Virtual memory and TLBs

What do we know about virtual memory?

- What is it used for?
- What are the advantages?
- How is it implemented?

A means of having a different scheme of addressing (some sort of translation of process view of memory to physical memory)
Virtual address space can be larger than physical address space (not cap) can take advantage of disk storage to help the memory hierarchy
provides isolation between processes
implemented: coordination between OS and HW

Physical memory

The memory physically available on a computer

Models how we've thought about memory so far

Allows random access to bytes using addresses

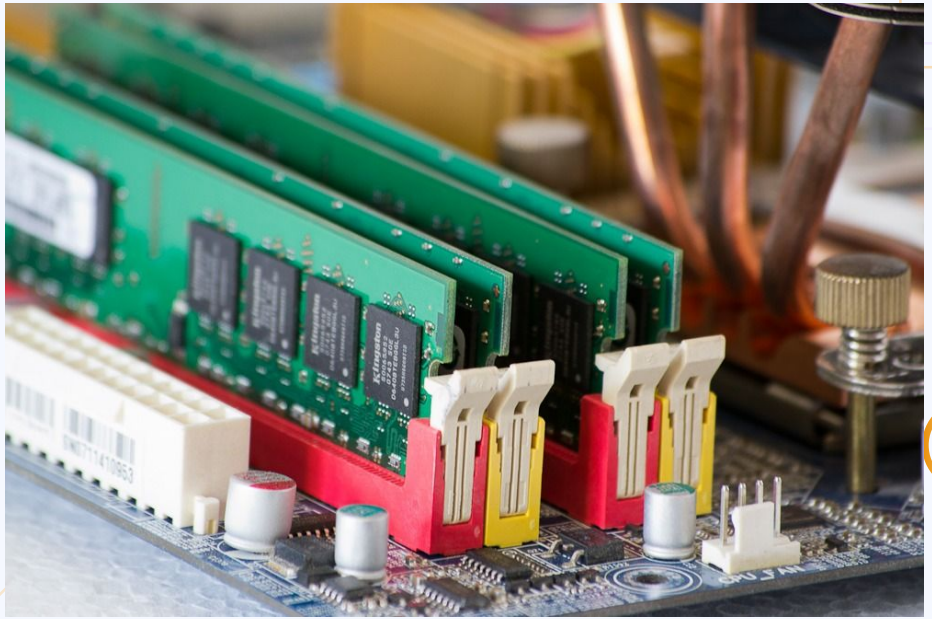


image source

Virtual memory

A way of using main memory as a “cache” for disk storage

Exists in modern OS (managed by SW/HW together)

Advantages:

- Allows coordination of memory between processes (less complexity from process POV)
- Allows process to see larger address space than fits in main memory

