Memoria Virtuale - Protection

Andrea Bartolini – a.bartolini@unibo.it

Derivate da slides Onur Mutlu, Tulio Salmon Cinotti

Memory Protection

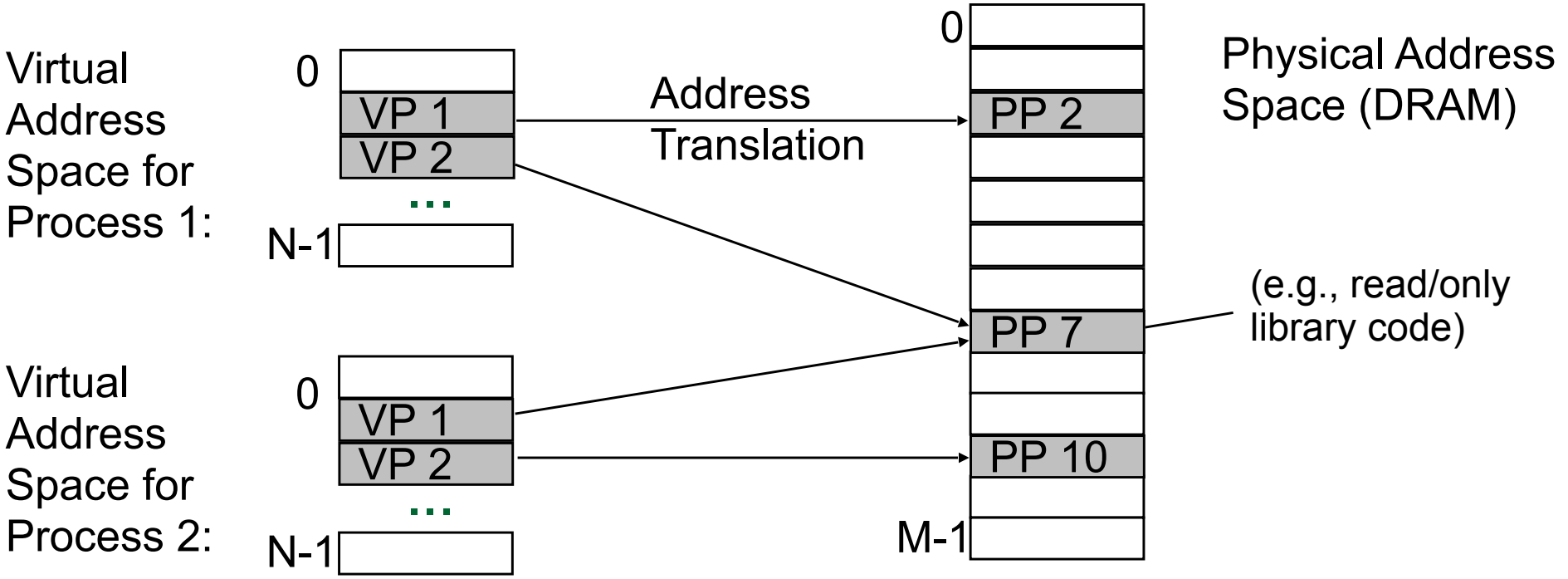
Memory Protection

- Multiple programs (processes) run at once
  - Each process has its own page table
  - Each process can use entire virtual address space without worrying about where other programs are

- A process can only access physical pages mapped in its page table – cannot overwrite memory of another process
  - Provides protection and isolation between processes
  - Enables access control mechanisms per page

Page Table is Per Process

- Each process has its own virtual address space
  - Full address space for each program
  - Simplifies memory allocation, sharing, linking and loading.



Access Protection/Control via Virtual Memory

Page-Level Access Control (Protection)

- Not every process is allowed to access every page
  - E.g., may need supervisor level privilege to access system pages

- Idea: Store access control information on a page basis in the process's page table

- Enforce access control at the same time as translation

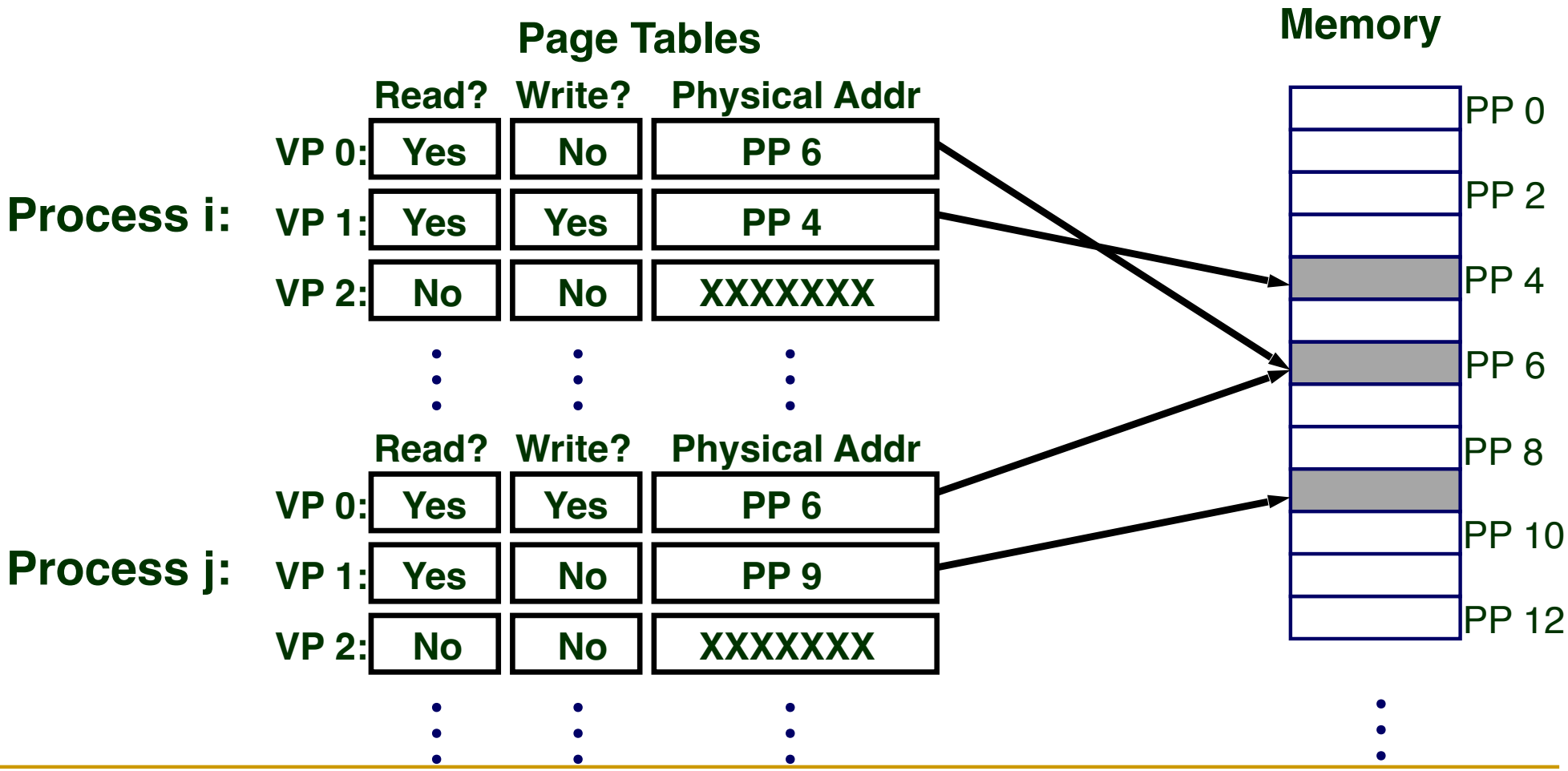
→ Virtual memory system serves two functions today

Address translation (for illusion of large physical memory)

Access control (protection)

VM as a Tool for Memory Access Protection

- Extend Page Table Entries (PTEs) with permission bits
- Check bits on each access and during a page fault
  - If violated, generate exception (Access Protection exception)



