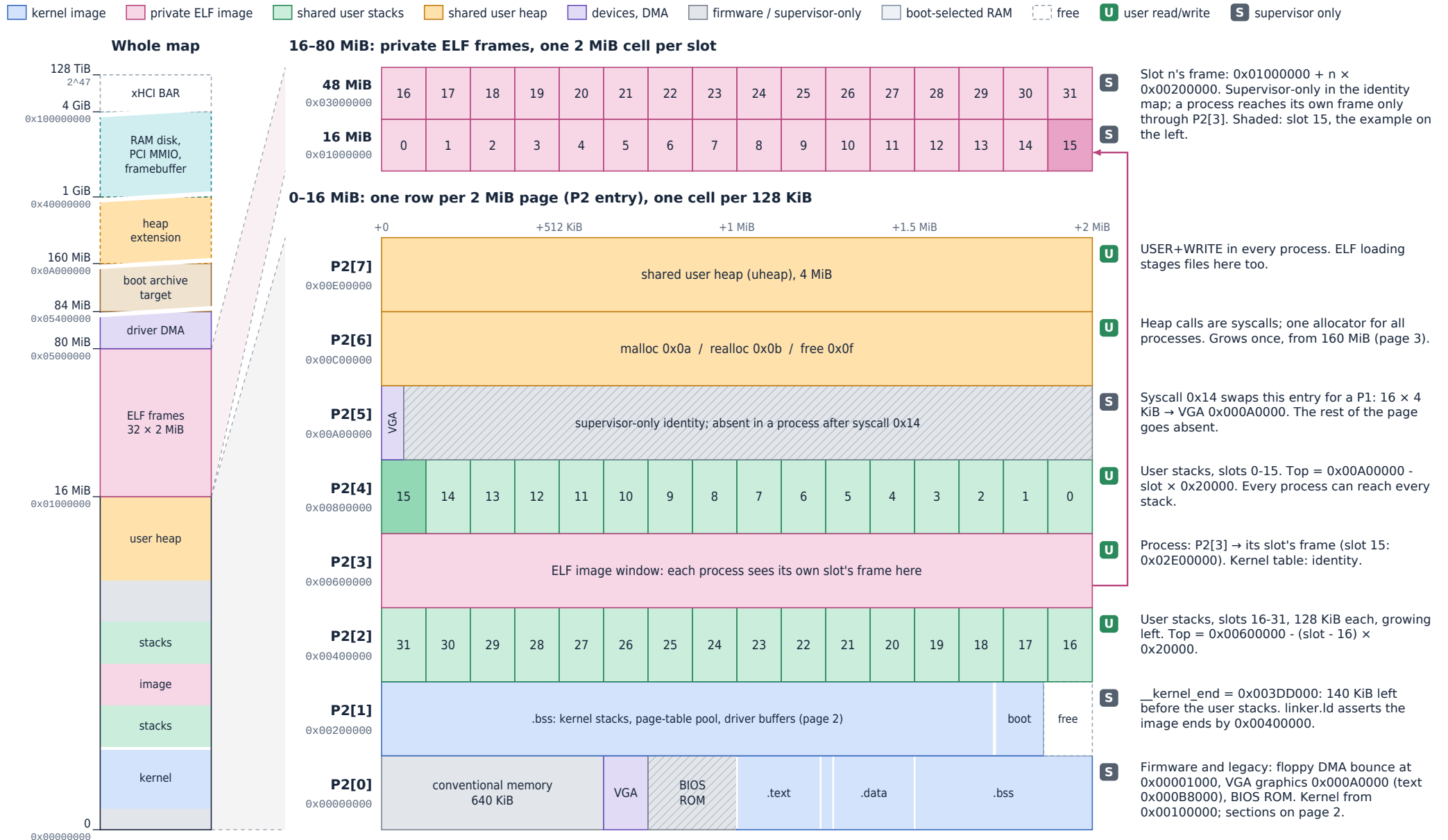


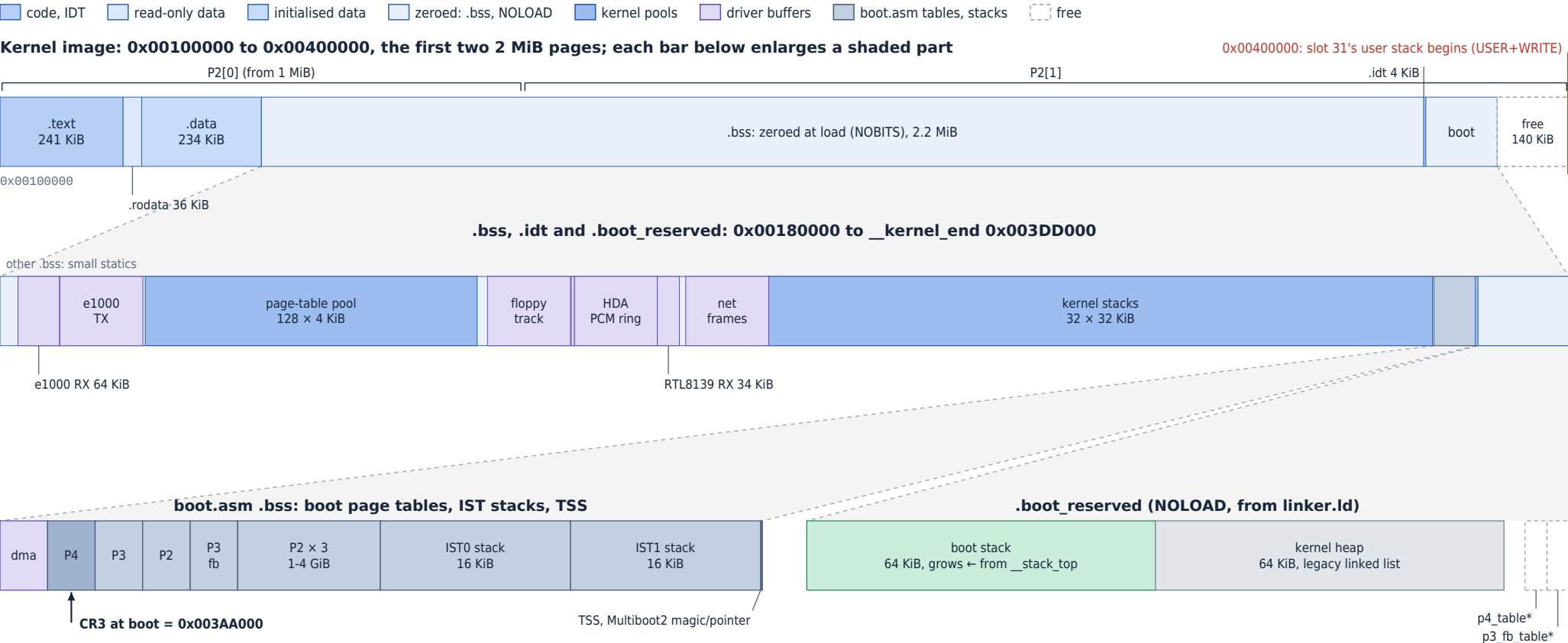
Address-space overview

boot.asm identity-maps 0–4 GiB in 2 MiB pages. Left: the whole map, not to scale. Right: the fixed low 80 MiB to scale. An identity mapping does not imply RAM behind it.



Linked kernel image

To scale, from target/kernel_graphics/x86_64-r2/release/kernel.elf (SHA-256 prefix fdc83a3b560072b9). Addresses are build-specific.



Sections

section	start	end	size	contents
.multiboot2_header	0x00100000	0x00100030	48 B	GRUB entry metadata
.text	0x00100030	0x0013C429	241 KiB	Kernel and driver code
.rodata	0x0013C430	0x001455BC	36 KiB	Constants, GDT, unwind stubs
.data	0x001455C0	0x0017FEF0	234 KiB	Scheduler state, .dma input
.bss	0x00180000	0x003B9070	2.2 MiB	Zeroed at load (NOBITS)
.idt	0x003B9070	0x003BA070	4 KiB	Interrupt descriptors
.boot_reserved	0x003BA070	0x003DD000	140 KiB	Boot stack, heap (NOLOAD)
free	0x003DD000	0x00400000	140 KiB	Headroom below the user stacks

* linker.ld reservations. boot.asm loads its own P4 (0x003AA000, left) into CR3; nm's p4_table is this unused copy.

