Week 8
CS4973/CS6640
02/26 2025
https://naizhengtan.github.io/25spring/

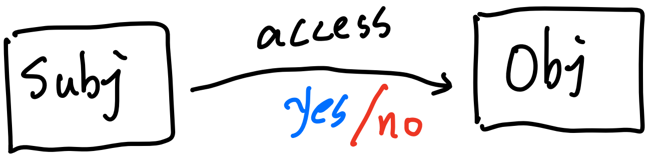
Memory protection introduction

☑ 1. Memory protection, the problem statement
☑ 2. Segmentaion (x86-32)
☑ 3. PMP (RISC-V)
☑ 4. Paging (brief)
☑ 5. Meltdown & its consequences
□ 6. Other possible solutions

Q: Motivation?

- process A → process B (X)
- user process → kernel objs (x)
- process A $\xrightarrow{\text{arbitrarily (X)}}$ all parts of A's memory (bug)

Q: Problem? Access Control



Q: WHY r/w/x ?

CPU $\xrightarrow{\text{op: r/w/x}}$ memory

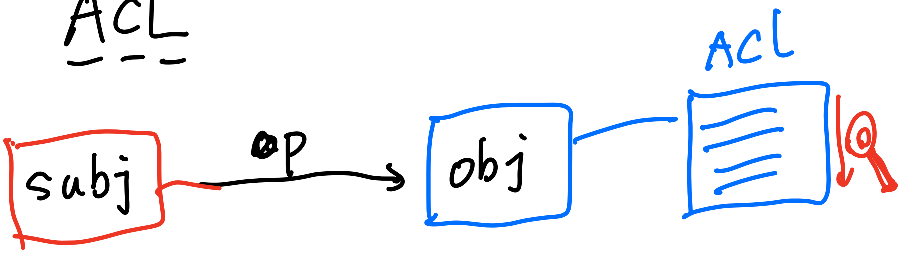
subj: instruction + ctx
op: read/write/execute
obj: a set of mem addresses

- invalid memory accesses.

invalid
- subj: user_level app → kernel obj
- op: process A $\xrightarrow{\text{exec}}$ own stack
- obj: process A → processB's mem

- abstract Solution ?

ACL



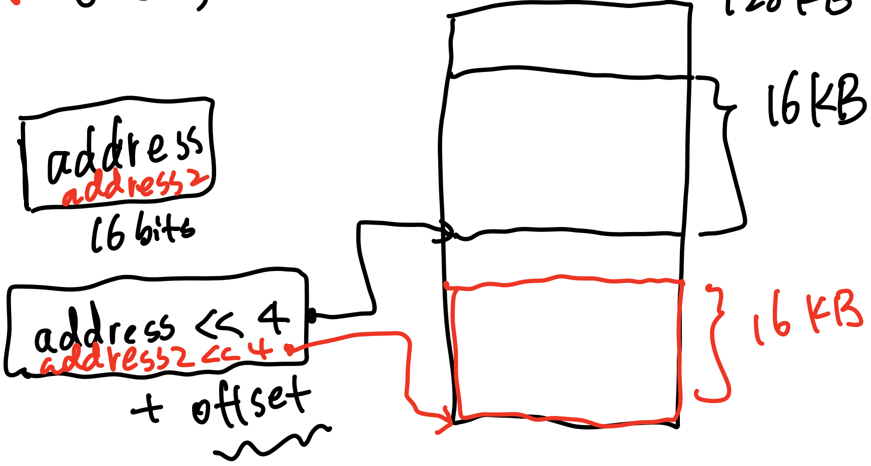
```
Multiple questions:
    Q1: what are memory objs? (memory granularity)
    Q2: who is the subj? (how to define subj)
    Q3: where to store the ACL?
```

2. Segmentation (X86)

- 8086, 1978 (16bit) → $\boxed{16 \text{ bits}}$ → 2^{16} B = $2^6 \cdot \underline{2^{10}}$ B = 64KB

physical mem 128KB

2^{20} → 1MB



- 80386, 1985 (X86-32)