Privilege Levels in x86

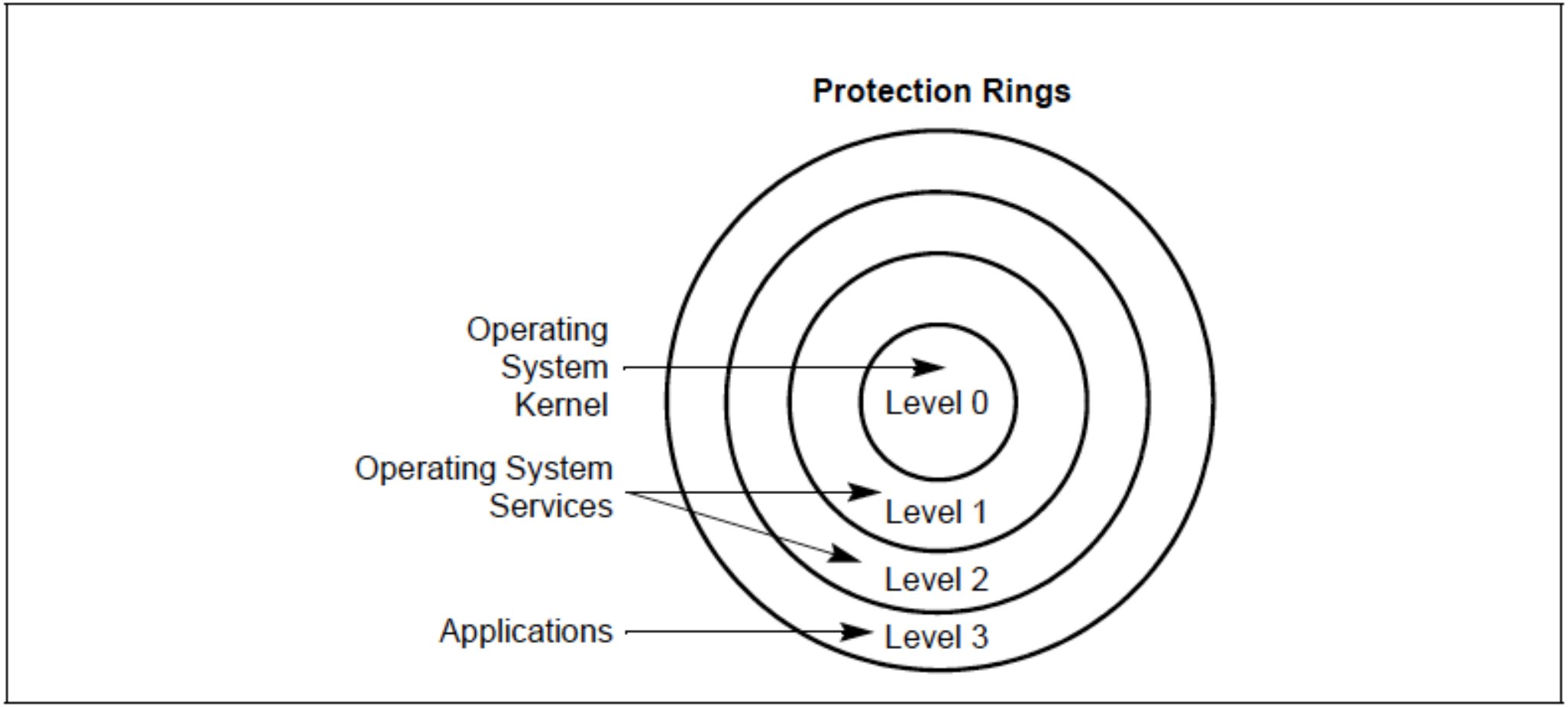


Figure 5-3. Protection Rings

Some Issues in Virtual Memory

Three Major Issues

1. How large is the page table and how do we store and access it?

2. How can we speed up translation & access control check?

3. When do we do the translation in relation to cache access?

- There are many other issues we will not cover in detail
  - What happens on a context switch?
  - How can you handle multiple page sizes?
  - …

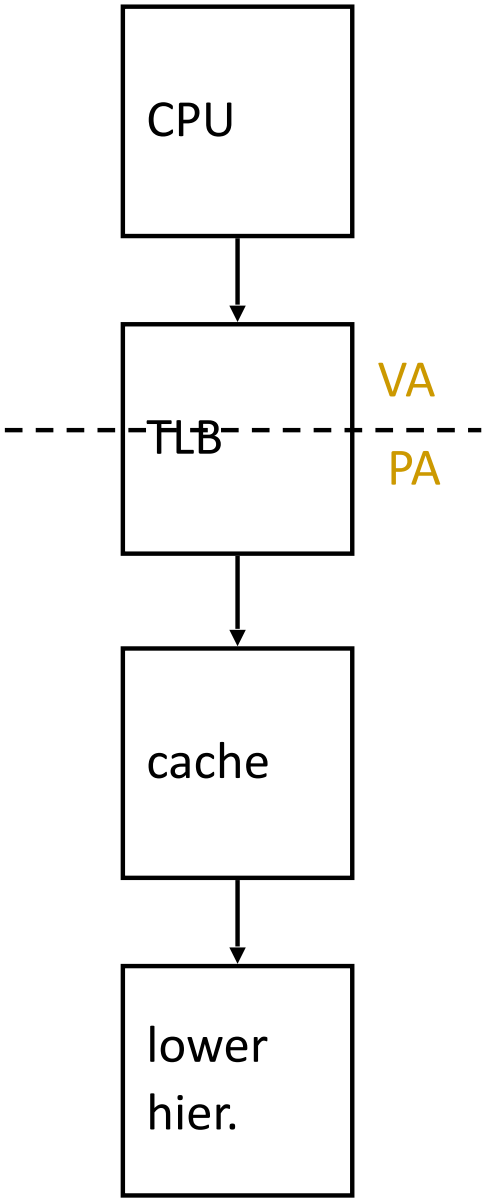
Teaser: Virtual Memory Issue III

- When do we do the address translation?
  - Before or after accessing the L1 cache?

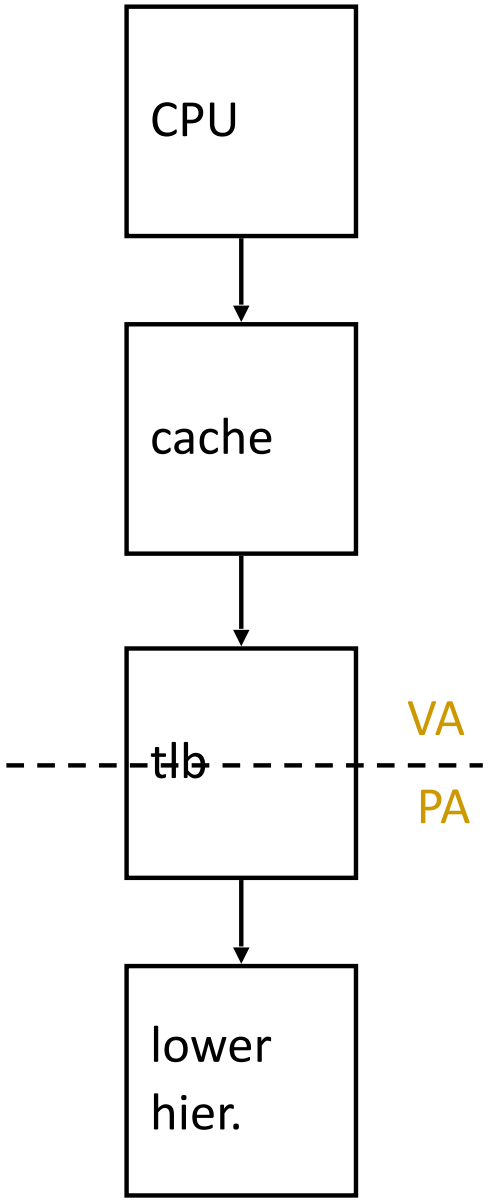
Homonyms and Synonyms

- Homonym: Same VA can map to two different PAs
  - Why?
    - VA is in different processes
- Synonym: Different VAs can map to the same PA
  - Why?
    - Different pages can share the same physical frame within or across processes
    - Reasons: shared libraries, shared data, copy-on-write pages within the same process, …
- Do homonyms and synonyms create problems when we have a cache?
  - Is the cache virtually or physically addressed?

