Week 7.b
CS6640
10/19 2023
https://naizhengtan.github.io/23fall/

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?

- a process A → process B's memory
- process → kernel / invalid
- process → ~~unused~~ memory (bug) ←

(code $\xrightarrow{r/w/x}$ memory should not access)

- access control

$$\boxed{subj} \xrightarrow{access} \boxed{obj}$$

$$CPU \xrightarrow{op} memory$$

subj: instruction + Ctx

op: r/w/x

obj: a set of memory addresses

Why op = r/w/x, why obj = addresses?

→ instruction → memory

mem a fix-sized array (bytes)

mem[idx] → 1B

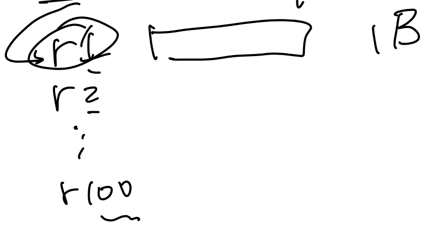
idx → address

①

- a bag of bytes
- random set

infi. seq of bytes

② "named registers"

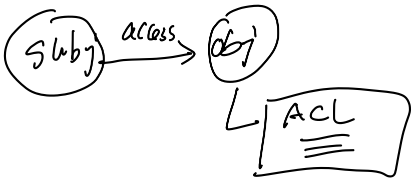


③ (inf set of bytes)

- memory protection: yes/no question

instruction + ctx $\xrightarrow{r/w/x}$ memory addresses ?

- Solution: ACL

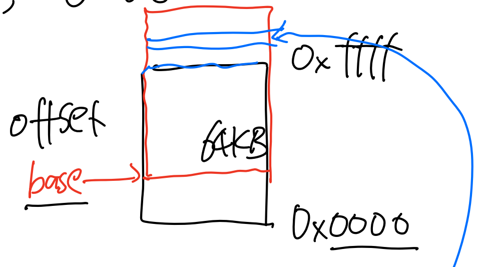


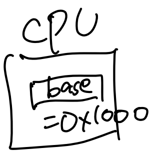
```
Multiple questions:
    Q1: what are memory objs? (memory granularity)
    Q2: who is the subj? (how to define subj)
    Q3: where to store the ACL?
```

```
2. x86 segmentation
```

- history:
  - 1978, 8086 (16-bit) → 2^{16}B = 64KB

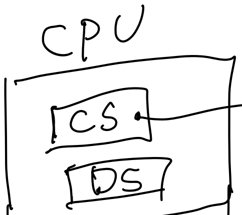
read address:
(base<<4)+offset



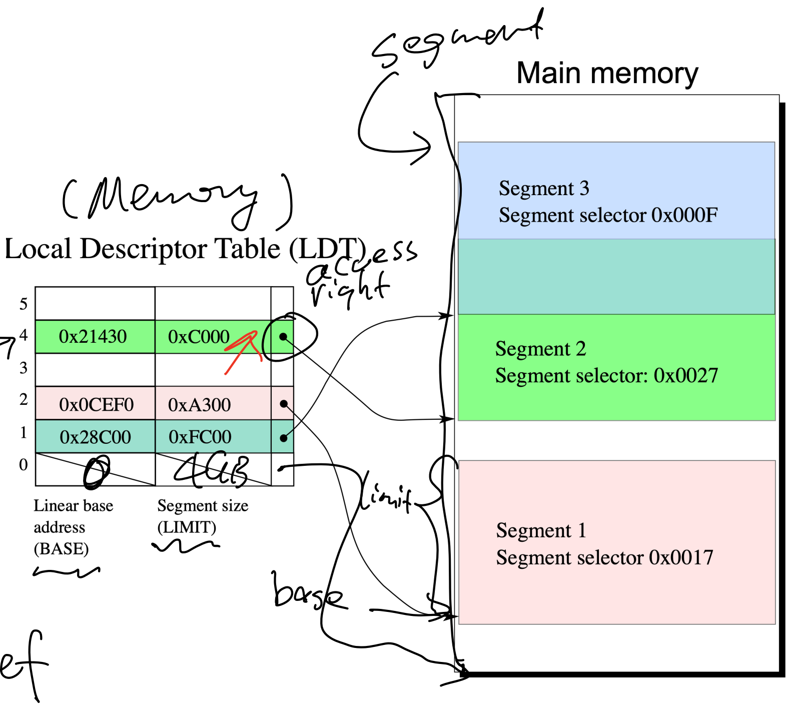


offset=0xf → 0x10000+0xf
=0x1000f

- x86-32 (80386)



