Week 6.a
CS6640
02/09 2026
https://naizhengtan.github.io/26spring/

- [ ] 1. CPU privilege levels
- [ ] 2. Memory protection, the problem
- [ ] 3. Protection schemes
- [ ] (A) Segmentaion (x86)
- [ ] (B) PMP (RISC-V)
- [ ] (C) Paging (brief)
- [ ] 4. Meltdown & its consequences
- [ ] 5. Other possible solutions

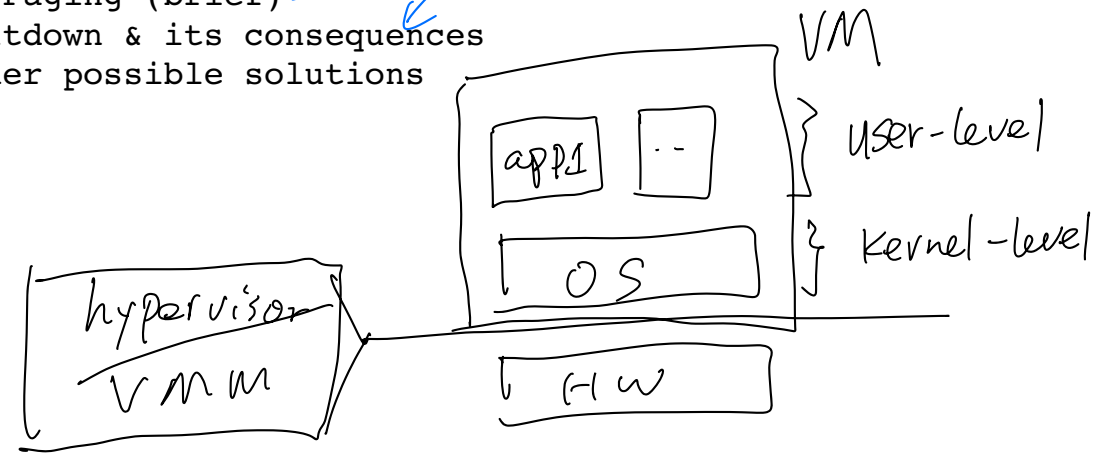
RISC-V:

M/S/U

high → low

low → high : { interrupt, exception, ecall

high → low : mret + mstatus.MPP



Q: motivation: why do people want privilege levels?

- security
- fault isolation
- resource multiplexing
- provide abstraction

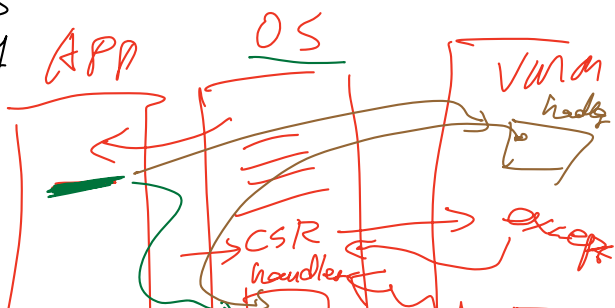
Q: running the same piece of code in unprivileged and privileged mode, what will be the difference?

- CSRs (registers)
- certain memory regions
- instructions

mtvec (CSR)

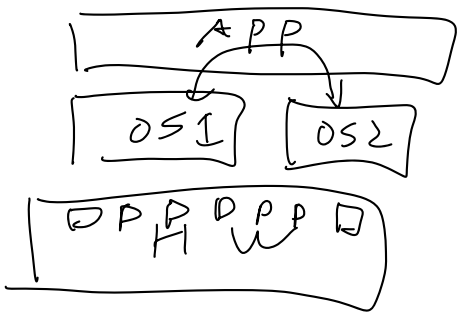
Program
- Code (instructions)
- registers
- memory

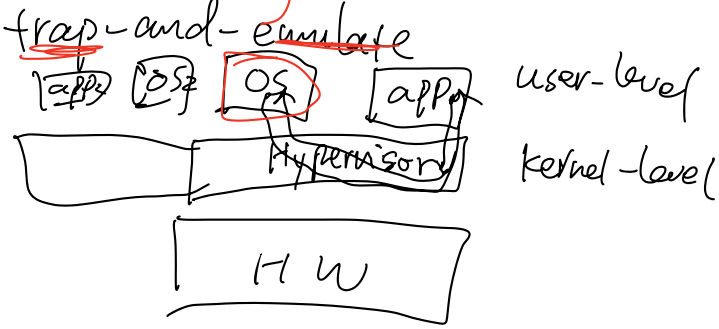
Mechanism : exceptions



Q: How could I know what privilege level the current CPU is running in?