0xdeadbeef → $\boxed{123}$

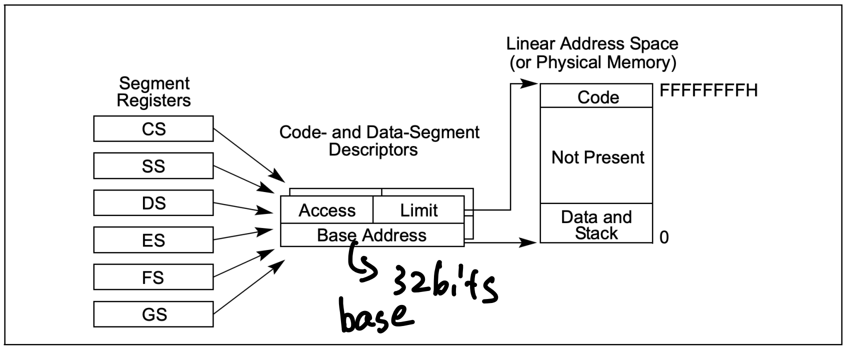


Figure 3-2. Flat Model

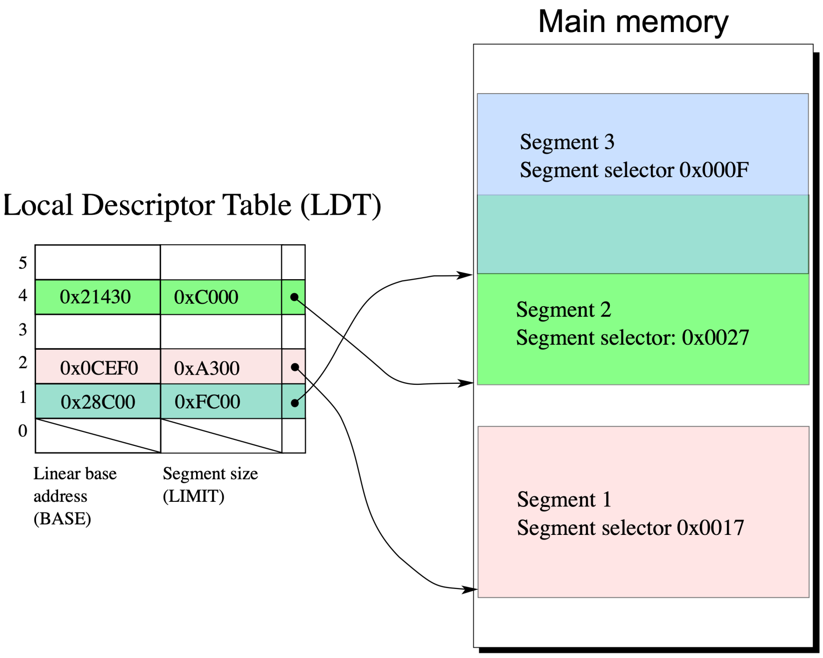
base + 0xdeadbeef

↳ physical addr

① check > limit

② check access

0xdeadbeef →



Q1: what are memory objs? (memory granularity)

segment (base, limit)

Q2: who is the subj? (how to define subj)

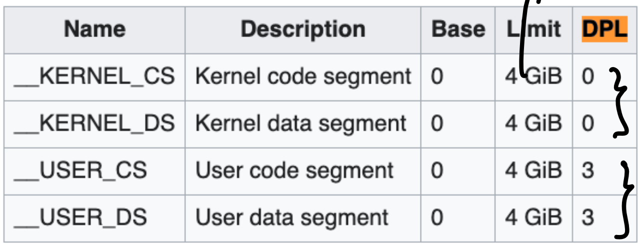
instruction + DPL (register status)

Q3: where to store the ACL?