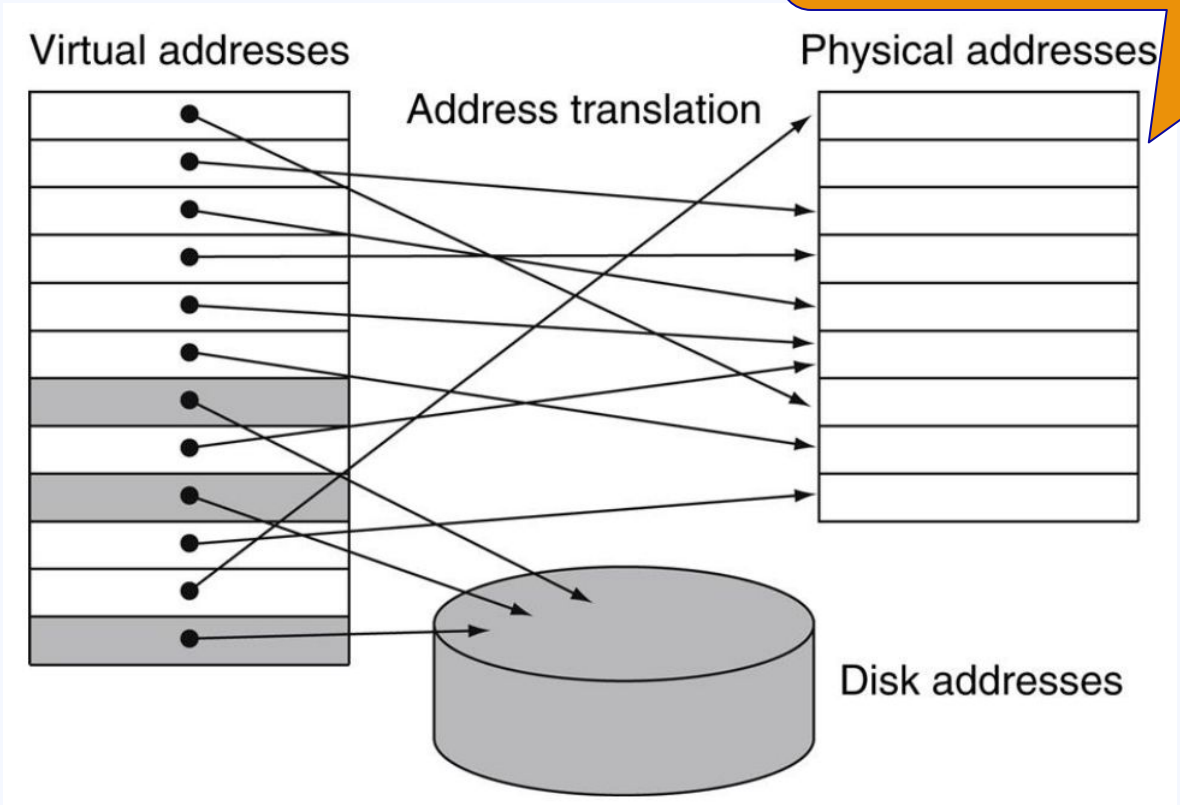
Complexities:

- Managing the mapping of physical to virtual memory
- Slow speed of disk

P&H Fig. 5.25

Each block of virtual memory is called a “page” (4KB on Intel chips, 16KB on Apple silicon)

Each process can have its own view of virtual addresses (0-N)



What determines how many physical pages our system has? How many virtual pages?

Avoiding page faults

A virtual memory “miss” is called a **page fault**

Since disk is even slower (100k* slowdown) than main memory, we want:

- Fairly large page sizes
- Fully associative placement of pages in memory (virtual page can map to any physical page)

Does it make sense for virtual memory systems to use write-through or write-back?

If virtual addresses can map to *any* physical address, how do we efficiently find a physical page?