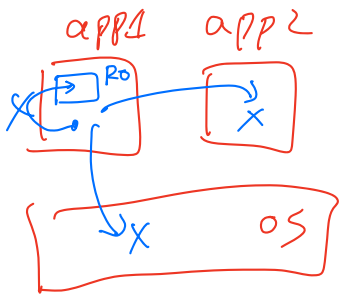
"""
RISC-V deliberately doesn't make it easy for code to discover what mode it is running it because this is a virtualisation hole. As a general principle, code should be designed for and implicitly know what mode it will run in. Applications code should assume it is in U mode. The operating system should assume it is in S mode (it might in fact be virtualised and running in U mode, with things U mode can't do trapped and emulated by the hypervisor).
"""
[from https://forums.sifive.com/t/how-to-determine-the-current-execution-privilege-mode/2823]

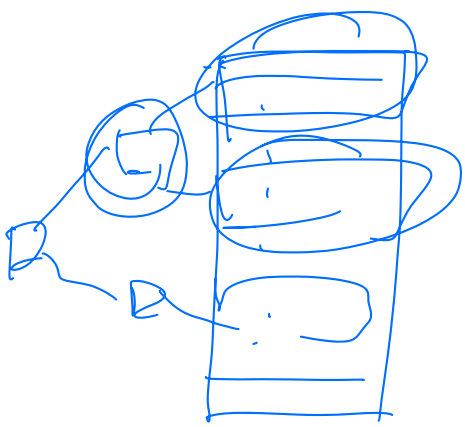




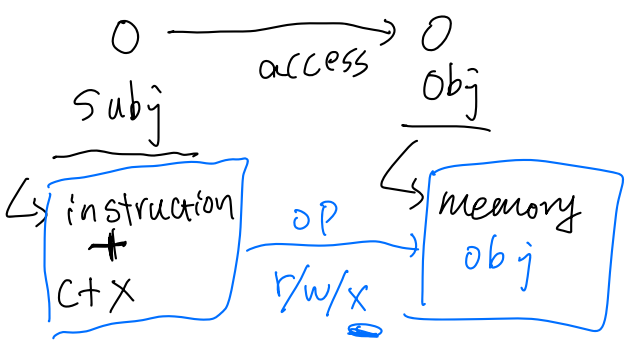
2. Memory protection, the problem

Q: Motivation? why do we need memory protection?





access control:

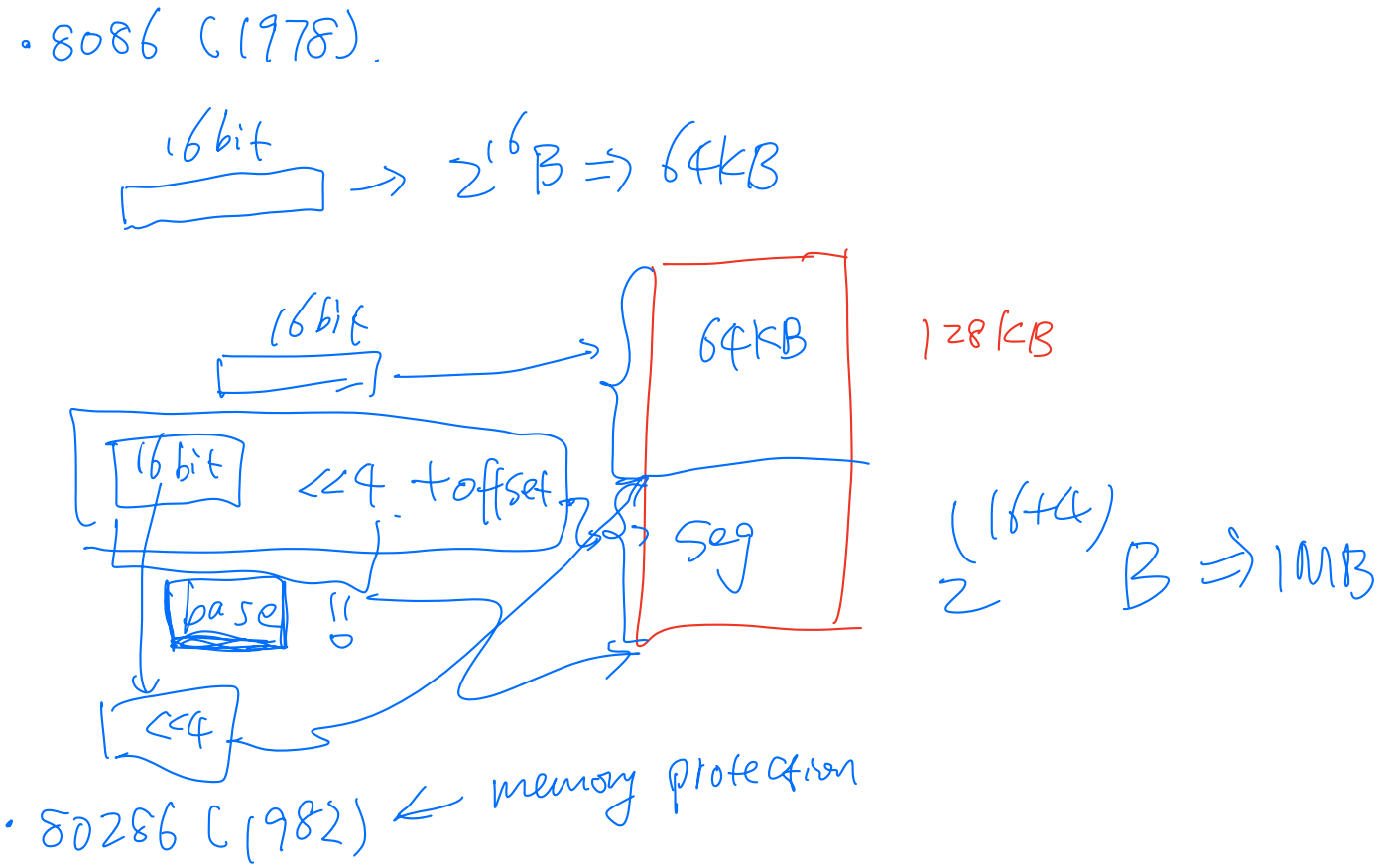


invalid ops:

1. invalid instr
2. invalid op
3. invalid mem

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

(A) segmentation (from x86)



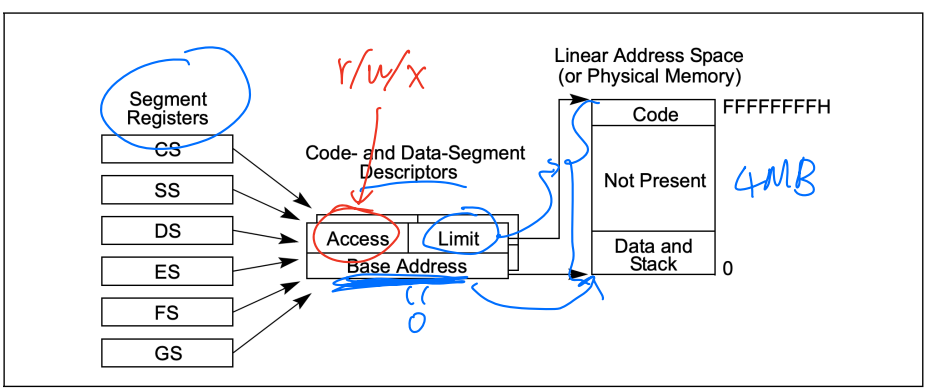
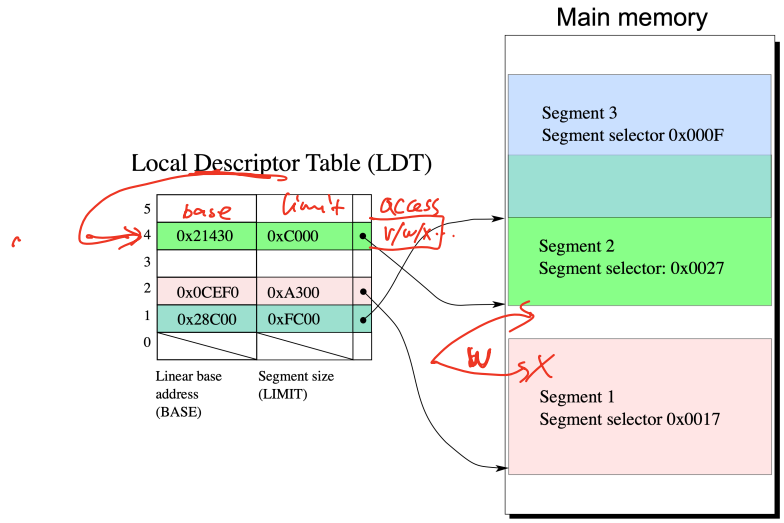