ld eax, 0xdeadbeef



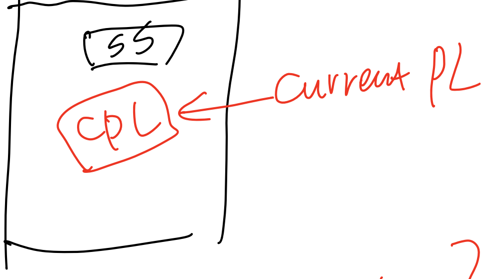
Local Descriptor Table (LDT)

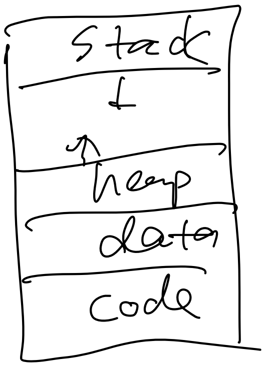
	Linear base address (BASE)	Segment size (LIMIT)
5		
4	0x21430	0xC000
3		
2	0x0CEF0	0xA300
1	0x28C00	0xFC00
0	0	4GB

Segment 3
Segment selector 0x000F

Segment 2
Segment selector: 0x0027

Segment 1
Segment selector 0x0017





- memory protection?
  - check access rights
  - DPL vs. CPL (RPL)

Q1: what is the granularity of memory obj?

Segment (base, limit)

Q2: how to define the subj?

instr + CPL

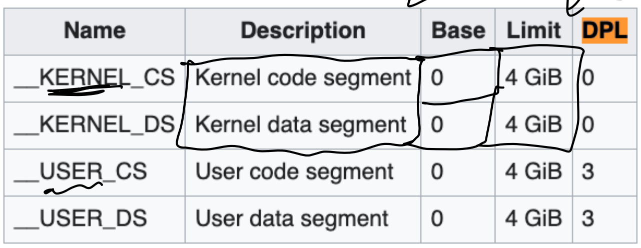
Q3: where to store the ACL?

Memory (descriptor table) + register

Linux

(32bit)