Pools, sizes checked against the source

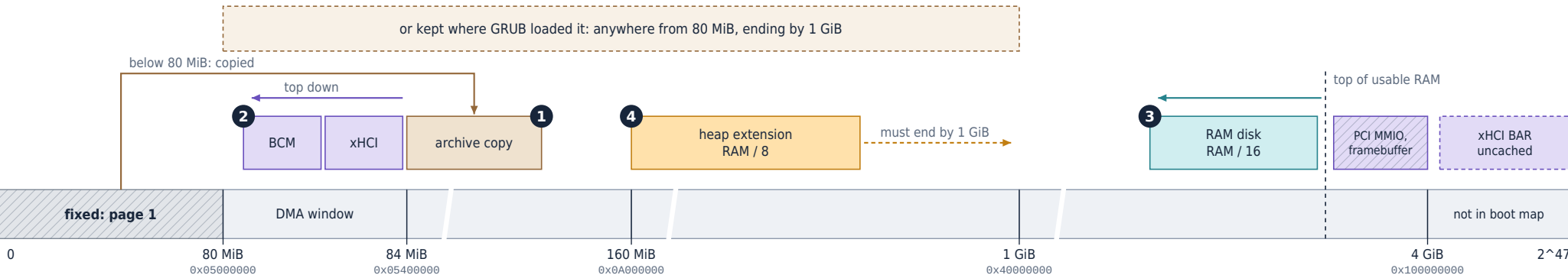
pool	range	size	source constant
Kernel stacks	[0x002A8740, 0x003A8740)	1 MiB	MAX_PROCESSES x STACK_SIZE
Process page tables	[0x001B8000, 0x00238000)	512 KiB	PAGE_TABLE_MEMORY_SIZE
Queued network frames	[0x0028873C, 0x002A873C)	128 KiB	FRAME_SLOTS x FRAME_SLOT_SIZE
HD Audio PCM ring	[0x0025D800, 0x0027D800)	128 KiB	hda::RING

Live boot tables: P4 0x003AA000, P3 0x003AB000, P2 0x003AC000, P2 for 1-4 GiB 0x003AE000. Driver DMA on page 3 is outside the image.

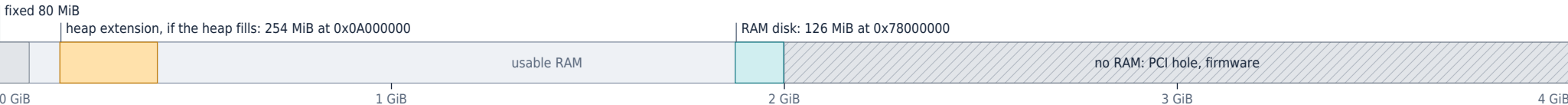
Boot-selected regions

Placed once per boot, in the numbered order, inside the Multiboot2 usable-RAM map. The upper map is schematic; the lower one is a real 2 GiB QEMU guest, to scale.

Where each region may go (schematic: segment widths are not to scale)



Example: QEMU -m 2G (SeaBIOS map), to scale. The archive's place depends on GRUB and is left out.