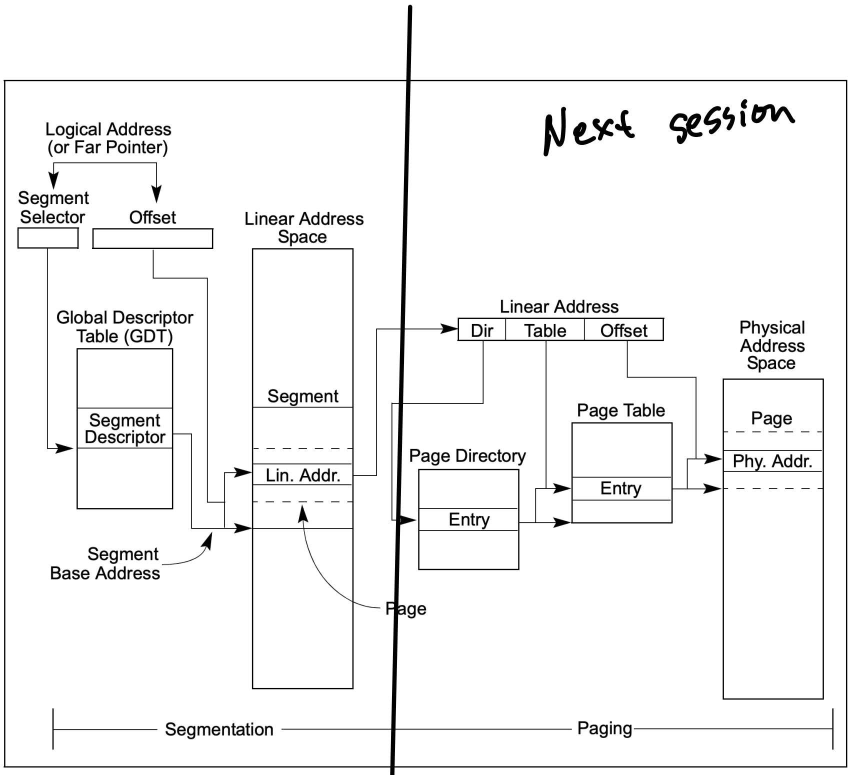
descriptor (in memory) + selector (register)

Linux: (32bits)

2^{32}



Name	Description	Base	Limit	DPL
__KERNEL_CS	Kernel code segment	0	4 GiB	0
__KERNEL_DS	Kernel data segment	0	4 GiB	0
__USER_CS	User code segment	0	4 GiB	3
__USER_DS	User data segment	0	4 GiB	3