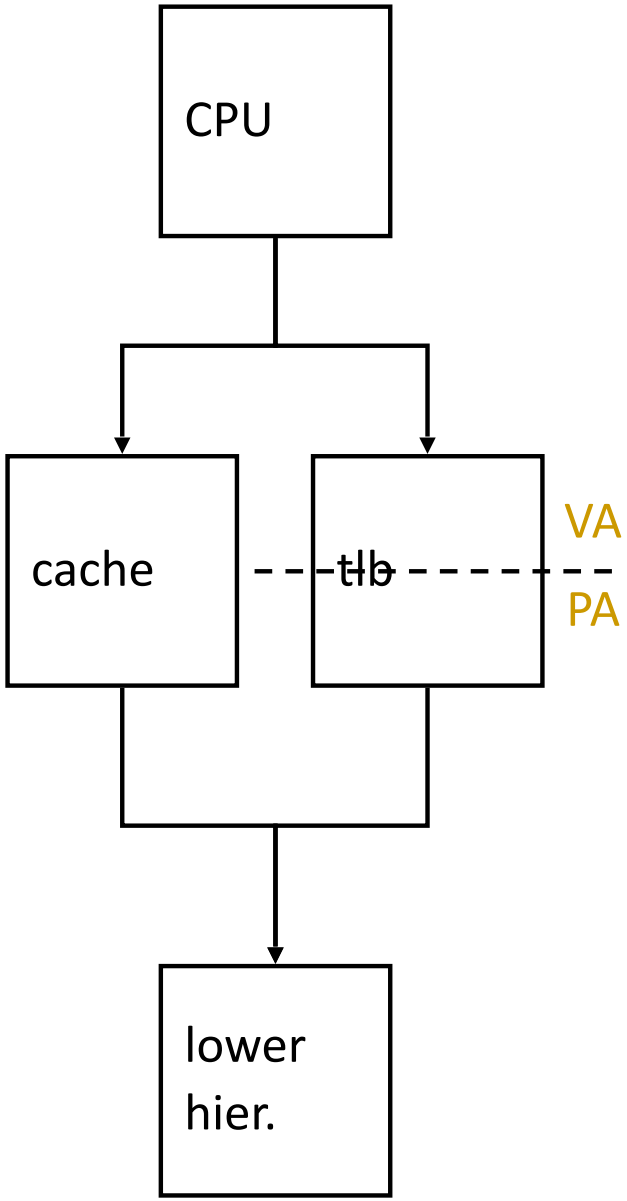
Cache-VM Interaction



physical cache

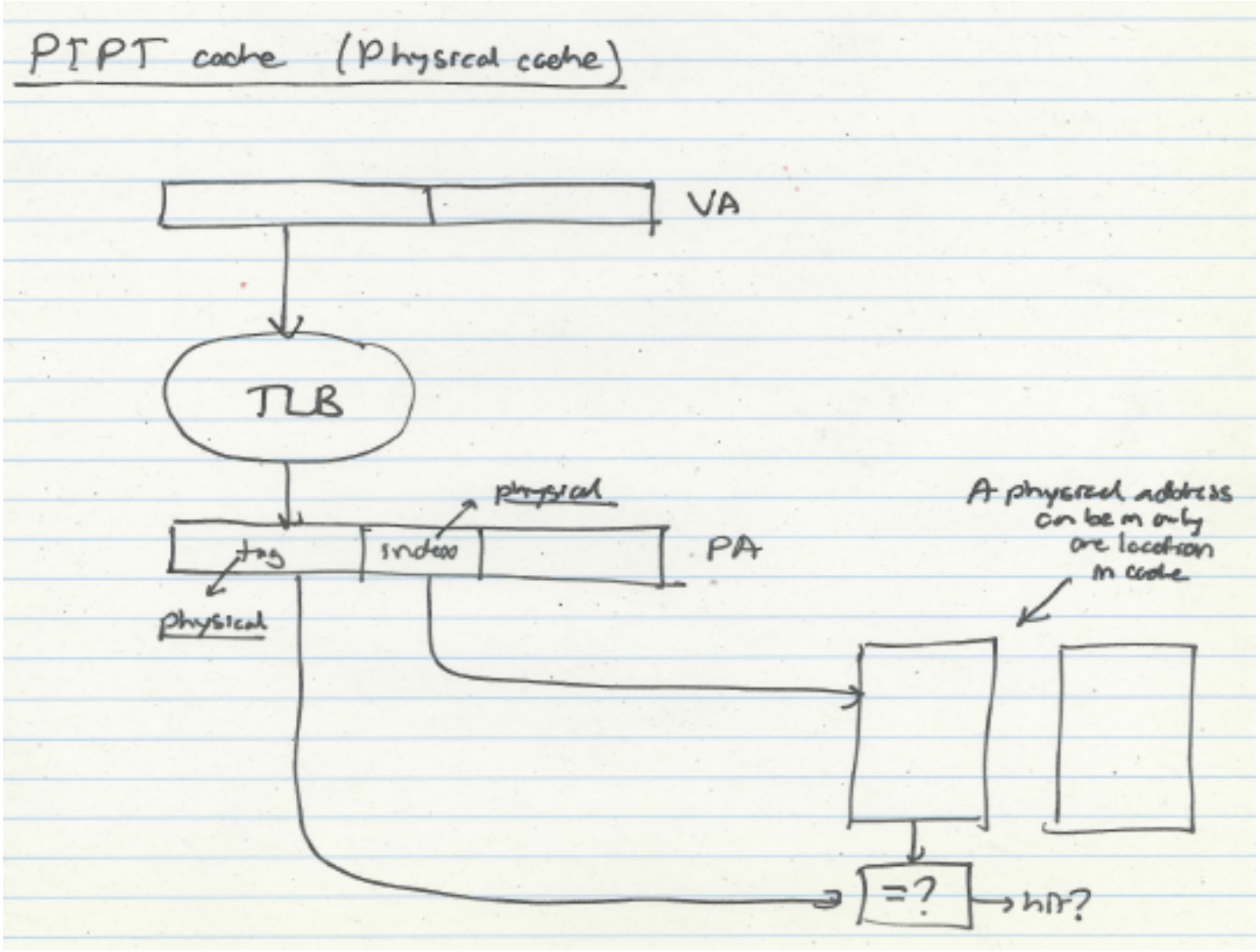


virtual (L1) cache

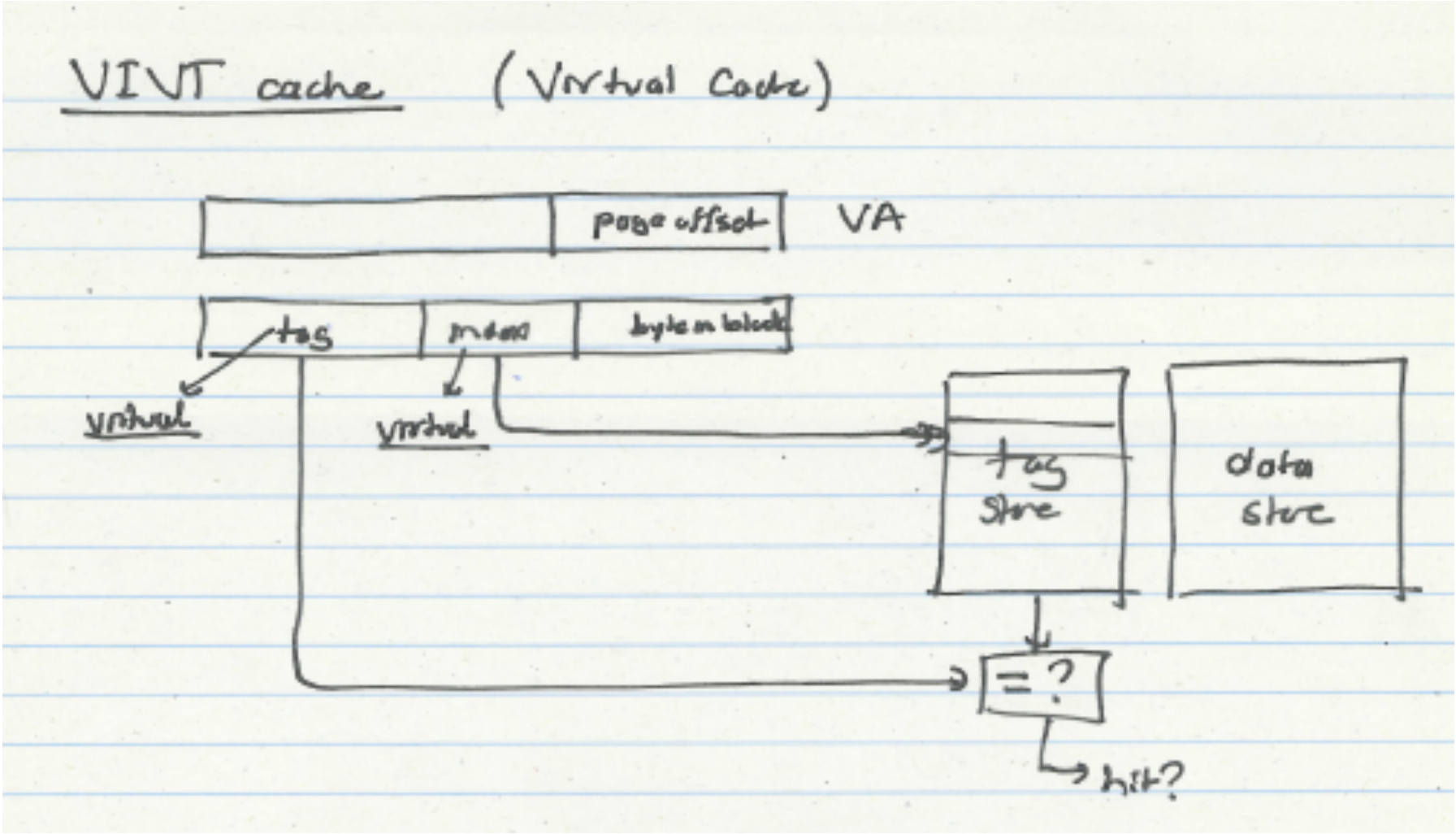


virtual-physical cache

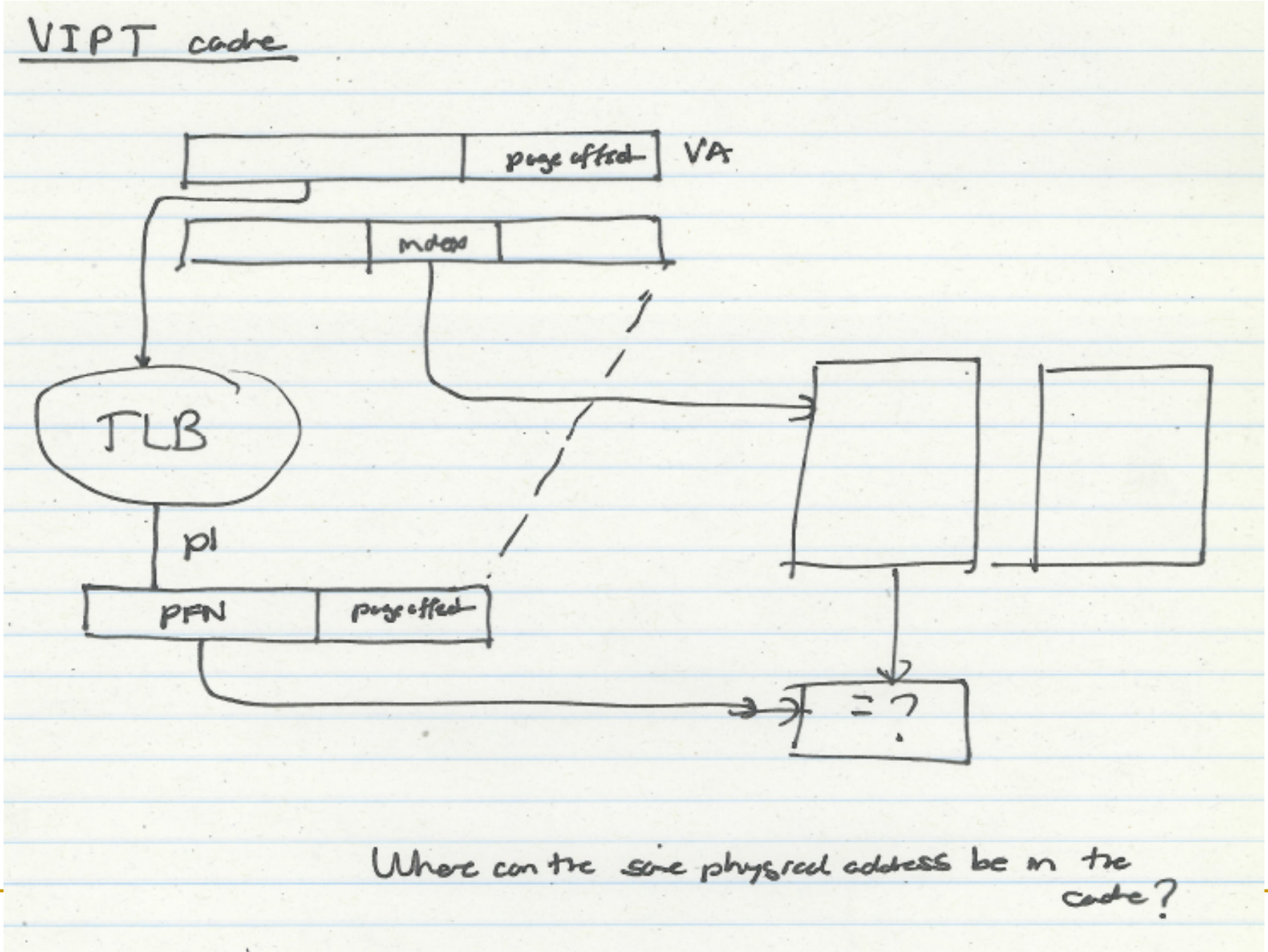
Physical Cache



Virtual Cache

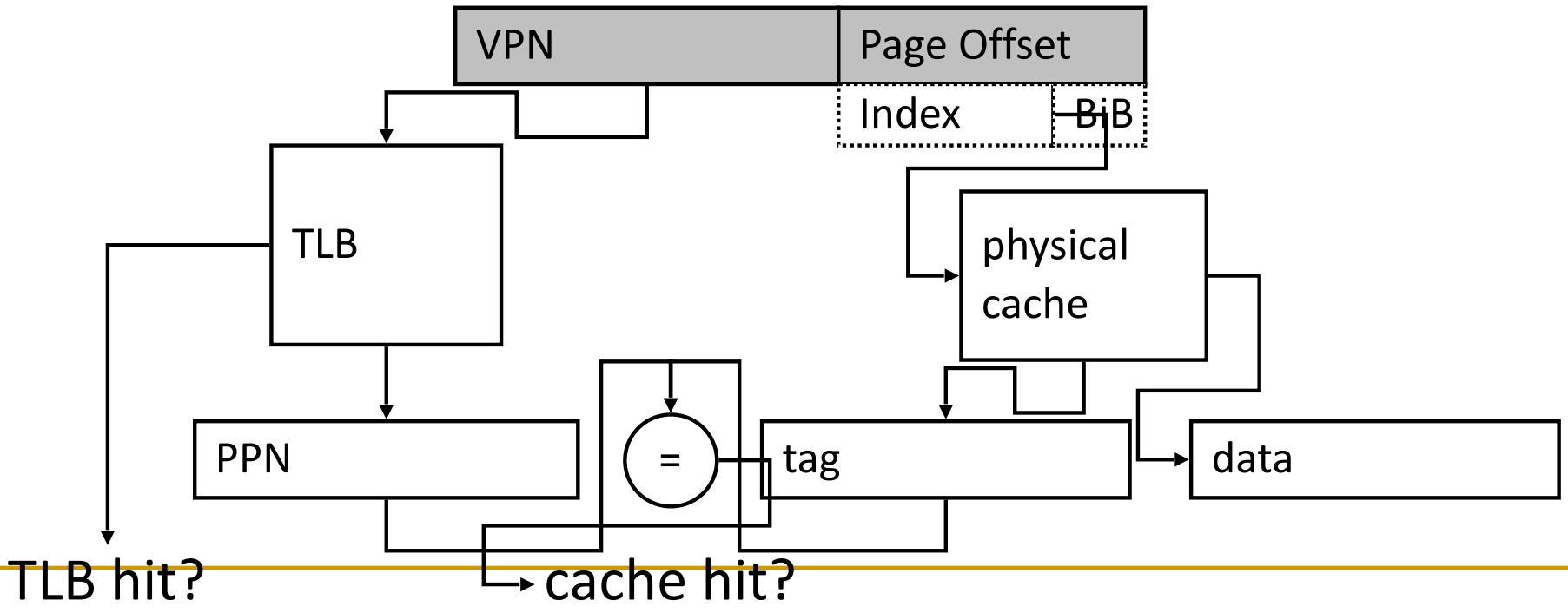


Virtual-Physical Cache



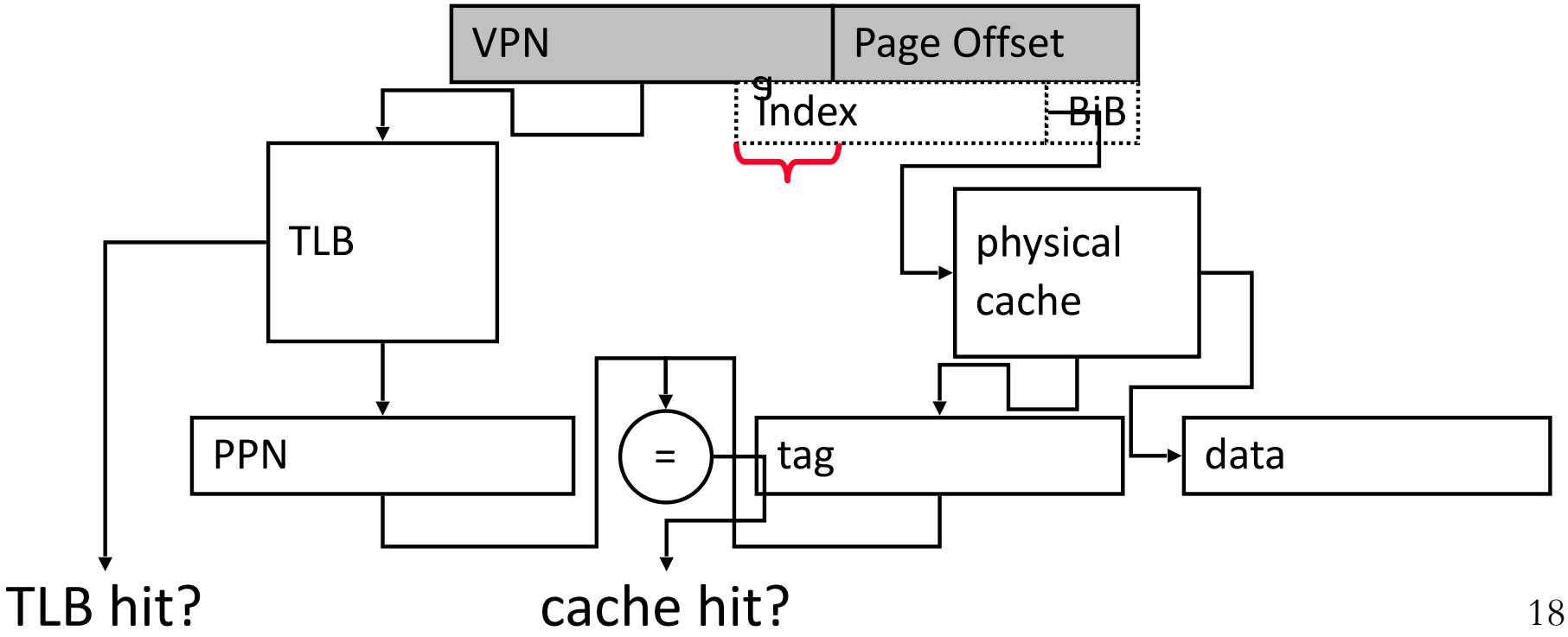
Virtually-Indexed Physically-Tagged

- If C≤(page_size × associativity), the cache index bits come only from page offset (same in VA and PA)
- If both cache and TLB are on chip
  - index both arrays concurrently using VA bits
  - check cache tag (physical) against TLB output at the end



Virtually-Indexed Physically-Tagged

- If C>(page_size × associativity), the cache index bits include VPN
  ⇒ Synonyms can cause problems
  - The same physical address can exist in two locations
- Solutions?



Virtual Memory
Summary

Virtual Memory Summary

- Virtual memory gives the illusion of "infinite" capacity
- A subset of virtual pages are located in physical memory
- A page table maps virtual pages to physical pages – this is called address translation