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

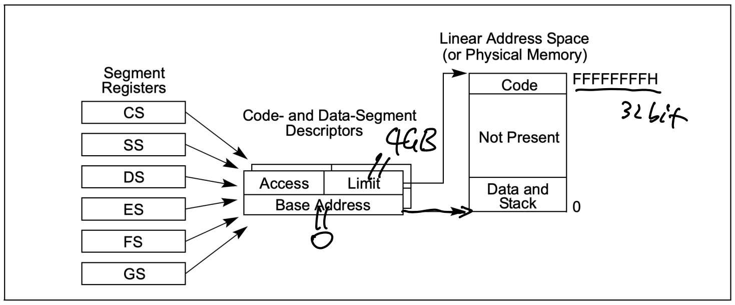


Figure 3-2. Flat Model

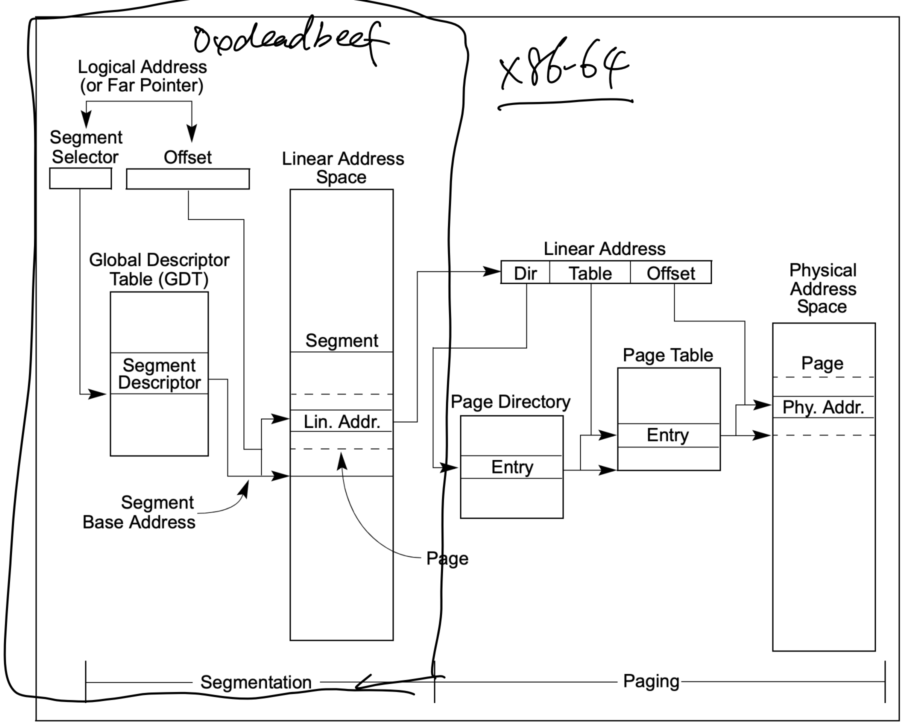
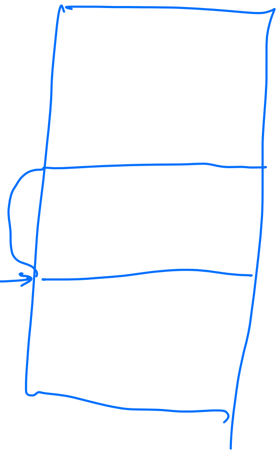
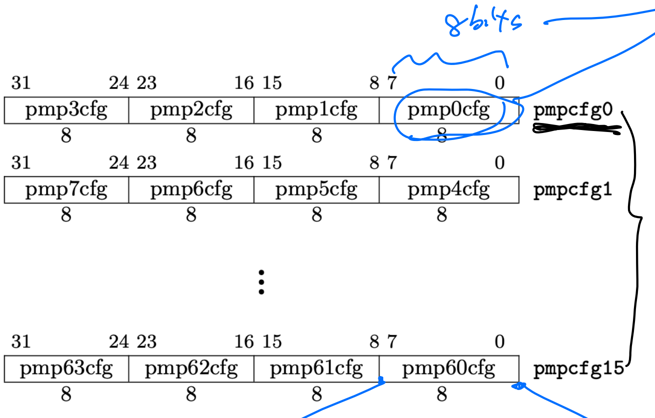


Figure 3-1. Segmentation and Paging

3. PMP (RV-32)

- PMP config registers (16) x4
- PMP addr registers (64)
  - access rights (limit
  - addr





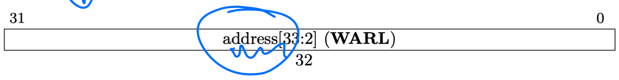


Figure 3.33: PMP address register format, RV32.

Figure 3.31: RV32 PMP configuration CSR layout.

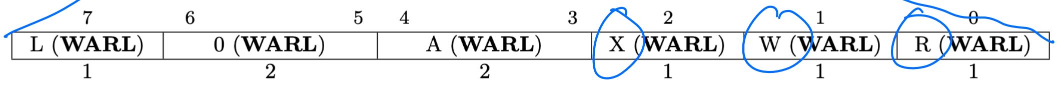


Figure 3.35: PMP configuration register format.

Q1: what is the granularity of memory obj?