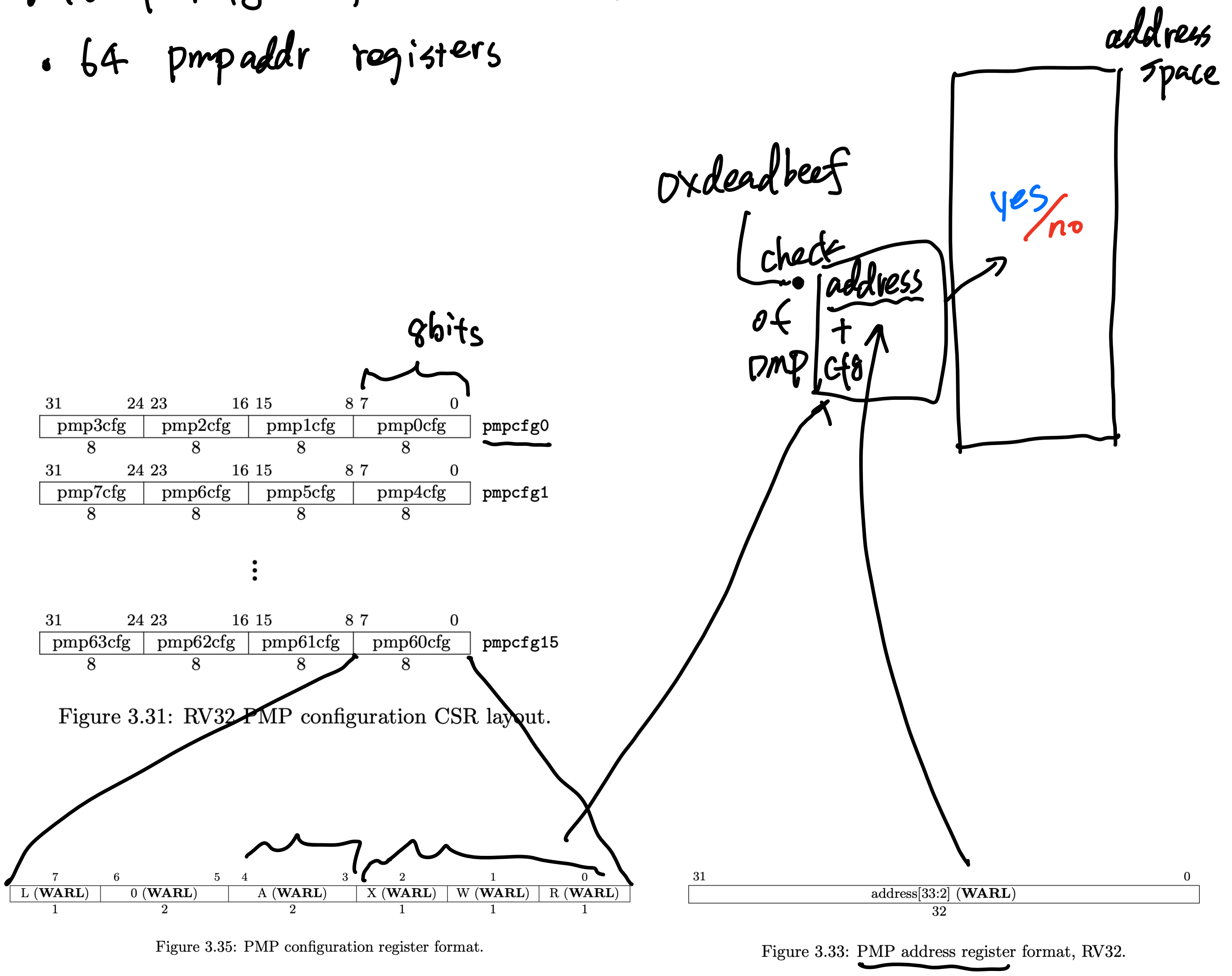


Next session

Figure 3-1. Segmentation and Paging

3. PMP (RISC-V)

- 16 pmpcfg registers $(16 \times \frac{32}{8} = 64)$
- 64 pmpaddr registers



31 24	23 16	15 8	7 0	
pmp3cfg	pmp2cfg	pmp1cfg	pmp0cfg	**pmpcfg0**
8	8	8	8	

31 24	23 16	15 8	7 0	
pmp7cfg	pmp6cfg	pmp5cfg	pmp4cfg	**pmpcfg1**
8	8	8	8	

⋮

31 24	23 16	15 8	7 0	
pmp63cfg	pmp62cfg	pmp61cfg	pmp60cfg	**pmpcfg15**
8	8	8	8	

Figure 3.31: RV32 PMP configuration CSR layout.