- A TLB speeds up address translation
- Multi-level page tables keep the page table size in check
- Using different page tables for different programs provides memory protection

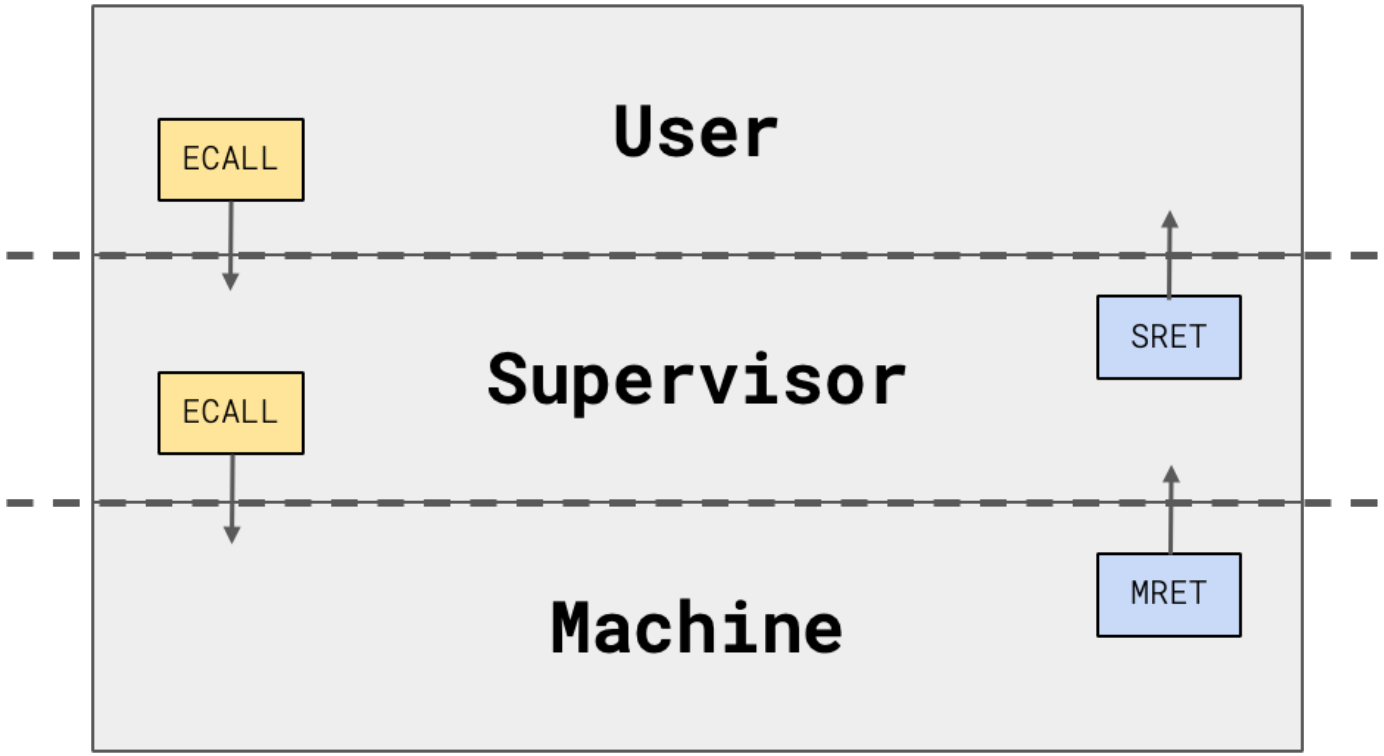
Multilevel On-Chip Caches

Characteristic	ARM Cortex-A53	Intel Core i7
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	Configurable 16 to 64 KiB each for instructions/data	32 KiB each for instructions/data per core
L1 cache associativity	Two-way (I), four-way (D) set associative	Four-way (I), eight-way (D) set associative
L1 replacement	Random	Approximated LRU
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, variable allocation policies (default is Write-allocate)	Write-back, No-write-allocate
L1 hit time (load-use)	Two clock cycles	Four clock cycles, pipelined
L2 cache organization	Unified (instruction and data)	Unified (instruction and data) per core
L2 cache size	128 KiB to 2 MiB	256 KiB (0.25 MiB)
L2 cache associativity	16-way set associative	8-way set associative
L2 replacement	Approximated LRU	Approximated LRU
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L2 hit time	12 clock cycles	10 clock cycles
L3 cache organization	–	Unified (instruction and data)
L3 cache size	–	8 MiB, shared
L3 cache associativity	–	16-way set associative
L3 replacement	–	Approximated LRU
L3 block size	–	64 bytes
L3 write policy	–	Write-back, Write-allocate
L3 hit time	–	35 clock cycles

2-Level TLB Organization

Characteristic	ARM Cortex-A53	Intel Core i7
Virtual address	48 bits	48 bits
Physical address	40 bits	44 bits
Page size	Variable: 4, 16, 64 KiB, 1, 2 MiB, 1 GiB	Variable: 4 KiB, 2/4 MiB
TLB organization	1 TLB for instructions and 1 TLB for data per core Both micro TLBs are fully associative, with 10 entries, round robin replacement 64-entry, four-way set-associative TLBs TLB misses handled in hardware	1 TLB for instructions and 1 TLB for data per core Both L1 TLBs are four-way set associative, LRU replacement L1 I-TLB has 128 entries for small pages, seven per thread for large pages L1 D-TLB has 64 entries for small pages, 32 for large pages The L2 TLB is four-way set associative, LRU replacement The L2 TLB has 512 entries TLB misses handled in hardware