Figure 3-2. Flat Model



Q1: what are memory objs? (memory granularity)

segment ([base, base+limit))

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

instructions + descriptors + priv level

Q3: where to store the ACL?

access bits in descriptors

$2^{32} = 4GB$

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

high

low

ring

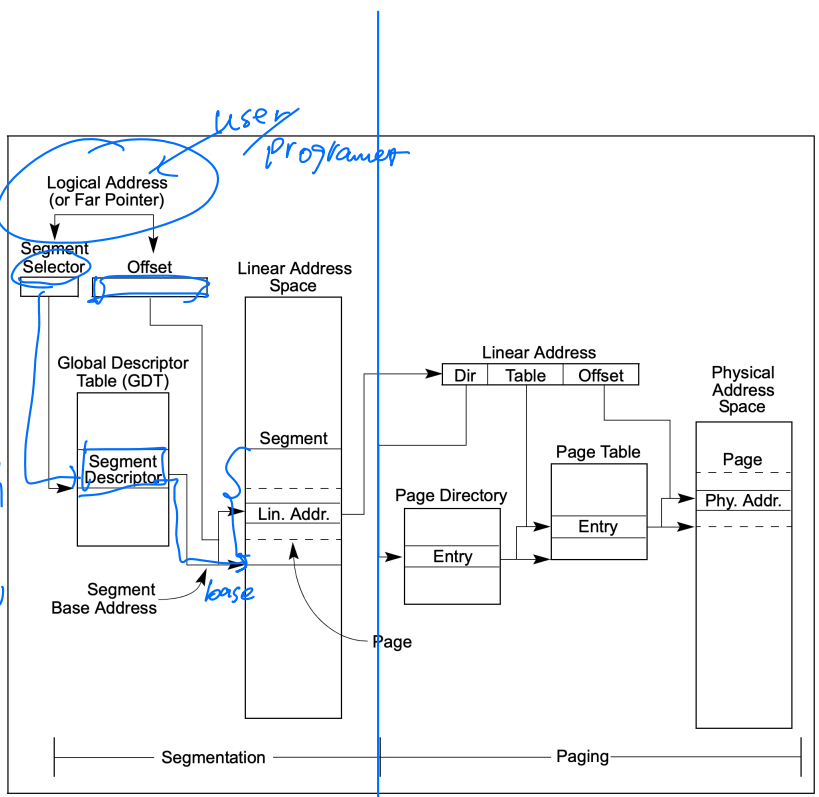


Figure 3-1. Segmentation and Paging

(B) PMP (from RISC-V)

Cheng Tan, OSI

OSI Week06a

Physical Memory Protection (PMP)

(a) pmpaddr[0-63]

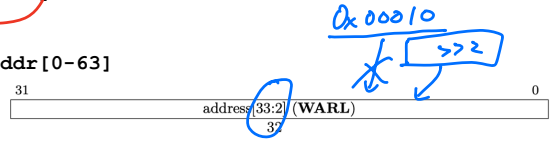


Figure 3.33: PMP address register format, RV32.

(b) pmpcfg[0-15]

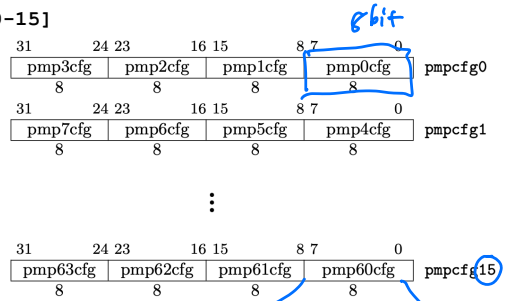


Figure 3.31: RV32 PMP configuration CSR layout.