"segment" (PMP addr, implicity (limit))

Q2: how to define the subj?

instruction

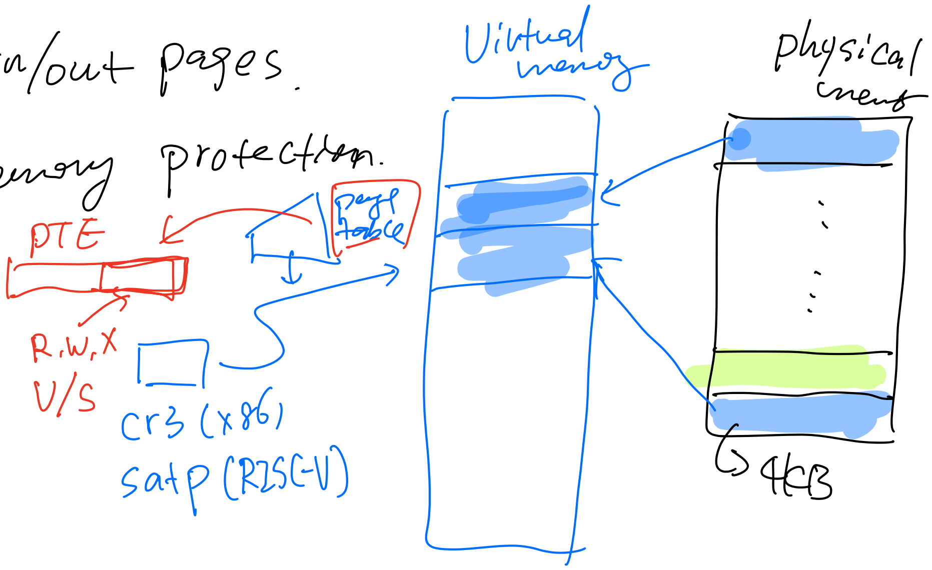
Q3: where to store the ACL?

registers.

4. paging (brief)

- 1962, swap in/out pages.
- later, add memory protection.

→ 0xdeadbeef



Q1: what is the granularity of memory obj?

page (4KB)

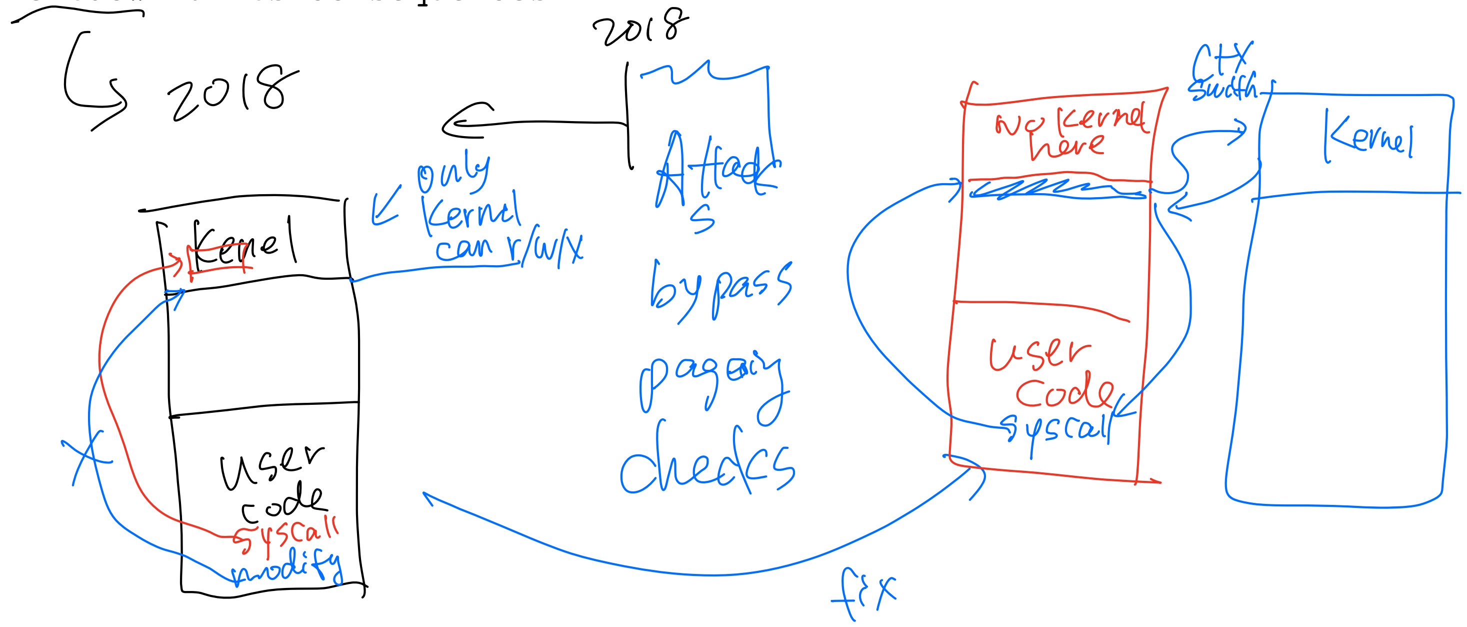
Q2: how to define the subj?