RV Privilege Levels

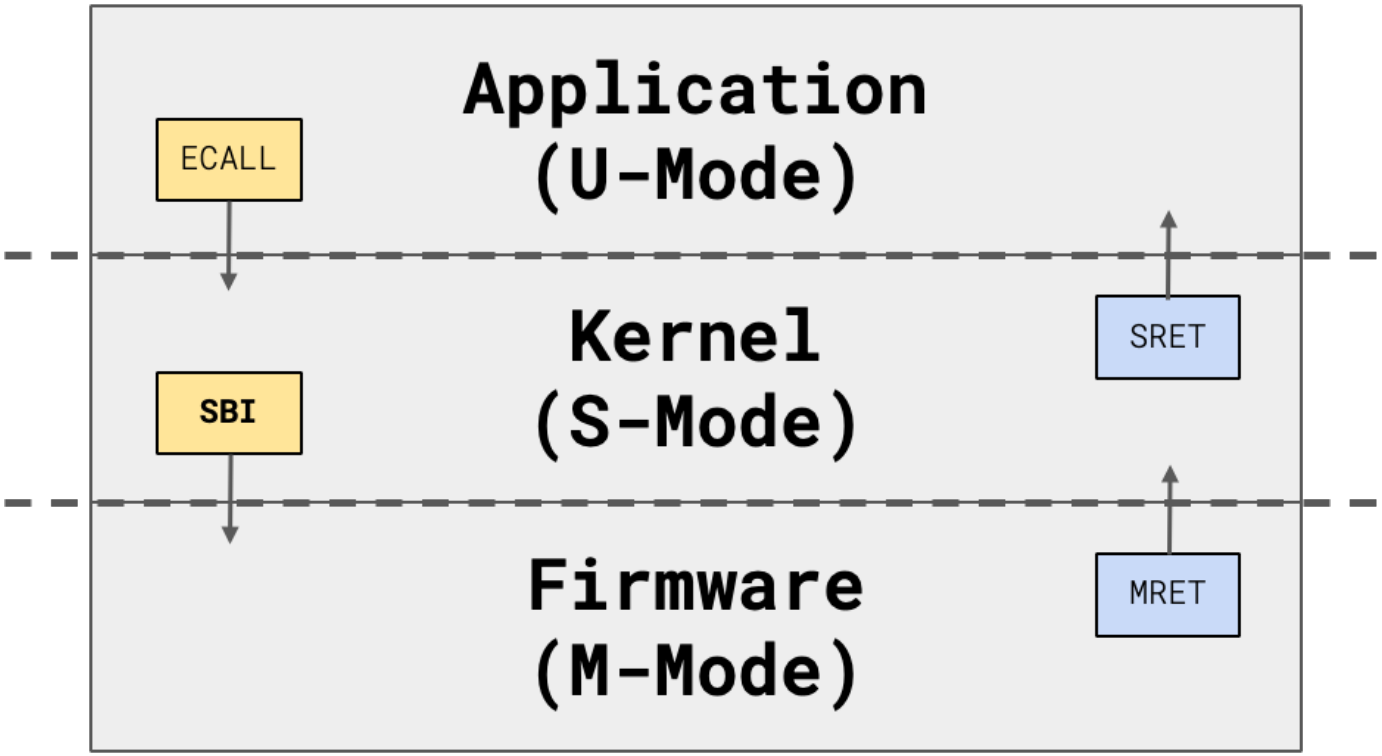


RISC-V Privilege Modes

Level	Encoding	Name	Abbreviation
0	00	User/Application	U
1	01	Supervisor	S
2	10	*Reserved*	
3	11	Machine	M

Table 1.1: RISC-V privilege levels.

RV Privilege Levels

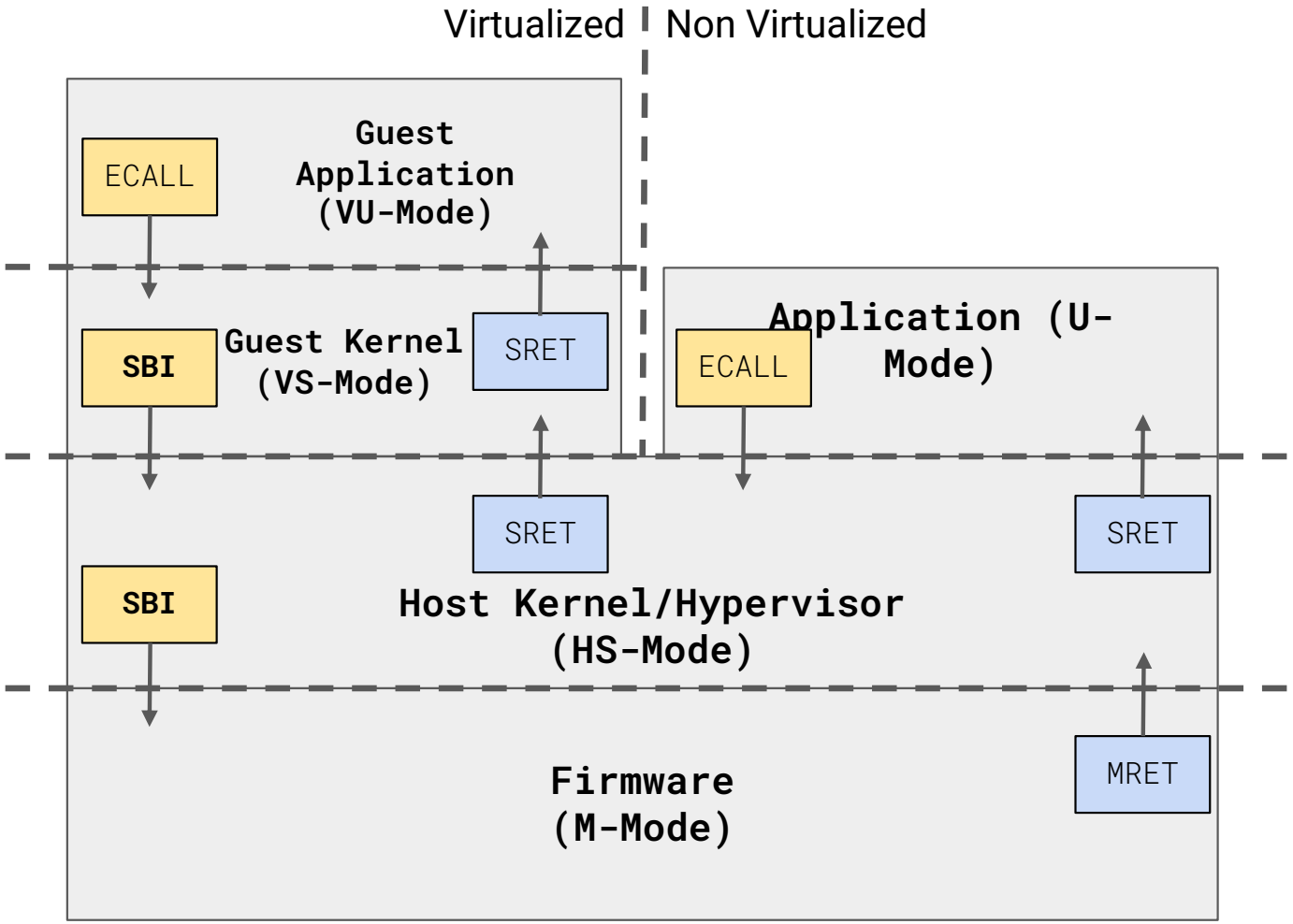


RISC-V Privilege Modes

Number of levels	Supported Modes	Intended Usage
1	M	Simple embedded systems
2	M, U	Secure embedded systems
3	M, S, U	Systems running Unix-like operating systems

Table 1.2: Supported combinations of privilege modes.

RV Privilege Levels



RISC-V Privilege Modes with **Hypervisor Extension**

RV Privilege Levels - CSRs

CSR Address [11:10]	CSR Address [9:8]	CSR Address [7:4]	Hex	Use and Accessibility
Unprivileged and User-Level CSRs				
00	00	XXXX	0x000–0x0FF	Standard read/write
01	00	XXXX	0x400–0x4FF	Standard read/write
10	00	XXXX	0x800–0x8FF	Custom read/write
11	00	0XXX	0xC00–0xC7F	Standard read-only
11	00	10XX	0xC80–0xCBF	Standard read-only
11	00	11XX	0xCC0–0xCFF	Custom read-only
Supervisor-Level CSRs				
00	01	XXXX	0x100–0x1FF	Standard read/write
01	01	0XXX	0x500–0x57F	Standard read/write
01	01	10XX	0x580–0x5BF	Standard read/write
01	01	11XX	0x5C0–0x5FF	Custom read/write
10	01	0XXX	0x900–0x97F	Standard read/write
10	01	10XX	0x980–0x9BF	Standard read/write
10	01	11XX	0x9C0–0x9FF	Custom read/write
11	01	0XXX	0xD00–0xD7F	Standard read-only
11	01	10XX	0xD80–0xDBF	Standard read-only
11	01	11XX	0xDC0–0xDFF	Custom read-only
Hypervisor and VS CSRs				
00	10	XXXX	0x200–0x2FF	Standard read/write
01	10	0XXX	0x600–0x67F	Standard read/write
01	10	10XX	0x680–0x6BF	Standard read/write
01	10	11XX	0x6C0–0x6FF	Custom read/write
10	10	0XXX	0xA00–0xA7F	Standard read/write
10	10	10XX	0xA80–0xABF	Standard read/write
10	10	11XX	0xAC0–0xAFF	Custom read/write
11	10	0XXX	0xE00–0xE7F	Standard read-only
11	10	10XX	0xE80–0xEBF	Standard read-only
11	10	11XX	0xEC0–0xEFF	Custom read-only
Machine-Level CSRs				
00	11	XXXX	0x300–0x3FF	Standard read/write
01	11	0XXX	0x700–0x77F	Standard read/write
01	11	100X	0x780–0x79F	Standard read/write
01	11	1010	0x7A0–0x7AF	Standard read/write debug CSRs
01	11	1011	0x7B0–0x7BF	Debug-mode-only CSRs
01	11	11XX	0x7C0–0x7FF	Custom read/write
10	11	0XXX	0xB00–0xB7F	Standard read/write
10	11	10XX	0xB80–0xBBF	Standard read/write
10	11	11XX	0xBC0–0xBFF	Custom read/write
11	11	0XXX	0xF00–0xF7F	Standard read-only
11	11	10XX	0xF80–0xFBF	Standard read-only
11	11	11XX	0xFC0–0xFFF	Custom read-only