7	6 5	4 3	2	1	0
L (**WARL**)	0 (**WARL**)	A (**WARL**)	X (**WARL**)	W (**WARL**)	R (**WARL**)
1	2	2	1	1	1

Figure 3.35: PMP configuration register format.

31 0
address[33:2] (**WARL**)
32

Figure 3.33: PMP address register format, RV32.

```
Q1: what are memory objs? (memory granularity)
```

PMP segment (PMP addr + PMP cfg)

```
Q2: who is the subj? (how to define subj)
```

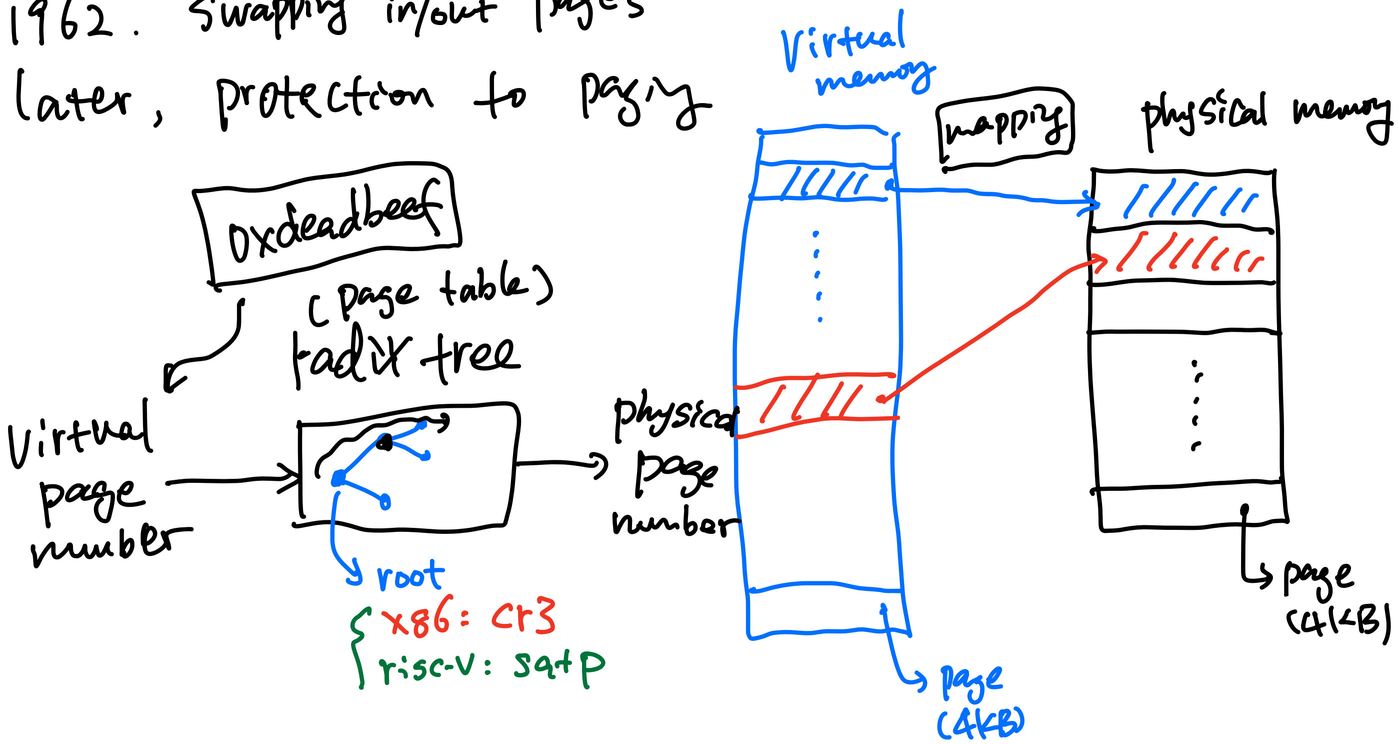
instruction

```
Q3: where to store the ACL?
```