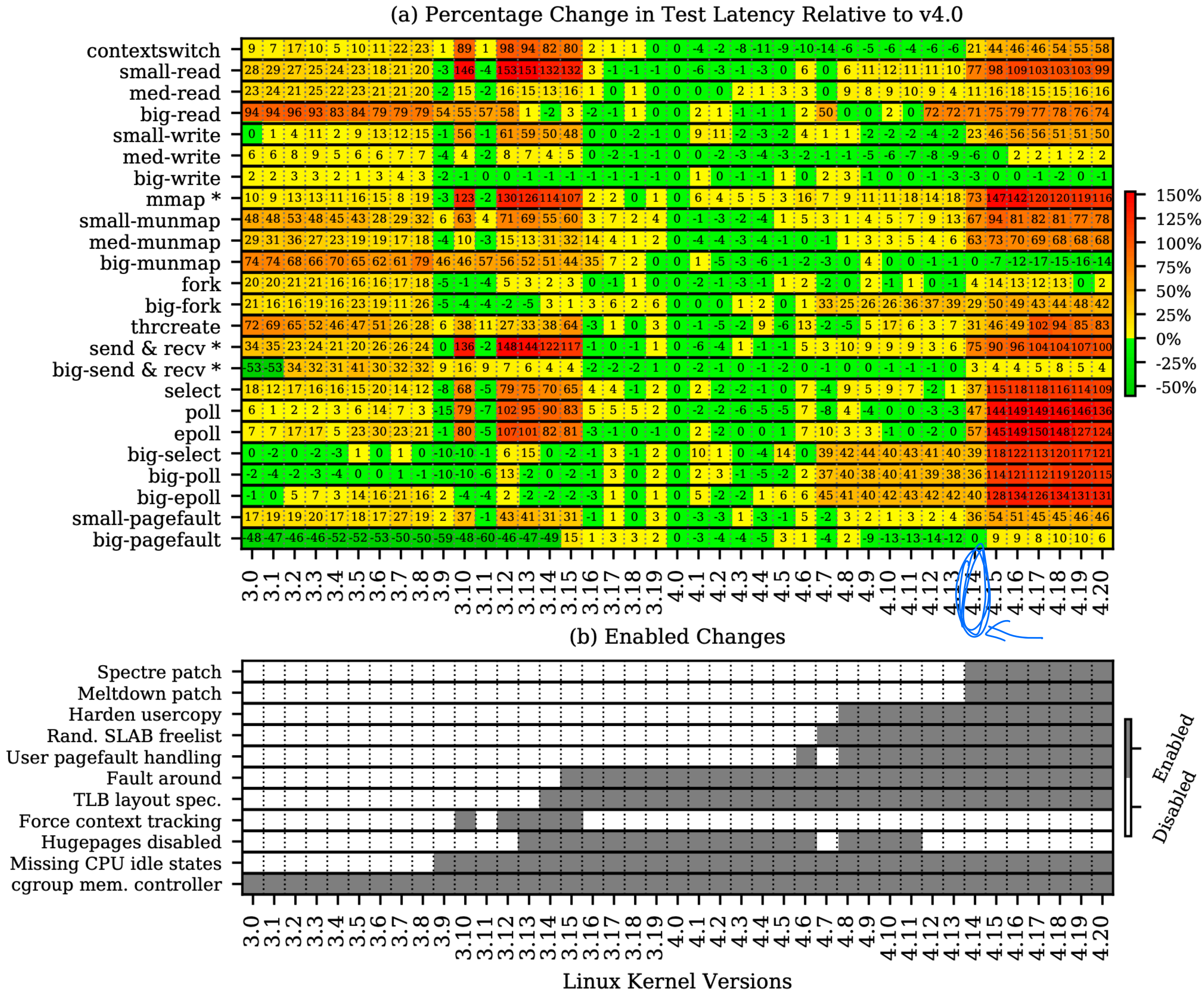
instr + PrivLevel + Cr3/satp

Q3: where to store the ACL?