(c) pmp[0-63]cfg

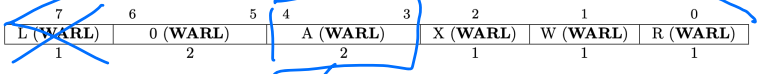
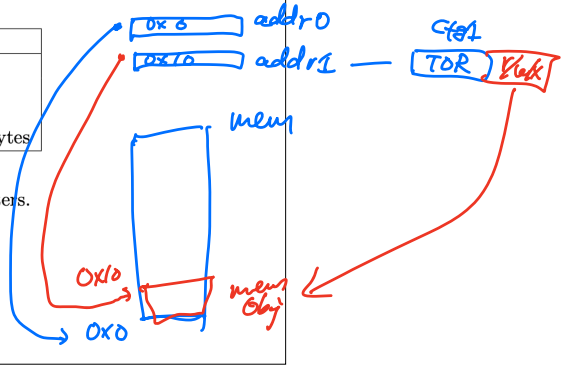


Figure 3.35: PMP configuration register format.

(d) pmp[0-63]cfg.A

A	Name	Description
0	OFF	Null region (disabled)
1	TOR	Top of range
2	NA4	Naturally aligned four-byte region
3	NAPOT	Naturally aligned power-of-two region, ≥8 bytes

Table 3.10: Encoding of A field in PMP configuration registers.



Q1: what are memory objs? (memory granularity)

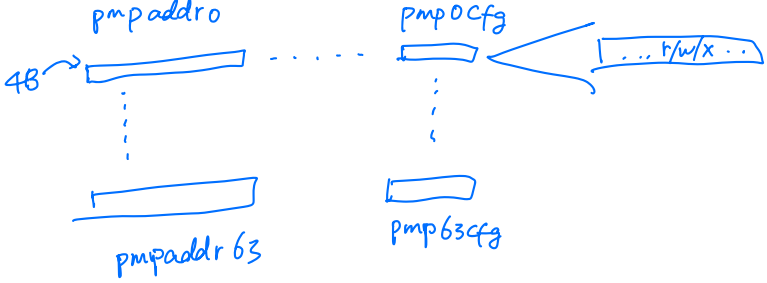
pmpaddr + pmpcfg.A

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

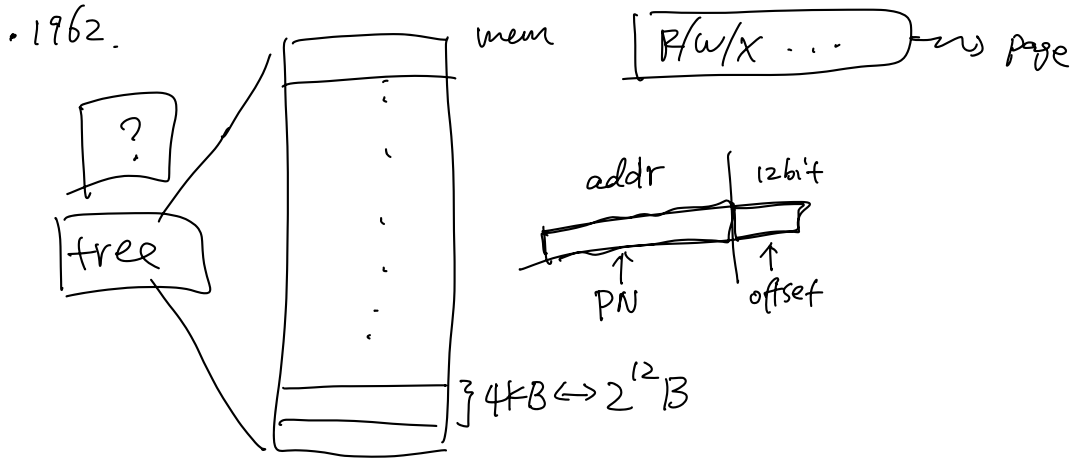
instructions + prv. lv.

Q3: where to store the ACL?

pmpcfg



(C) paging (brief)



Q1: what are memory objs? (memory granularity)

page (4KB)

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

instruct + page table (root: satp) → CSR

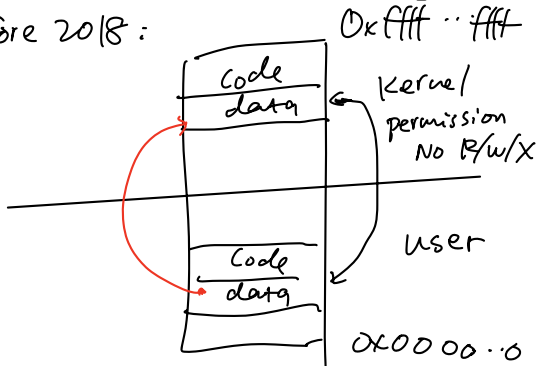
CR3 (x86)