registers

4. paging (brief)

- 1962. swapping in/out pages
- later, protection to paging



Q1: what are memory objs? (memory granularity)

page

Q2: who is the subj? (how to define subj)

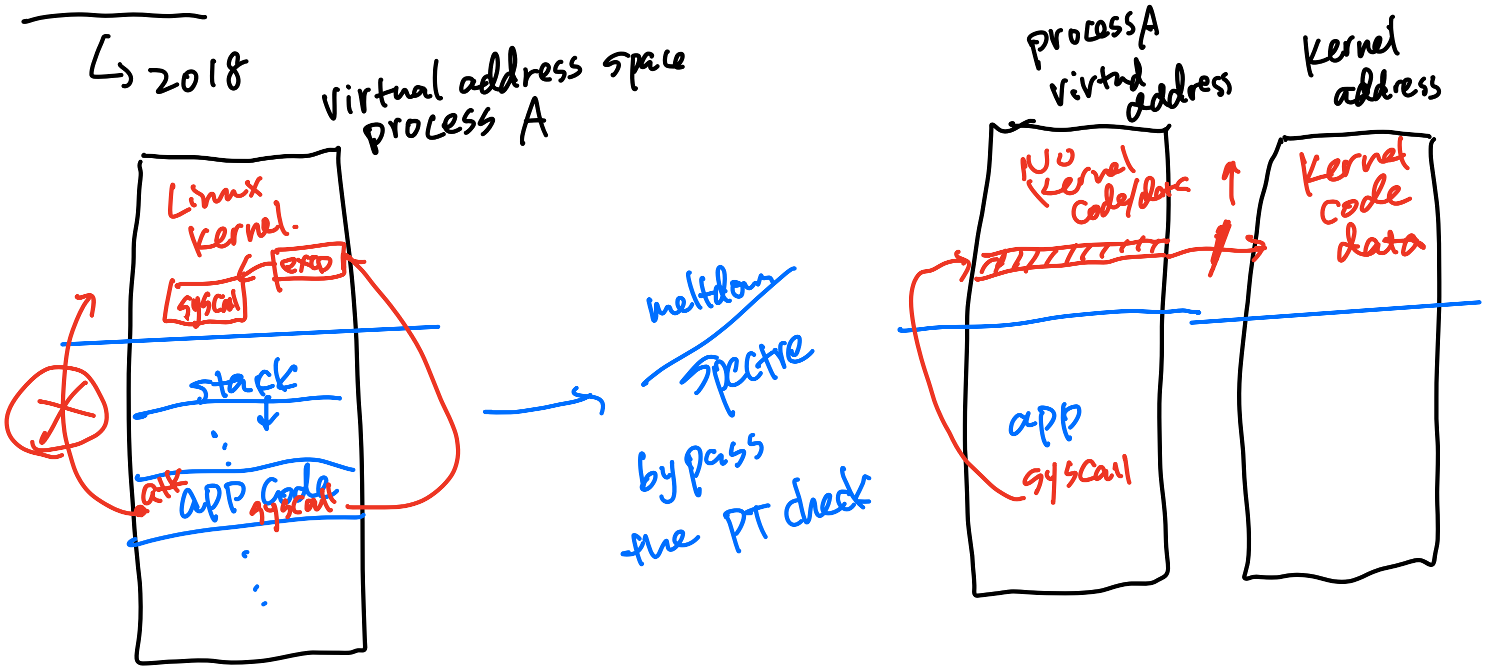
instruction + satp + priv_level

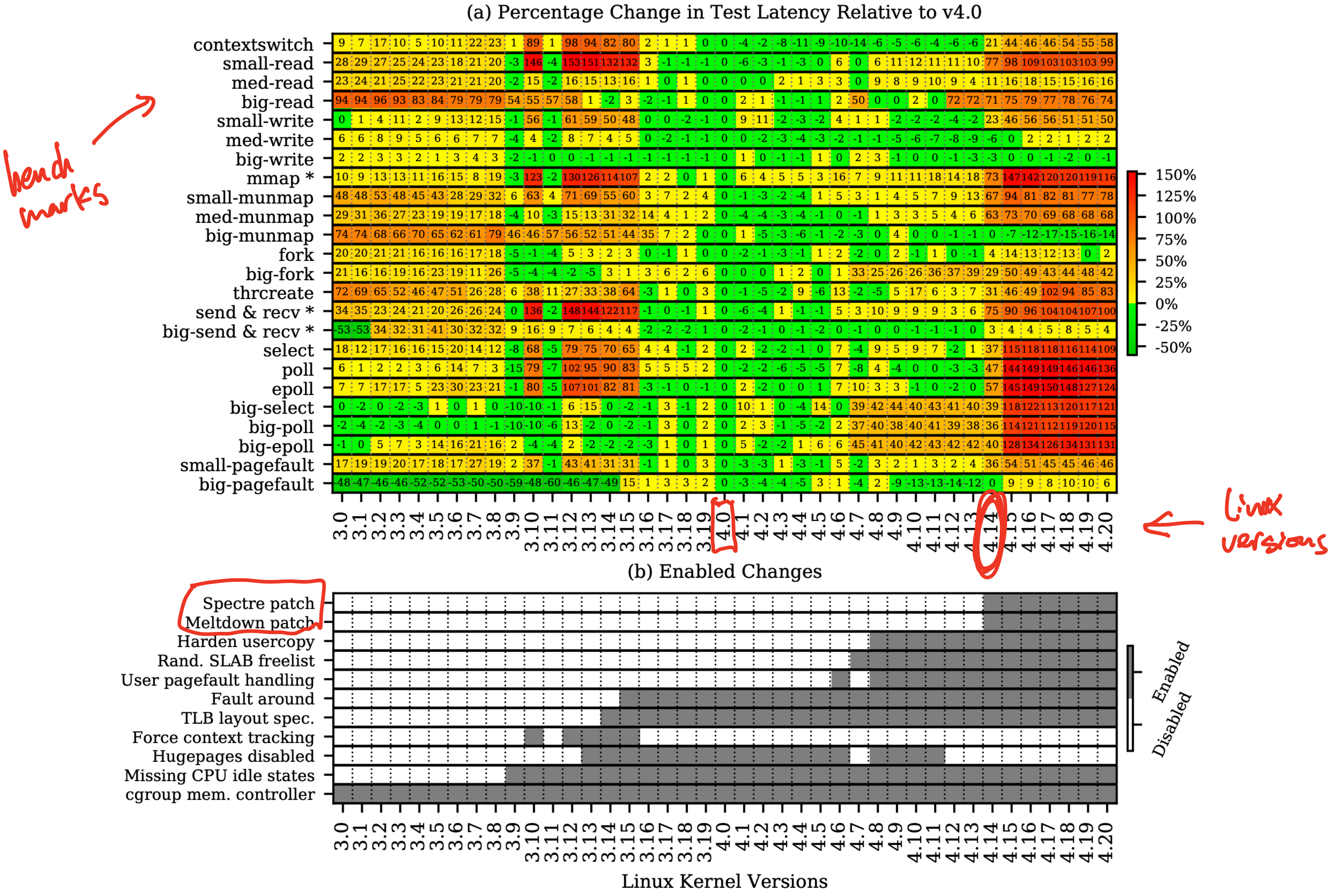
Q3: where to store the ACL?