Volume II: RISC-V Privileged Architectures

RV Privilege Levels - CSRs

Number	Privilege	Name	Description
Unprivileged Floating-Point CSRs			
0x001	URW	fflags	Floating-Point Accrued Exceptions.
0x002	URW	frm	Floating-Point Dynamic Rounding Mode.
0x003	URW	fcsr	Floating-Point Control and Status Register (frm + fflags).
Unprivileged Counter/Timers			
0xC00	URO	cycle	Cycle counter for RDCYCLE instruction.
0xC01	URO	time	Timer for RDTIME instruction.
0xC02	URO	instret	Instructions-retired counter for RDINSTRET instruction.
0xC03	URO	hpmcounter3	Performance-monitoring counter.
0xC04	URO	hpmcounter4	Performance-monitoring counter.
		⋮	
0xC1F	URO	hpmcounter31	Performance-monitoring counter.
0xC80	URO	cycleh	Upper 32 bits of cycle, RV32 only.
0xC81	URO	timeh	Upper 32 bits of time, RV32 only.
0xC82	URO	instreth	Upper 32 bits of instret, RV32 only.
0xC83	URO	hpmcounter3h	Upper 32 bits of hpmcounter3, RV32 only.
0xC84	URO	hpmcounter4h	Upper 32 bits of hpmcounter4, RV32 only.
		⋮	
0xC9F	URO	hpmcounter31h	Upper 32 bits of hpmcounter31, RV32 only.

RV Privilege Levels - CSRs

Number	Privilege	Name	Description
Supervisor Trap Setup			
0x100	SRW	sstatus	Supervisor status register.
0x104	SRW	sie	Supervisor interrupt-enable register.
0x105	SRW	stvec	Supervisor trap handler base address.
0x106	SRW	scounteren	Supervisor counter enable.
Supervisor Configuration			
0x10A	SRW	senvcfg	Supervisor environment configuration register.
Supervisor Trap Handling			
0x140	SRW	sscratch	Scratch register for supervisor trap handlers.
0x141	SRW	sepc	Supervisor exception program counter.
0x142	SRW	scause	Supervisor trap cause.
0x143	SRW	stval	Supervisor bad address or instruction.
0x144	SRW	sip	Supervisor interrupt pending.
Supervisor Protection and Translation			
0x180	SRW	satp	Supervisor address translation and protection.
Debug/Trace Registers			
0x5A8	SRW	scontext	Supervisor-mode context register.

RV Privilege Levels - CSRs

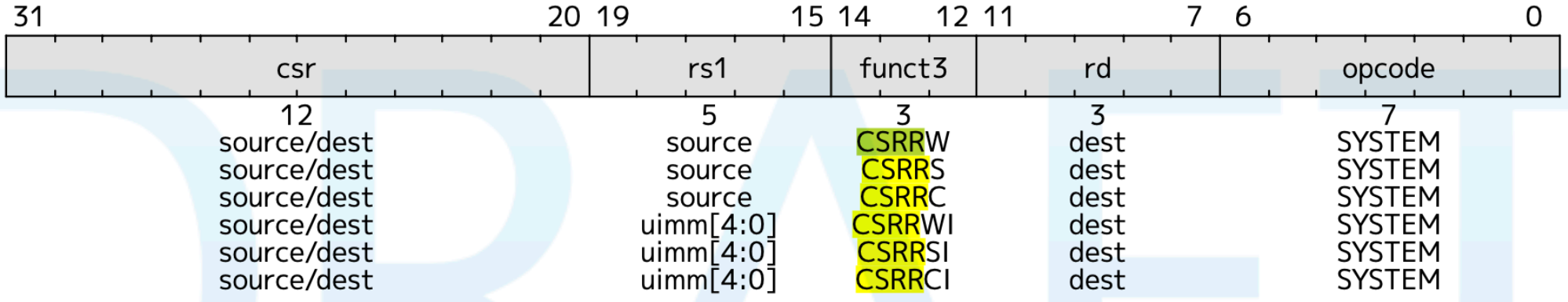
Number	Privilege	Name	Description
		Machine Information Registers	
0xF11	MRO	mvendorid	Vendor ID.
0xF12	MRO	marchid	Architecture ID.
0xF13	MRO	mimpid	Implementation ID.
0xF14	MRO	mhartid	Hardware thread ID.
0xF15	MRO	mconfigptr	Pointer to configuration data structure.
		Machine Trap Setup	
0x300	MRW	mstatus	Machine status register.
0x301	MRW	misa	ISA and extensions
0x302	MRW	medeleg	Machine exception delegation register.
0x303	MRW	mideleg	Machine interrupt delegation register.
0x304	MRW	mie	Machine interrupt-enable register.
0x305	MRW	mtvec	Machine trap-handler base address.
0x306	MRW	mcounteren	Machine counter enable.
0x310	MRW	mstatush	Additional machine status register, RV32 only.
		Machine Trap Handling	
0x340	MRW	mscratch	Scratch register for machine trap handlers.
0x341	MRW	mepc	Machine exception program counter.
0x342	MRW	mcause	Machine trap cause.
0x343	MRW	mtval	Machine bad address or instruction.
0x344	MRW	mip	Machine interrupt pending.
0x34A	MRW	mtinst	Machine trap instruction (transformed).
0x34B	MRW	mtval2	Machine bad guest physical address.
		Machine Configuration	
0x30A	MRW	menvcfg	Machine environment configuration register.
0x31A	MRW	menvcfgh	Additional machine env. conf. register, RV32 only.
0x747	MRW	mseccfg	Machine security configuration register.
0x757	MRW	mseccfgh	Additional machine security conf. register, RV32 only.
		Machine Memory Protection	
0x3A0	MRW	pmpcfg0	Physical memory protection configuration.
0x3A1	MRW	pmpcfg1	Physical memory protection configuration, RV32 only.
0x3A2	MRW	pmpcfg2	Physical memory protection configuration.
0x3A3	MRW	pmpcfg3	Physical memory protection configuration, RV32 only.
		⋮	
0x3AE	MRW	pmpcfg14	Physical memory protection configuration.
0x3AF	MRW	pmpcfg15	Physical memory protection configuration, RV32 only.
0x3B0	MRW	pmpaddr0	Physical memory protection address register.
0x3B1	MRW	pmpaddr1	Physical memory protection address register.
		⋮	
0x3EF	MRW	pmpaddr63	Physical memory protection address register.
		Machine Non-Maskable Interrupt Handling	
0x740	MRW	mnscratch	Resumable NMI scratch register.
0x741	MRW	mnepc	Resumable NMI program counter.
0x742	MRW	mncause	Resumable NMI cause.
0x744	MRW	mnstatus	Resumable NMI status.
		Machine Counter/Timers	
0xB00	MRW	mcycle	Machine cycle counter.
0xB02	MRW	minstret	Machine instructions-retired counter.
0xB03	MRW	mhpmcounter3	Machine performance-monitoring counter.
0xB04	MRW	mhpmcounter4	Machine performance-monitoring counter.
		⋮	
0xB1F	MRW	mhpmcounter31	Machine performance-monitoring counter.
0xB80	MRW	mcycleh	Upper 32 bits of mcycle, RV32 only.
0xB82	MRW	minstreth	Upper 32 bits of minstret, RV32 only.
0xB83	MRW	mhpmcounter3h	Upper 32 bits of mhpmcounter3, RV32 only.
0xB84	MRW	mhpmcounter4h	Upper 32 bits of mhpmcounter4, RV32 only.
		⋮	
0xB9F	MRW	mhpmcounter31h	Upper 32 bits of mhpmcounter31, RV32 only.
		Machine Counter Setup	
0x320	MRW	mcountinhibit	Machine counter-inhibit register.
0x323	MRW	mhpmevent3	Machine performance-monitoring event selector.
0x324	MRW	mhpmevent4	Machine performance-monitoring event selector.
		⋮	
0x33F	MRW	mhpmevent31	Machine performance-monitoring event selector.
		Debug/Trace Registers (shared with Debug Mode)	
0x7A0	MRW	tselect	Debug/Trace trigger register select.
0x7A1	MRW	tdata1	First Debug/Trace trigger data register.
0x7A2	MRW	tdata2	Second Debug/Trace trigger data register.
0x7A3	MRW	tdata3	Third Debug/Trace trigger data register.
0x7A8	MRW	mcontext	Machine-mode context register.
		Debug Mode Registers	
0x7B0	DRW	dcsr	Debug control and status register.
0x7B1	DRW	dpc	Debug program counter.
0x7B2	DRW	dscratch0	Debug scratch register 0.
0x7B3	DRW	dscratch1	Debug scratch register 1.