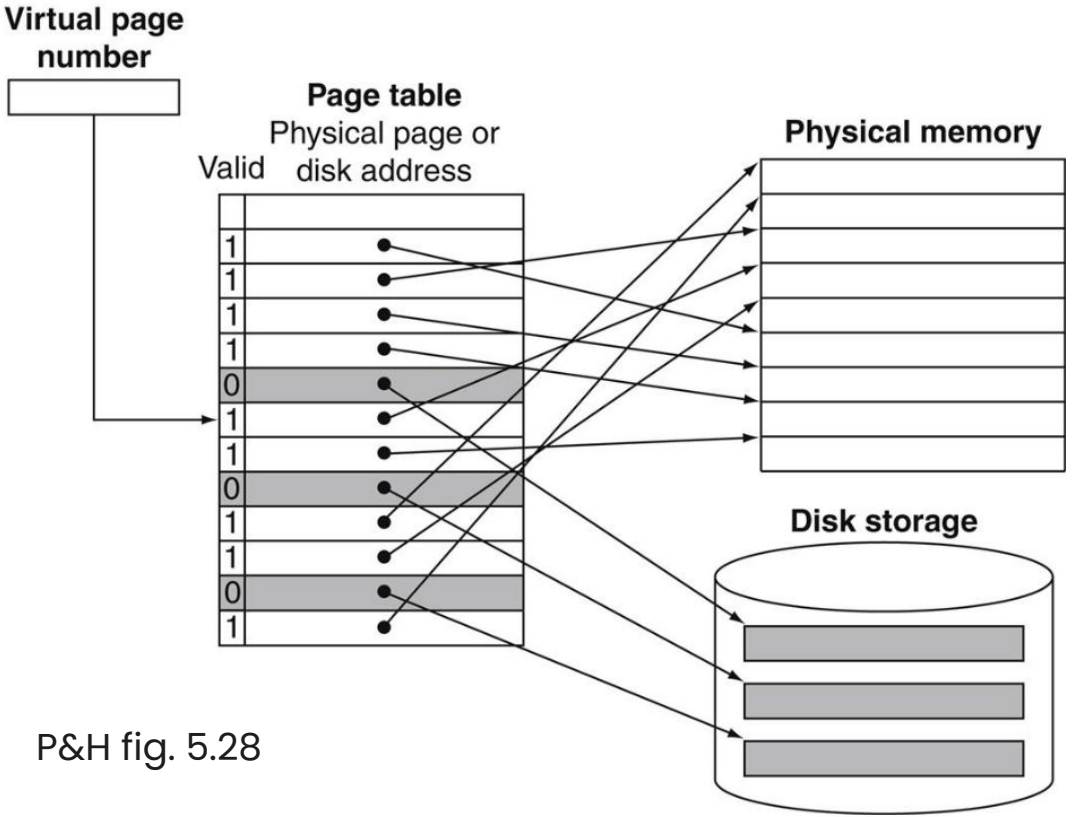
Page tables

Live in main memory (separate from pages themselves)

One for each process

Map virtual addresses to physical addresses (**not** a cache – **why?**)

Page faults managed by OS **(why?)**



P&H fig. 5.28

For our running example, how much space does a single page table take up?

? ? ?

What should happen if our virtual address space is so big that the page table can’t efficiently fit in main memory?

our page table might be sparsely filled

- multiple levels of page tables
- start with a smaller page table than we need and grow it
- hash into a smaller page table

page your page tables

Address translation in RISC-V

RISC-V spec vol. 2

4.1.11 Supervisor Address Translation and Protection (satp) Register

The satp register is an SXLEN-bit read/write register, formatted as shown in Figure 4.14 for SXLEN=32 and Figure 4.15 for SXLEN=64, which controls supervisor-mode address translation and protection. This register holds the physical page number (PPN) of the root page table, i.e., its supervisor physical address divided by 4 KiB; an address space identifier (ASID), which facilitates address-translation fences on a per-address-space basis; and the MODE field, which selects the current address-translation scheme. Further details on the access to this register are described in Section 3.1.6.5.

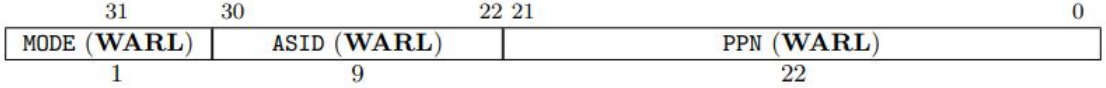


Figure 4.14: Supervisor address translation and protection register satp when SXLEN=32.

SXLEN=32		
Value	Name	Description
0	Bare	No translation or protection.
1	Sv32	Page-based 32-bit virtual addressing (see Section 4.3).

Virtual memory in RISC-V 32bit

32 bit virtual address space (4kb pages) → 20 bit virtual page numbers (VPNs)

22-bit physical page number (PPN)

Page tables are the size of a page

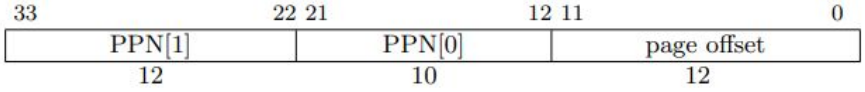


Figure 4.17: Sv32 physical address.