memory (indexed by satp)

5. meltdown & Linux perf

↳ 2018




(a) Percentage Change in Test Latency Relative to v4.0

	3.0	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9	3.10	3.11	3.12	3.13	3.14	3.15	3.16	3.17	3.18	3.19	4.0	4.1	4.2	4.3	4.4	4.5	4.6	4.7	4.8	4.9	4.10	4.11	4.12	4.13	4.14	4.15	4.16	4.17	4.18	4.19	4.20
contextswitch	9	7	17	10	5	10	11	22	23	1	89	1	98	94	82	80	2	1	1	0	0	-4	-2	-8	-11	-9	-10	-14	-6	-5	-6	-4	-6	-6	21	44	46	46	54	55	58
small-read	28	29	27	25	24	23	18	21	20	-3	146	-4	153	151	132	132	3	-1	-1	-1	0	-6	-3	-1	-3	0	6	0	6	11	12	11	11	10	77	98	109	103	103	103	99
med-read	23	24	21	25	22	23	21	21	20	-2	15	-2	16	15	13	16	1	0	1	0	0	0	0	2	1	3	3	0	9	8	9	10	9	4	11	16	18	15	15	16	16
big-read	94	94	96	93	83	84	79	79	79	54	55	57	58	1	-2	3	-2	-1	1	0	0	2	1	-1	-1	1	2	50	0	0	2	0	72	72	71	75	79	77	78	76	74
small-write	0	1	4	11	2	9	13	12	15	-1	56	-1	61	59	50	48	0	0	-2	-1	0	9	11	-2	-3	-2	4	1	1	-2	-2	-2	-4	-2	23	46	56	56	51	51	50
med-write	6	6	8	9	5	6	6	7	7	-4	4	-2	8	7	4	5	0	-2	-1	-1	0	0	-2	-3	-4	-3	-2	-1	-1	-5	-6	-7	-8	-9	-6	0	2	2	1	2	2
big-write	2	2	3	3	2	1	3	4	3	-2	-1	0	0	-1	-1	-1	-1	-1	-1	-1	0	1	0	-1	-1	1	0	2	3	-1	0	0	-1	-3	-3	0	0	-1	-2	0	-1
mmap *	10	9	13	13	11	16	15	8	19	-3	123	-2	130	126	114	107	2	2	0	1	0	6	4	5	5	3	16	7	9	11	11	18	14	18	73	147	142	120	120	119	116
small-munmap	48	48	53	48	45	43	28	29	32	6	63	4	71	69	55	60	3	7	2	4	0	-1	-3	-2	-4	1	5	3	1	4	5	7	9	13	67	94	81	82	81	77	78
med-munmap	29	31	36	27	23	19	19	17	18	-4	10	-3	15	13	31	32	14	4	1	2	0	-4	-4	-3	-4	-1	0	-1	1	3	3	5	4	6	63	73	70	69	68	68	68
big-munmap	74	74	68	66	70	65	62	61	79	46	46	57	56	52	51	44	35	7	2	0	0	1	-5	-3	-6	-1	-2	-3	0	4	0	0	-1	-1	0	-7	-12	-17	-15	-16	-14
fork	20	20	21	21	16	16	16	17	18	-5	-1	-4	5	3	2	3	0	-1	1	0	0	-2	-1	-3	-1	1	2	-2	0	2	-1	1	0	-1	4	14	13	12	13	0	2
big-fork	21	16	16	19	16	23	19	11	26	-5	-4	-4	-2	-5	3	1	3	6	2	6	0	0	0	1	2	0	1	33	25	26	26	36	37	39	29	50	49	43	44	48	42
thrcreate	72	69	65	52	46	47	51	26	28	6	38	11	27	33	38	64	-3	1	0	3	0	-1	-5	-2	9	-6	13	-2	-5	5	17	6	3	7	31	46	49	102	94	85	83
send & recv *	34	35	23	24	21	20	26	26	24	0	136	-2	148	144	122	117	-1	0	-1	1	0	-6	-4	1	-1	-1	5	3	10	9	9	9	3	6	75	90	96	104	104	107	100
big-send & recv *	-53	-53	34	32	31	41	30	32	32	9	16	9	7	6	4	4	-2	-2	-2	1	0	-2	-1	0	-1	-2	0	-1	0	0	-1	-1	-1	0	3	4	4	5	8	5	4
select	18	12	17	16	16	15	20	14	12	-8	68	-5	79	75	70	65	4	4	-1	2	0	2	-2	-2	-1	0	7	-4	9	5	9	7	-2	1	37	115	118	118	116	114	109
poll	6	1	2	2	3	6	14	7	3	-15	79	-7	102	95	90	83	5	5	5	2	0	-2	-2	-6	-5	-5	7	-8	4	-4	0	0	-3	-3	47	144	149	149	146	146	136
epoll	7	7	17	17	5	23	30	23	21	-1	80	-5	107	101	82	81	-3	-1	0	-1	0	2	-2	0	0	1	7	10	3	3	-1	0	-2	0	57	145	149	150	148	127	124
big-select	0	-2	0	-2	-3	1	0	1	0	-10	-10	-1	6	15	0	-2	-1	3	-1	2	0	10	1	0	-4	14	0	39	42	44	40	43	41	40	39	118	122	113	120	117	121
big-poll	-2	-4	-2	-3	-4	0	0	1	-1	-10	-10	-6	13	-2	0	-2	-1	3	0	2	0	2	3	-1	-5	-2	2	37	40	38	40	41	39	38	36	114	121	112	119	120	115
big-epoll	-1	0	5	7	3	14	16	21	16	2	-4	-4	2	-2	-2	-2	-3	1	0	1	0	5	-2	-2	1	6	6	45	41	40	42	43	42	42	40	128	134	126	134	131	131
small-pagefault	17	19	19	20	17	18	17	27	19	2	37	-1	43	41	31	31	-1	1	0	3	0	-3	-3	1	-3	-1	5	-2	3	2	1	3	2	4	36	54	51	45	45	46	46
big-pagefault	-48	-47	-46	-46	-52	-52	-53	-50	-50	-59	-48	-60	-46	-47	-49	15	1	3	3	2	0	-3	-4	-4	-5	3	1	-4	2	-9	-13	-13	-14	-12	0	9	9	8	10	10	6