page table (memory)

5. Meltdown & its consequences





(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
contextswitch	9	7	17	10	5	10	11	22	23	1	89	1	98	94	82	80	2	1	1	0
small-read	28	29	27	25	24	23	18	21	20	-3	146	-4	153	151	132	132	3	-1	-1	-1
med-read	23	24	21	25	22	23	21	21	20	-2	15	-2	16	15	13	16	1	0	1	0
big-read	94	94	96	93	83	84	79	79	79	54	55	57	58	1	-2	3	-2	-1	1	0
small-write	0	1	4	11	2	9	13	12	15	-1	56	-1	61	59	50	48	0	0	-2	-1
med-write	6	6	8	9	5	6	6	7	7	-4	4	-2	8	7	4	5	0	-2	-1	-1
big-write	2	2	3	3	2	1	3	4	3	-2	-1	0	0	-1	-1	-1	-1	-1	-1	-1
mmap *	10	9	13	13	11	16	15	8	19	-3	123	-2	130	126	114	107	2	2	0	1
small-munmap	48	48	53	48	45	43	28	29	32	6	63	4	71	69	55	60	3	7	2	4
med-munmap	29	31	36	27	23	19	19	17	18	-4	10	-3	15	13	31	32	14	4	1	2
big-munmap	74	74	68	66	70	65	62	61	79	46	46	57	56	52	51	44	35	7	2	0
fork	20	20	21	21	16	16	16	17	18	-5	-1	-4	5	3	2	3	0	-1	1	0
big-fork	21	16	16	19	16	23	19	11	26	-5	-4	-4	-2	-5	3	1	3	6	2	6
thrcreate	72	69	65	52	46	47	51	26	28	6	38	11	27	33	38	64	-3	1	0	3
send & recv *	34	35	23	24	21	20	26	26	24	0	136	-2	148	144	122	117	-1	0	-1	1
big-send & recv *	-53	-53	34	32	31	41	30	32	32	9	16	9	7	6	4	4	-2	-2	-2	1
select	18	12	17	16	16	15	20	14	12	-8	68	-5	79	75	70	65	4	4	-1	2
poll	6	1	2	2	3	6	14	7	3	-15	79	-7	102	95	90	83	5	5	5	2
epoll	7	7	17	17	5	23	30	23	21	-1	80	-5	107	101	82	81	-3	-1	0	-1
big-select	0	-2	0	-2	-3	1	0	1	0	-10	-10	-1	6	15	0	-2	-1	3	-1	2
big-poll	-2	-4	-2	-3	-4	0	0	1	-1	-10	-10	-6	13	-2	0	-2	-1	3	0	2
big-epoll	-1	0	5	7	3	14	16	21	16	2	-4	-4	2	-2	-2	-2	-3	1	0	1
small-pagefault	17	19	19	20	17	18	17	27	19	2	37	-1	43	41	31	31	-1	1	0	3
big-pagefault	-48	-47	-46	-46	-52	-52	-53	-50	-50	-59	-48	-60	-46	-47	-49	15	1	3	3	2