Table 2.6: Currently allocated RISC-V machine-level CSR addresses.

RV Privilege Levels - CSRs

11.1. CSR Instructions

All CSR instructions atomically read-modify-write a single CSR, whose CSR specifier is encoded in the 12-bit *csr* field of the instruction held in bits 31-20. The immediate forms use a 5-bit zero-extended immediate encoded in the *rs1* field.



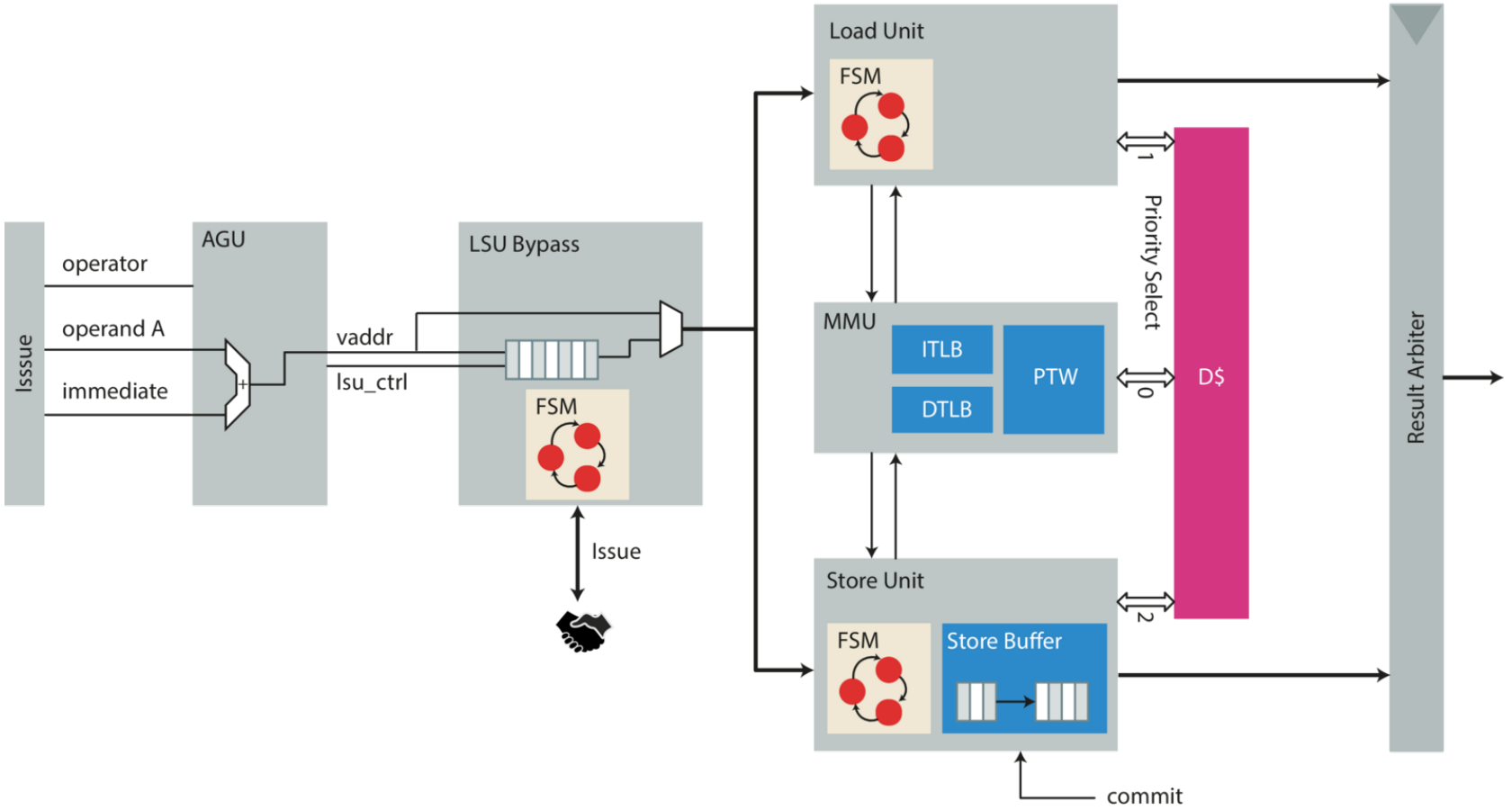
CSRRW (Atomic Read/Write CSR)

CSRRS (Atomic Read and Set Bits in CSR

CSRRC (Atomic Read and Clear Bits in CSR)

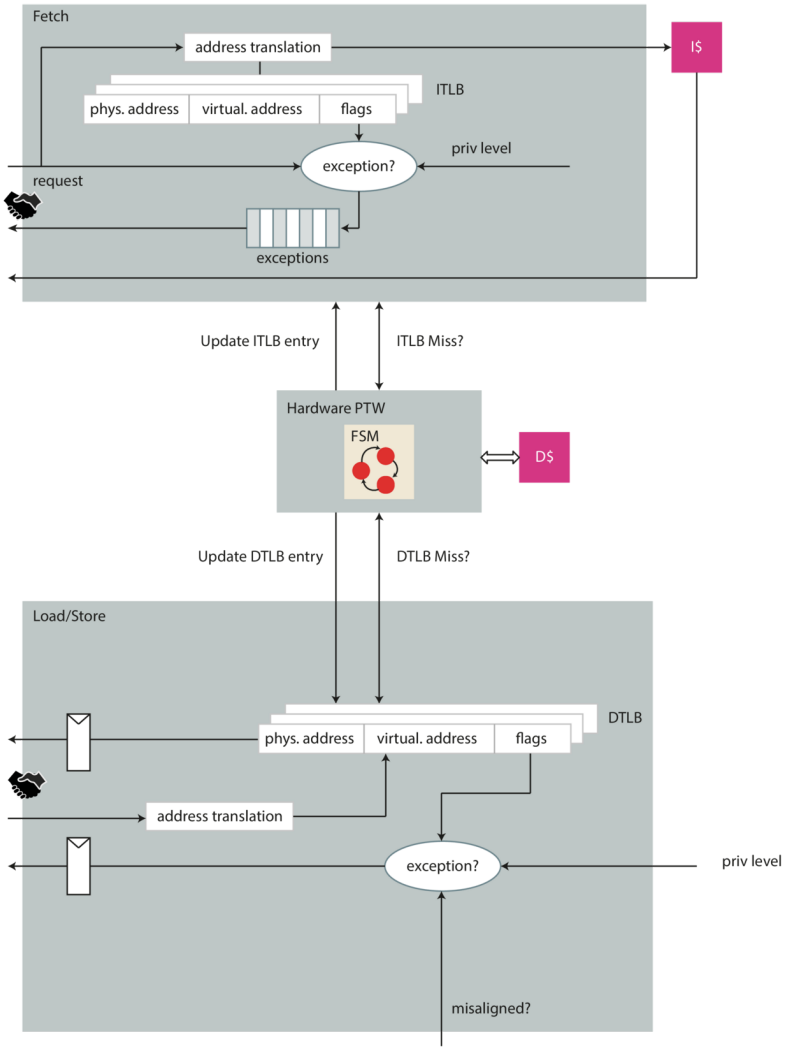
Volume I : The RISC-V Instruction Se

CVA6 - Load and Store Unit



https://cva6.readthedocs.io/en/latest/03_cva6_design/ex_stage.html

CVA6 - MMU



https://cva6.readthedocs.io/en/latest/03_cva6_design/ex_stage.html