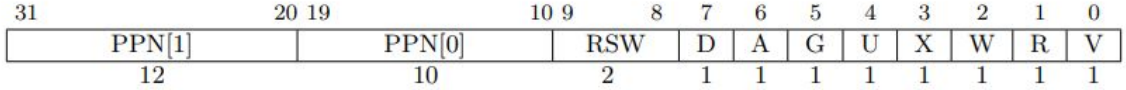
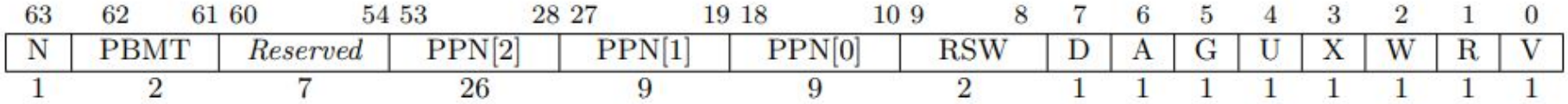


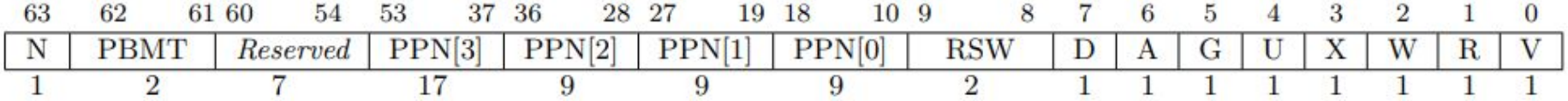
Figure 4.18: Sv32 page table entry.

Virtual memory in RISC-V 64bit



63	62 61	60 54	53 28	27 19	18 10	9 8	7	6	5	4	3	2	1	0
N	PBMT	*Reserved*	PPN[2]	PPN[1]	PPN[0]	RSW	D	A	G	U	X	W	R	V
1	2	7	26	9	9	2	1	1	1	1	1	1	1	1

Figure 4.21: Sv39 page table entry.



63	62 61	60 54	53 37	36 28	27 19	18 10	9 8	7	6	5	4	3	2	1	0
N	PBMT	*Reserved*	PPN[3]	PPN[2]	PPN[1]	PPN[0]	RSW	D	A	G	U	X	W	R	V
1	2	7	17	9	9	9	2	1	1	1	1	1	1	1	1

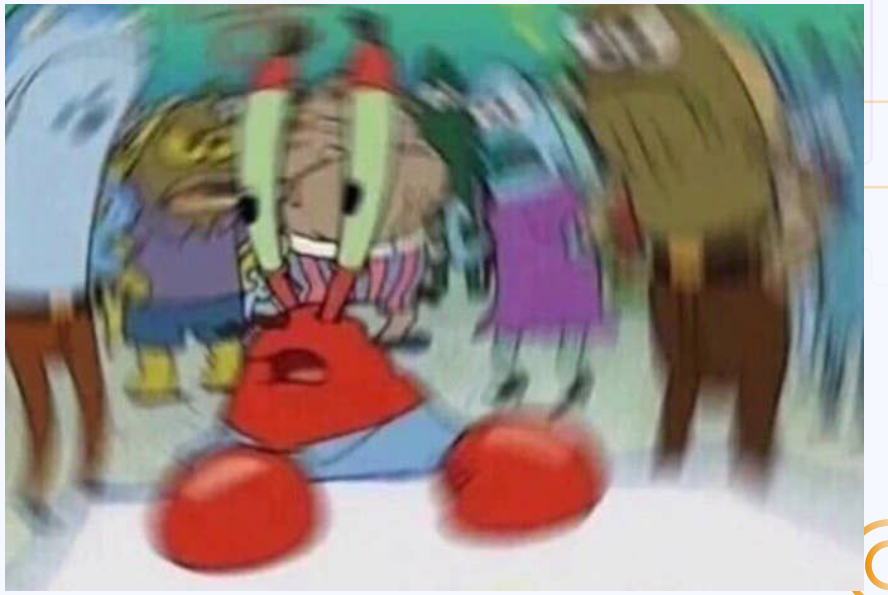
Figure 4.24: Sv48 page table entry.