	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	0	-4	-2	-8	-11	-9	-10	-14	-6	-5	-6	-4	-6	-6	21	44	46	46	54	55	58
small-read	0	-6	-3	-1	-3	0	6	0	6	11	12	11	11	10	77	98	109	103	103	103	99
med-read	0	0	0	2	1	3	3	0	9	8	9	10	9	4	11	16	18	15	15	16	16
big-read	0	2	1	-1	-1	1	2	50	0	0	2	0	72	72	71	75	79	77	78	76	74
small-write	0	9	11	-2	-3	-2	4	1	1	-2	-2	-2	-4	-2	23	46	56	56	51	51	50
med-write	0	0	-2	-3	-4	-3	-2	-1	-1	-5	-6	-7	-8	-9	-6	0	2	2	1	2	2
big-write	0	1	0	-1	-1	1	0	2	3	-1	0	0	-1	-3	-3	0	0	-1	-2	0	-1
mmap *	0	6	4	5	5	3	16	7	9	11	11	18	14	18	73	147	142	120	120	119	116
small-munmap	0	-1	-3	-2	-4	1	5	3	1	4	5	7	9	13	67	94	81	82	81	77	78
med-munmap	0	-4	-4	-3	-4	-1	0	-1	1	3	3	5	4	6	63	73	70	69	68	68	68
big-munmap	0	1	-5	-3	-6	-1	-2	-3	0	4	0	0	-1	-1	0	-7	-12	-17	-15	-16	-14
fork	0	-2	-1	-3	-1	1	2	-2	0	2	-1	1	0	-1	4	14	13	12	13	0	2
big-fork	0	0	0	1	2	0	1	33	25	26	26	36	37	39	29	50	49	43	44	48	42
thrcreate	0	-1	-5	-2	9	-6	13	-2	-5	5	17	6	3	7	31	46	49	102	94	85	83
send & recv *	0	-6	-4	1	-1	-1	5	3	10	9	9	9	3	6	75	90	96	104	104	107	100
big-send & recv *	0	-2	-1	0	-1	-2	0	-1	0	0	-1	-1	-1	0	3	4	4	5	8	5	4
select	0	2	-2	-2	-1	0	7	-4	9	5	9	7	-2	1	37	115	118	118	116	114	109
poll	0	-2	-2	-6	-5	-5	7	-8	4	-4	0	0	-3	-3	47	144	149	149	146	146	136
epoll	0	2	-2	0	0	1	7	10	3	3	-1	0	-2	0	57	145	149	150	148	127	124
big-select	0	10	1	0	-4	14	0	39	42	44	40	43	41	40	39	118	122	113	120	117	121
big-poll	0	2	3	-1	-5	-2	2	37	40	38	40	41	39	38	36	114	121	112	119	120	115
big-epoll	0	5	-2	-2	1	6	6	45	41	40	42	43	42	42	40	128	134	126	134	131	131
small-pagefault	0	-3	-3	1	-3	-1	5	-2	3	2	1	3	2	4	36	54	51	45	45	46	46
big-pagefault	0	-3	-4	-4	-5	3	1	-4	2	-9	-13	-13	-14	-12	0	9	9	8	10	10	6