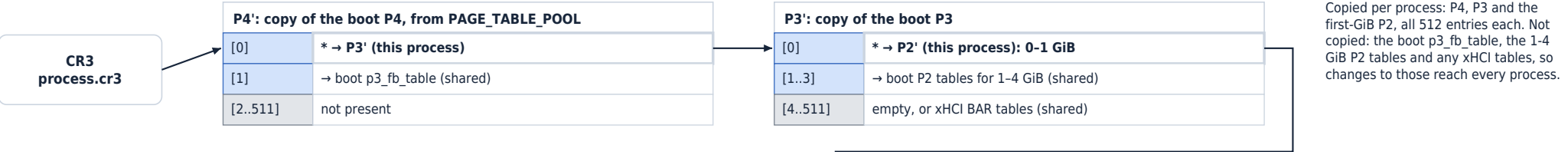


Boot archive: /mnt/tar 1 keep If GRUB loaded it at or above 80 MiB and it ends by 1 GiB. else Copy it to 0x05400000 (84 MiB), or to the first page past the boot information if that is in the way. check The copy must be usable RAM ending by 1 GiB; otherwise there is no archive. size The module's length. life Read-only; stays in RAM for the whole boot. Runs before anything writes above the kernel.	xHCI and Broadcom DMA 2 size size_of::<Dma>() each, 4 KiB-aligned; only when the device is present. where Highest fit below 84 MiB, so inside the 80-84 MiB gap. avoids Fixed memory, the archive, the boot information; Broadcom also avoids xHCI. holds xHCI: rings, contexts, scratchpads, up to four MMIO page tables. Broadcom: descriptor rings and packet buffers. access Supervisor-only.	RAM disk: /mnt/tmp 3 size Usable RAM / 16 in 2 MiB pages, at least 2 MiB. Halved until it fits; no mount if 2 MiB does not. where Highest 2 MiB-aligned fit below 4 GiB inside one usable region. avoids [0, 80 MiB), the archive, both DMA regions. format FAT16 (FAT12 at 2 MiB), formatted at every boot. access Supervisor-only identity mapping.	User-heap extension 4 when The first time the 4 MiB heap runs out; tried once. size Usable RAM / 8 in 2 MiB pages, at least 2 MiB; halved until it fits. where From 0x0A000000; past the archive (2 MiB-aligned) if they overlap. Ends by 1 GiB. checks One usable region; not over /mnt/tmp. access USER+WRITE, patched into the kernel P2 and every live process's P2 (page 4).
---	--	--	--

Kernel and process page tables

create_user_page_table(slot) copies three boot tables per process and rewires them. * marks every entry that can differ from the kernel's; the rest is shared.

Walk for a process in slot n



P2 and P2': the first GiB in 2 MiB pages

entry	covers	kernel P2 (boot.asm)	process P2' (copy)
[0]	0-2 MiB: firmware, kernel	S identity	same
[1]	2-4 MiB: kernel	S identity	same
[2]	4-6 MiB: stacks of slots 16-31	U identity	same
[3]	6-8 MiB: ELF image window	U identity	U * slot n's frame
[4]	8-10 MiB: stacks of slots 0-15	U identity	same
[5]	10-12 MiB: VGA window	S identity	U * after 0x14: P1
[6]	12-14 MiB: user heap	U identity	same
[7]	14-16 MiB: user heap	U identity	same
[8..79]	16-160 MiB: frames, DMA, archive	S identity	same
[80..511]	160 MiB-1 GiB: heap extension	S identity, U once grown	U * once grown, too

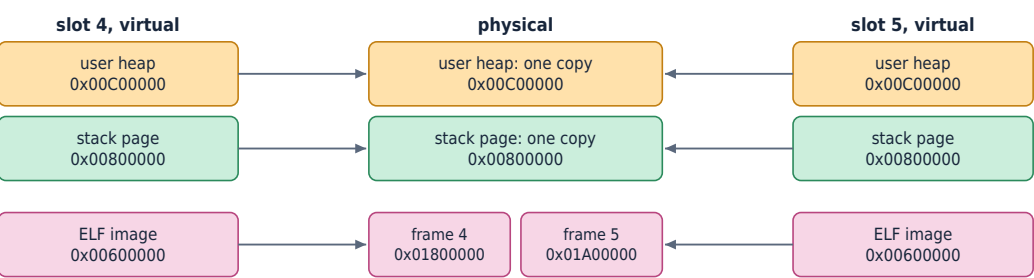
The kernel walks the boot tables themselves. save_kernel_cr3() records the CR3 live when init_processes() runs as KERNEL_CR3, the source of every copy. Later changes to the kernel's first-GiB P2 miss running processes' copies, so set_user_range() patches each copy too.

slot n's private 2 MiB frame
0x01000000 + n × 0x00200000

P1: 16 × 4 KiB → VGA 0x000A0000
rest of the 2 MiB page: absent

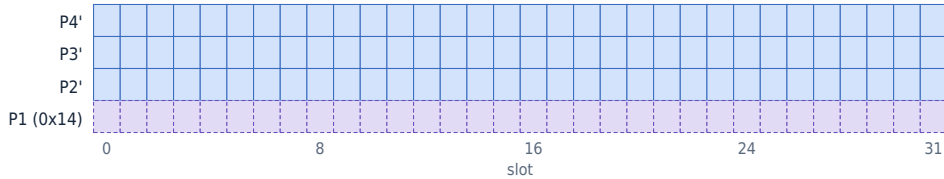
set_user_range(): USER+WRITE in the kernel P2 and every registered P2'

Same virtual addresses, different frames



Both slots run code at 0x00600000; P2[3] sends each to its own frame. Stacks and the heap resolve to the same physical pages in every process.

PAGE_TABLE_POOL: 128 × 4 KiB at 0x001B8000 (.bss)



4 pages per slot is capacity, not placement: a bump pointer hands out fresh pages, then a free list of returned ones, zeroed before reuse. Running out panics.