In earlier lectures, we said memory is really slow, which means that virtual memory makes memory accesses *really really* slow (looking up translation in page table + then doing access). What can be done?

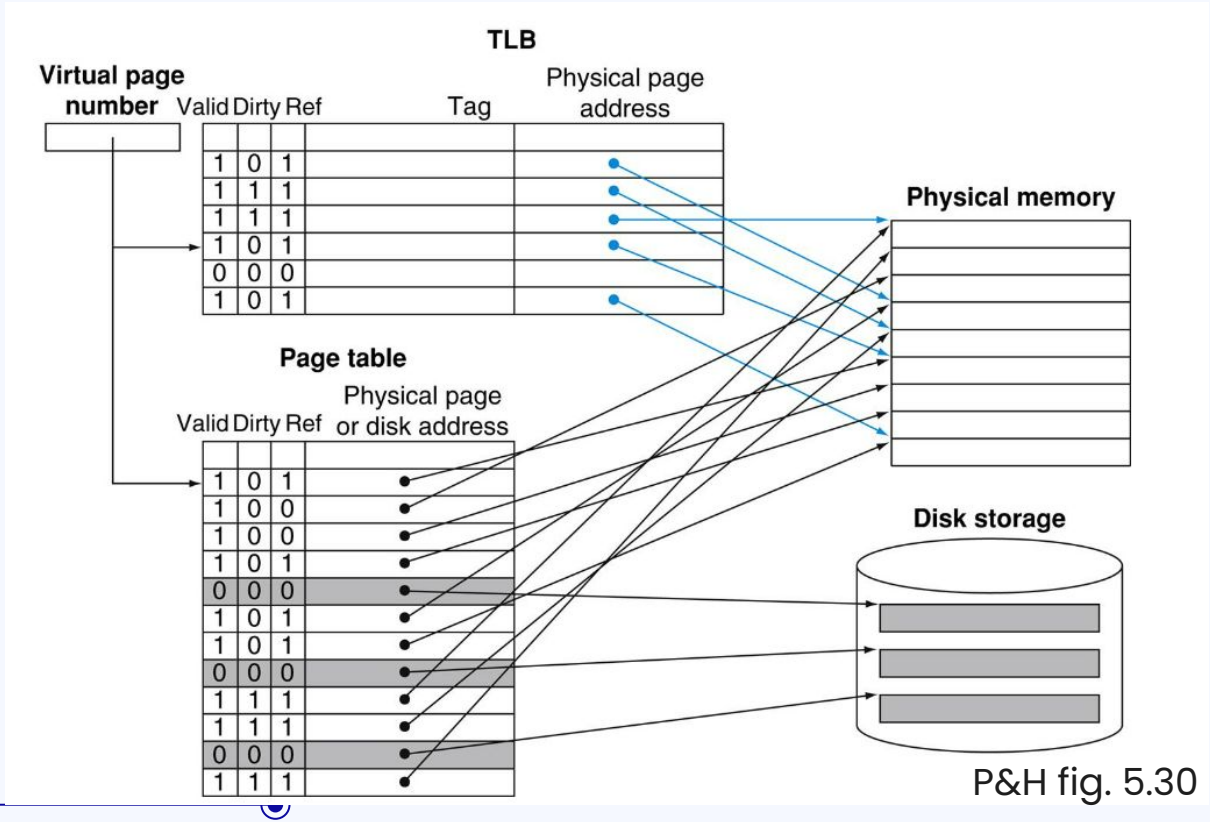
TLBs: a cache for the page table

For those counting: we have

- L1 I-cache
- L1 D-cache
- L2 cache
- L3 cache
- Main memory acting as a cache for disk
- TLBs acting as a cache for page tables (translation of virtual to physical addresses)
- ??? probably other caches in the future