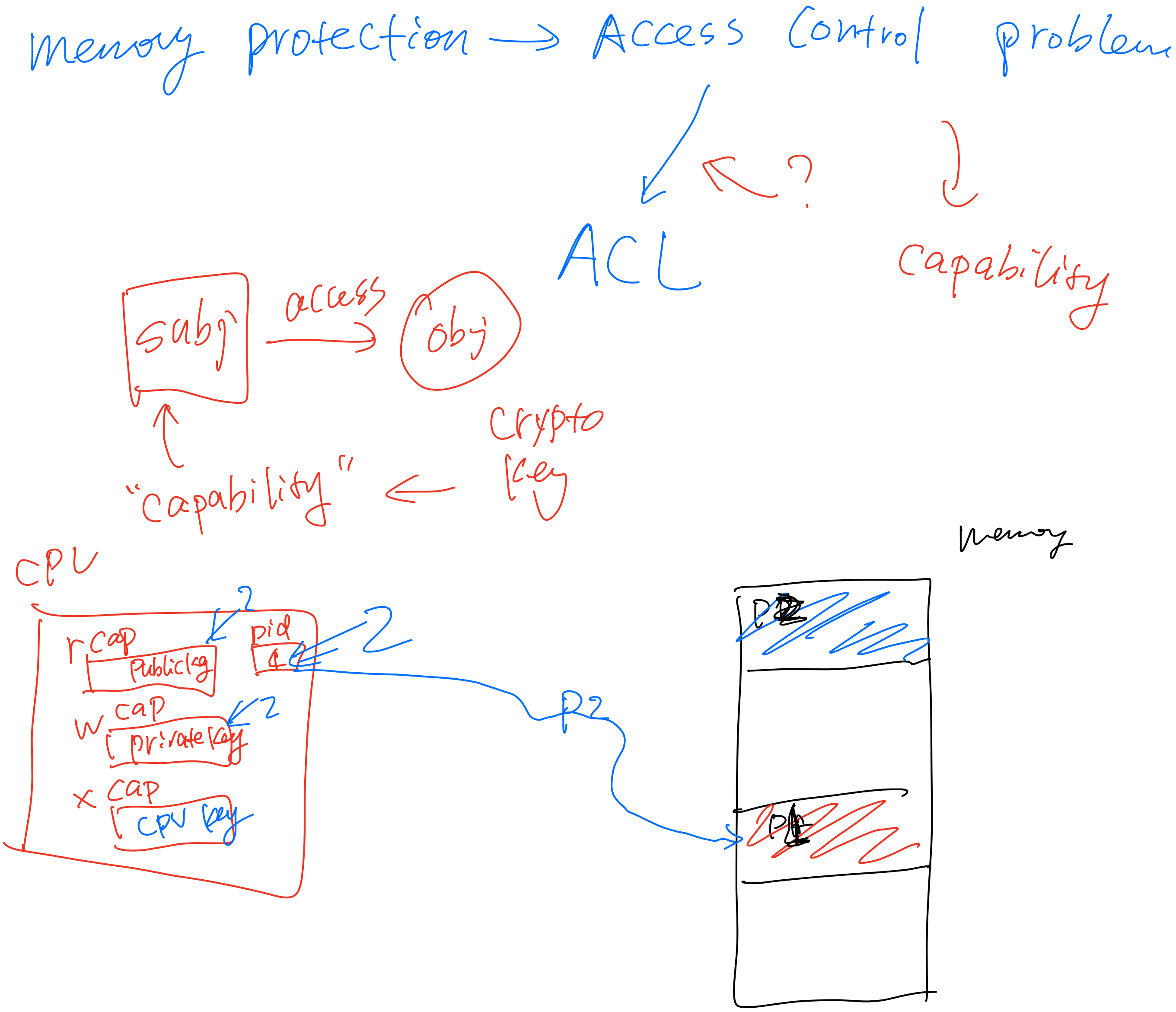
(b) Enabled Changes

	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
Spectre patch	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
Meltdown patch	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
Harden usercopy	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
Rand. SLAB freelist	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
User pagefault handling	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
Fault around	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	■	■	■	■	■
TLB layout spec.	□	□	□	□	□	□	□	□	□	□	□	□	□	□	■	■	■	■	■	■
Force context tracking	□	□	□	□	□	□	□	□	□	□	■	□	■	■	■	■	□	□	□	□
Hugepages disabled	□	□	□	□	□	□	□	□	□	□	□	□	□	■	■	■	■	■	■	■
Missing CPU idle states	□	□	□	□	□	□	□	□	□	■	■	■	■	■	■	■	■	■	■	■
cgroup mem. controller	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

	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
Spectre patch	□	□	□	□	□	□	□	□	□	□	□	□	□	□	■	■	■	■	■	■	■
Meltdown patch	□	□	□	□	□	□	□	□	□	□	□	□	□	□	■	■	■	■	■	■	■
Harden usercopy	□	□	□	□	□	□	□	□	■	■	■	■	■	■	■	■	■	■	■	■	■
Rand. SLAB freelist	□	□	□	□	□	□	□	■	■	■	■	■	■	■	■	■	■	■	■	■	■
User pagefault handling	□	□	□	□	□	□	■	□	■	■	■	■	■	■	■	■	■	■	■	■	■
Fault around	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
TLB layout spec.	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Force context tracking	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
Hugepages disabled	■	■	■	■	■	■	■	□	■	■	■	■	□	□	□	□	□	□	□	□	□
Missing CPU idle states	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
cgroup mem. controller	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

Linux Kernel Versions

6. Capability-based processor (a wild idea)


memory protection
Access Control problem
ACL
?
capability
subj
access
obj
crypto key
"capability"
CPU
r cap
public key
pid
1
2
w cap
private key
x cap
CPU key
P2
memory
P2
P1