(b) Enabled Changes

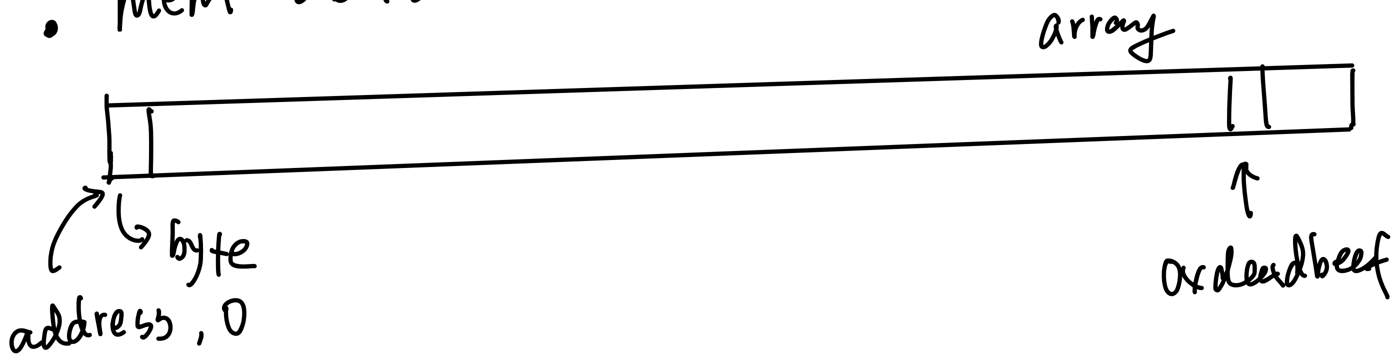
Linux Kernel Versions

BACK UP Pages

Q: memory protection $\overset{?}{=}$ access control

Q: r/w/x ?

- mem abstraction



- tree-base mem abstraction?
- infinite long memory