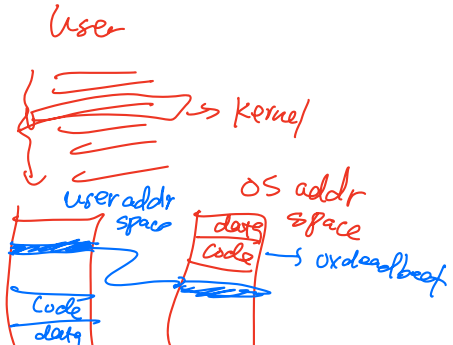
Q3: where to store the ACL?

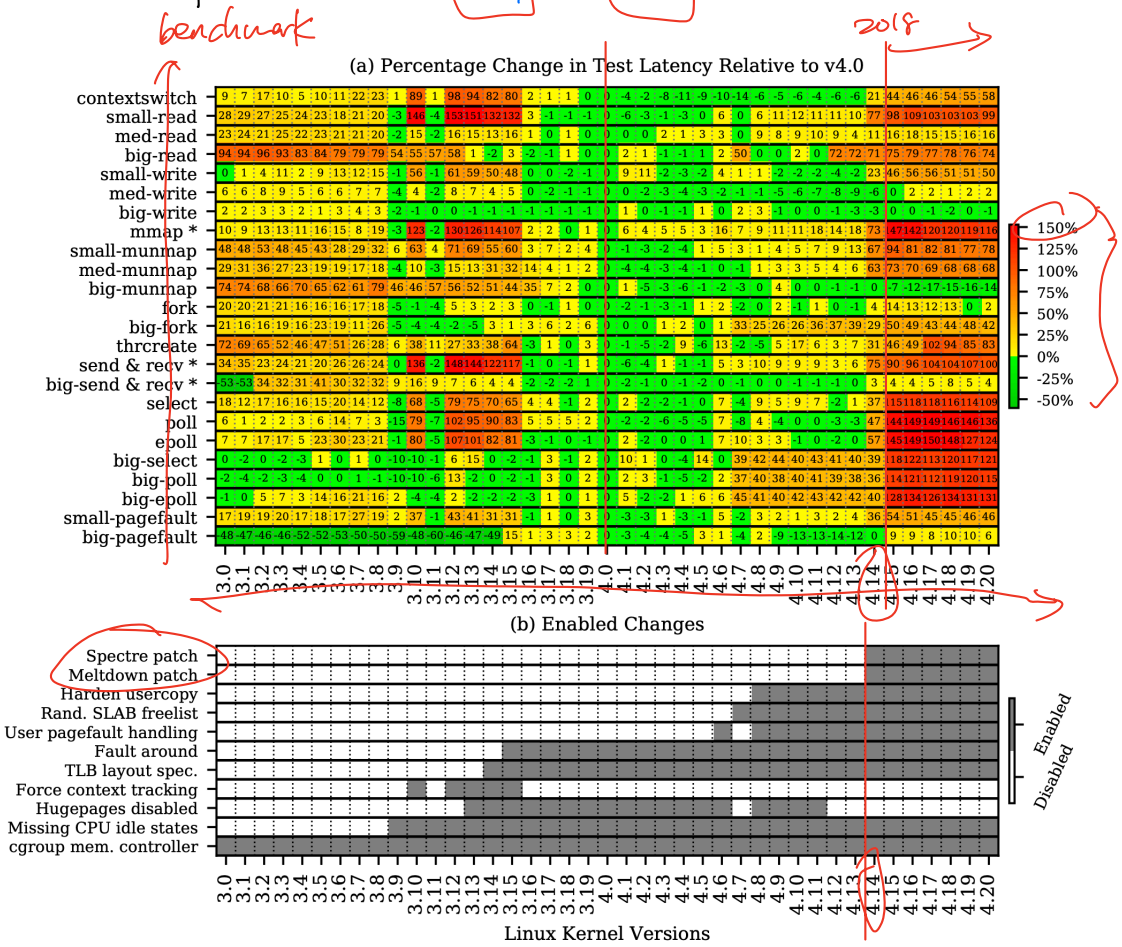
page table entry

4. Meltdown & its consequences

before 2018:







Q: What are the fundamental approaches to provide isolation?