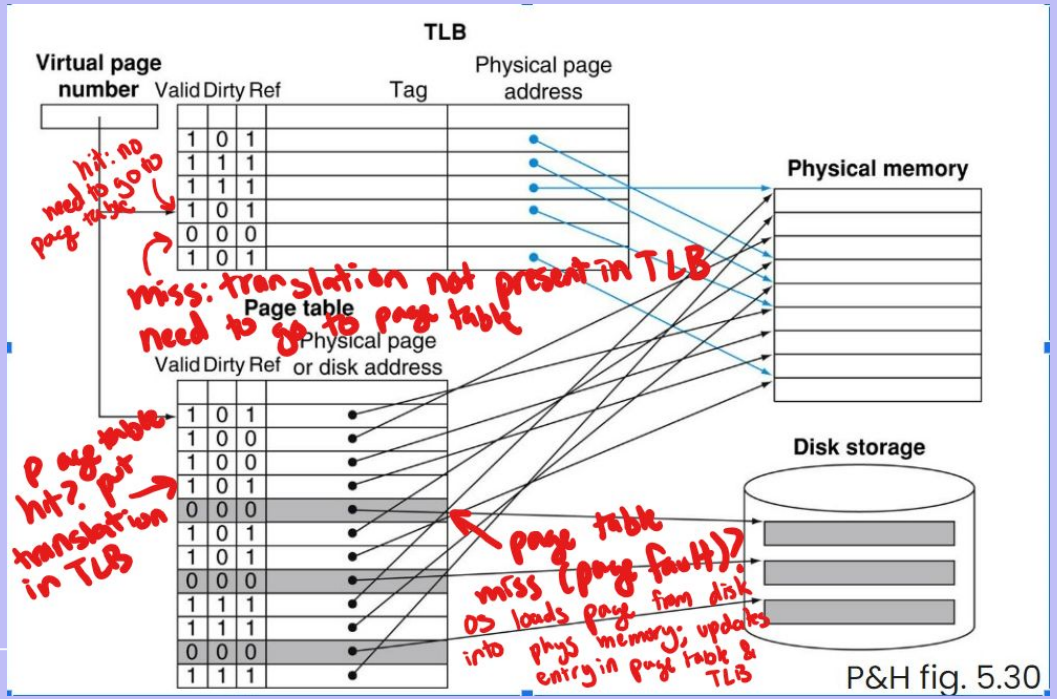
TLBs: does this clear it up?



P&H fig. 5.30

? ? ?

What is a TLB miss? What happens on a TLB miss?



For an instruction like `lw 10 0(sp)`, which do we do first?

- Check L1 cache
- Check main memory
- Check TLB
- Check page table

Interaction of TLB and cache (PIPT)

PIPT (explained on next slide) is physical cache (need to know physical address before indexing into cache)

P&H fig. 5.33

TLB	Page table	Cache	Possible? If so, under what circumstance?
Hit	Hit	Miss	Possible, although the page table is never really checked if TLB hits.
Miss	Hit	Hit	TLB misses, but entry found in page table; after retry, data is found in cache.
Miss	Hit	Miss	TLB misses, but entry found in page table; after retry, data misses in cache.
Miss	Miss	Miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
Hit	Miss	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Miss	Hit	Impossible: cannot have a translation in TLB if page is not present in memory.
Miss	Miss	Hit	Impossible: data cannot be allowed in cache if the page is not in memory.

VIPT caches

PIPT (physically indexed, physically tagged) caches come at page translation cost

VIVT (virtually indexed, virtually tagged) caches cause aliasing issues (two virtual addresses mapping to same physical address)

VIPT: perform cache lookup + TLB lookup in parallel

both virtual and physical address have the same page offset

cache size now limited to page size

VIPT

Sunday, February 25, 2024 4:41 PM

two ways to interpret an address: