Virtual Memory

Process Abstraction, Part 2: Private Address Space

Motivation: why not direct physical memory access?

Address translation with pages

Optimizing translation: translation lookaside buffer

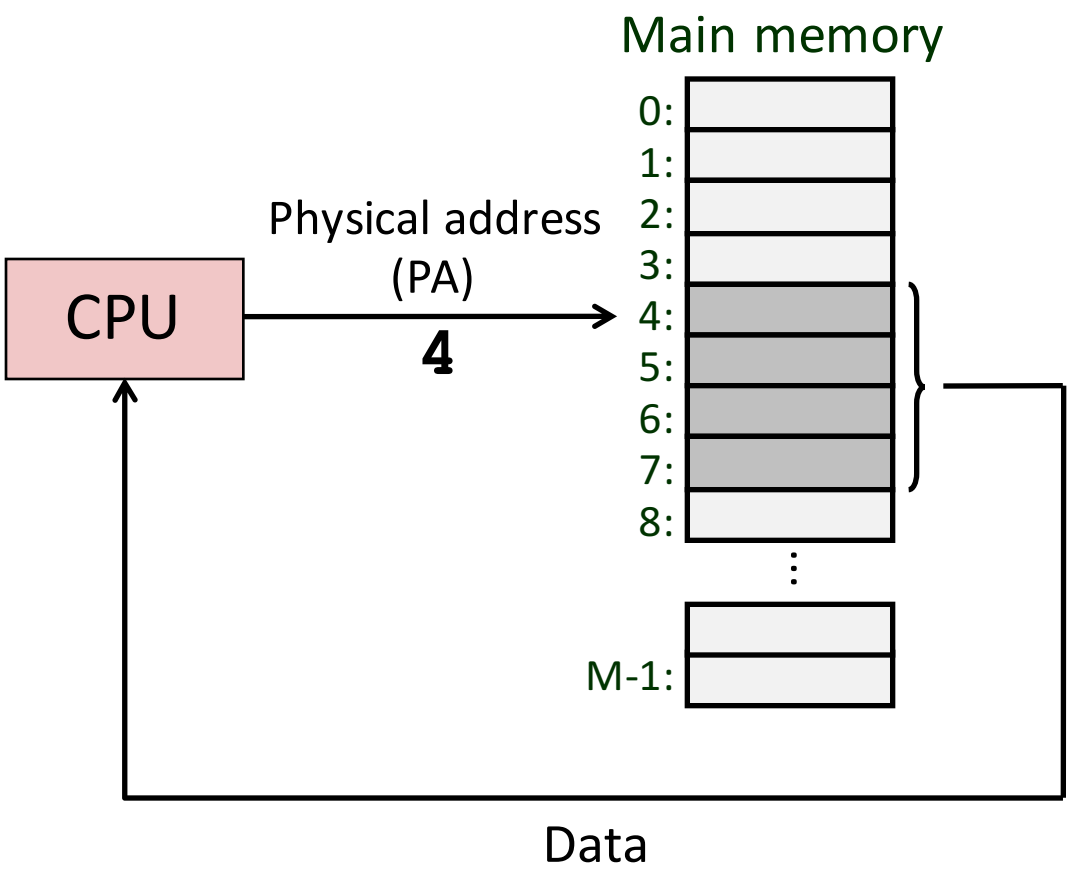
Extra benefits: sharing and protection

Memory as a contiguous array of bytes is a lie! Why?

Problems with Physical Addressing

Fine for small embedded systems without processes.

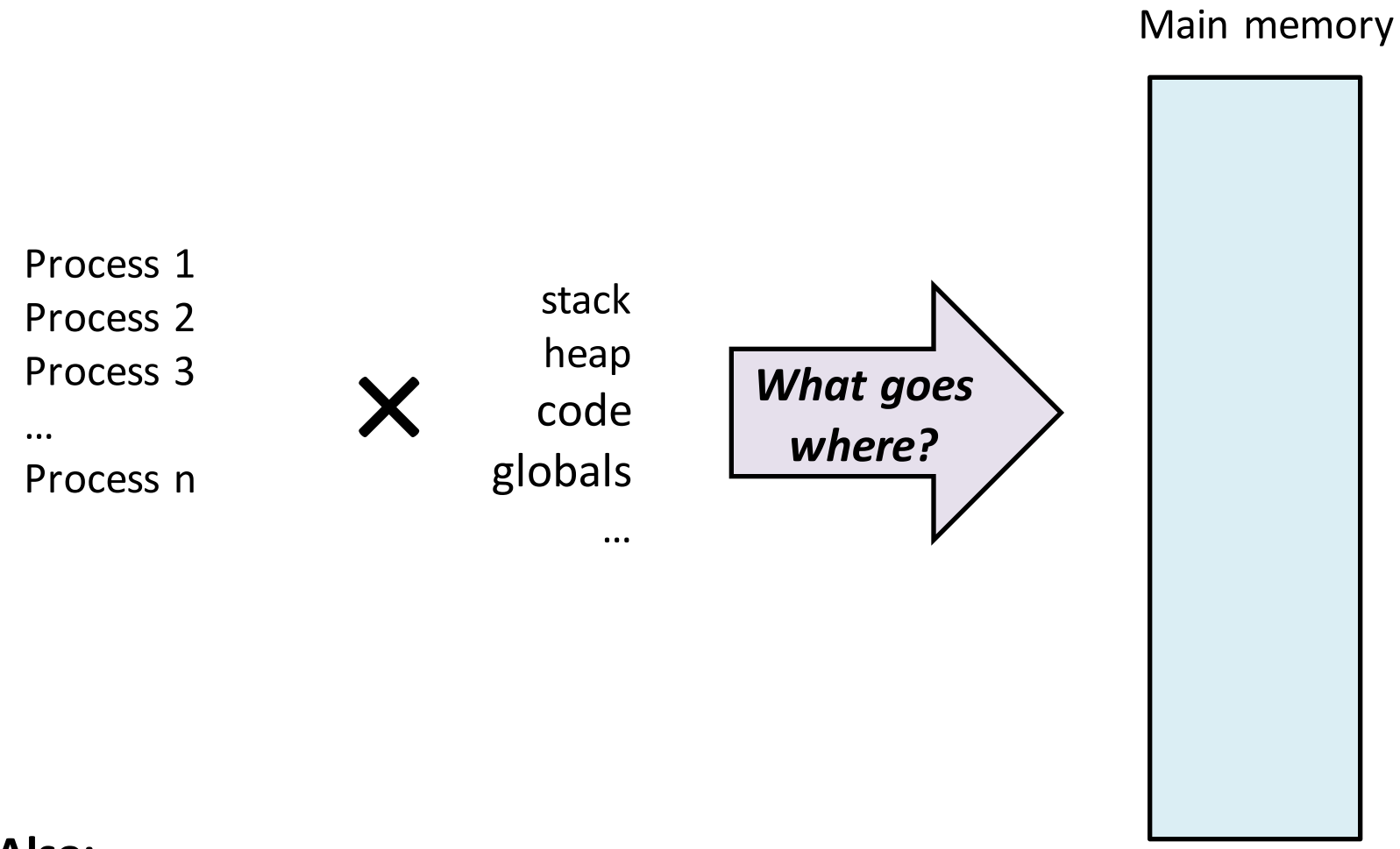
Elevators, microwaves, radio-powered devices, …



What about larger systems?

With many processes?

Problem 1: Memory Management



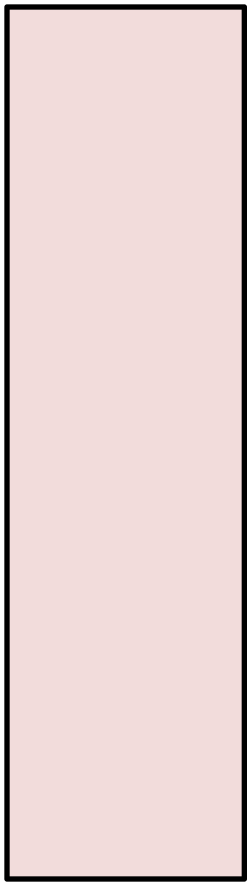
Also:

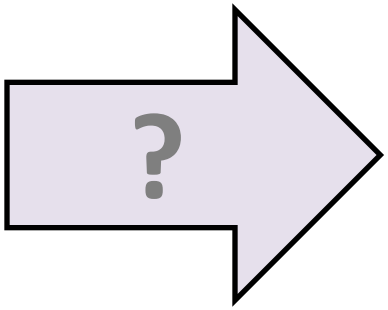
Context switches must swap out entire memory contents.

Isn't that **expensive**?

Problem 2: Capacity

64-bit addresses can address several exabytes (18,446,744,073,709,551,616 bytes)



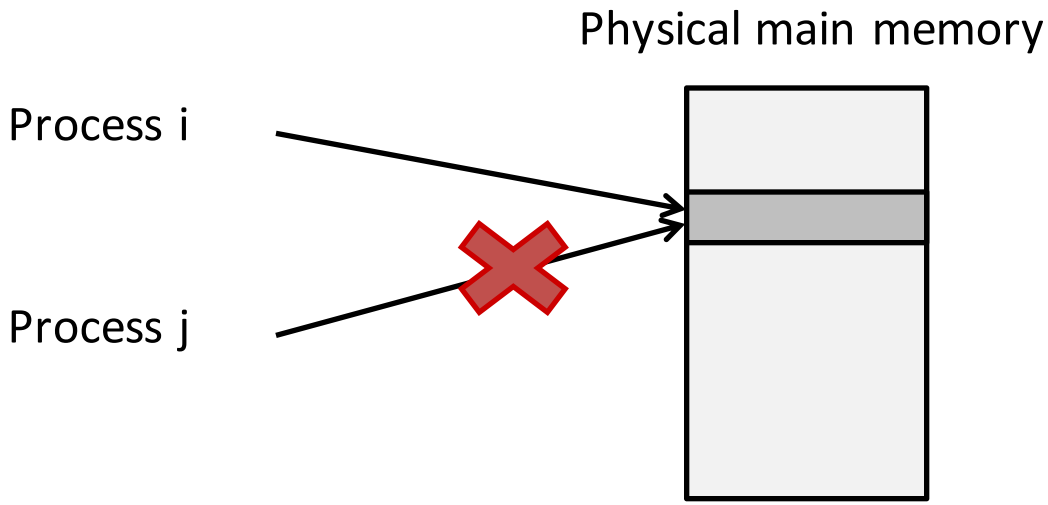


Physical main memory offers a few gigabytes (e.g. 8,589,934,592 bytes)

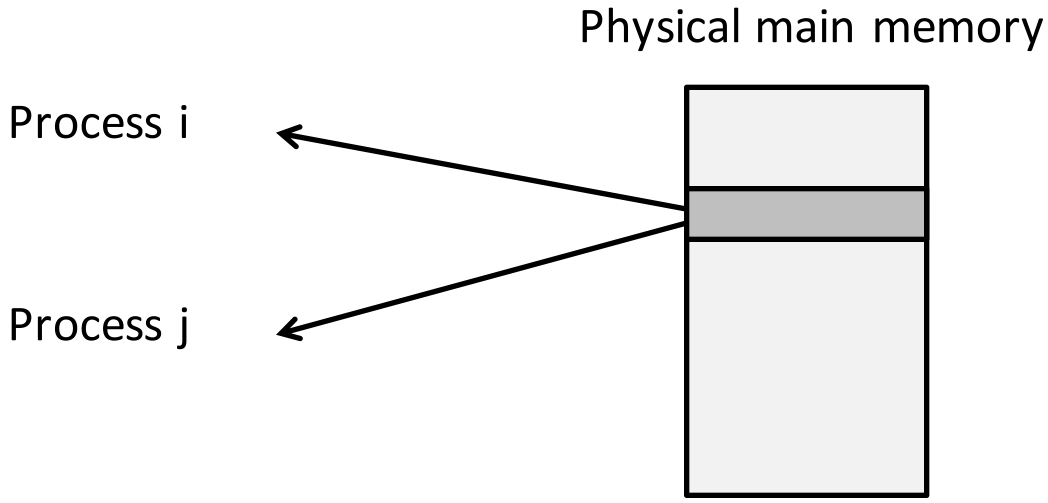
(Actually, it's smaller than that dot compared to virtual memory.)

1 virtual address space per process, with many processes…

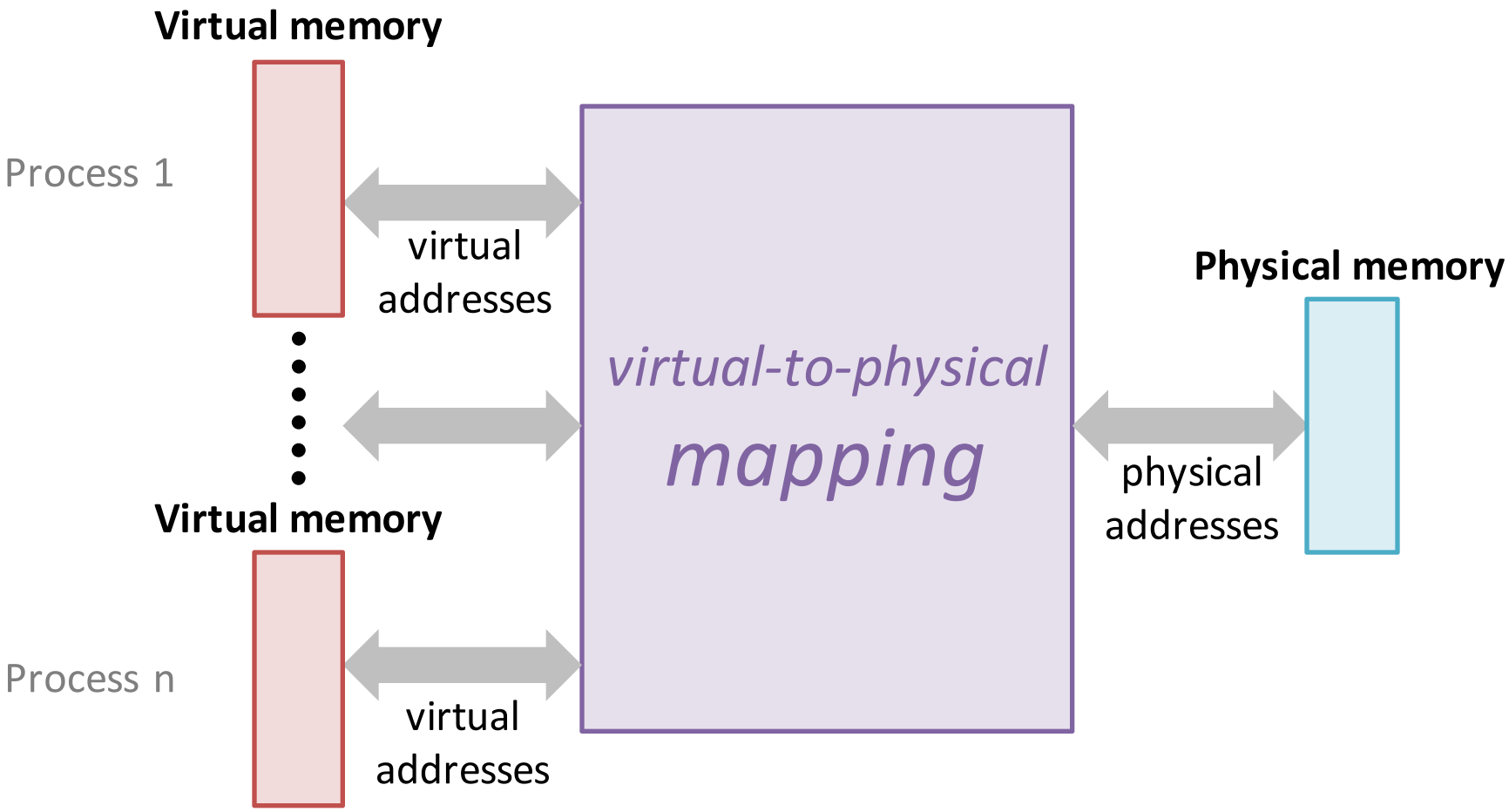
Problem 3: Protection



Problem 4: Sharing



Solution: Virtual Memory (address *indirection*)



Private virtual address space per process.

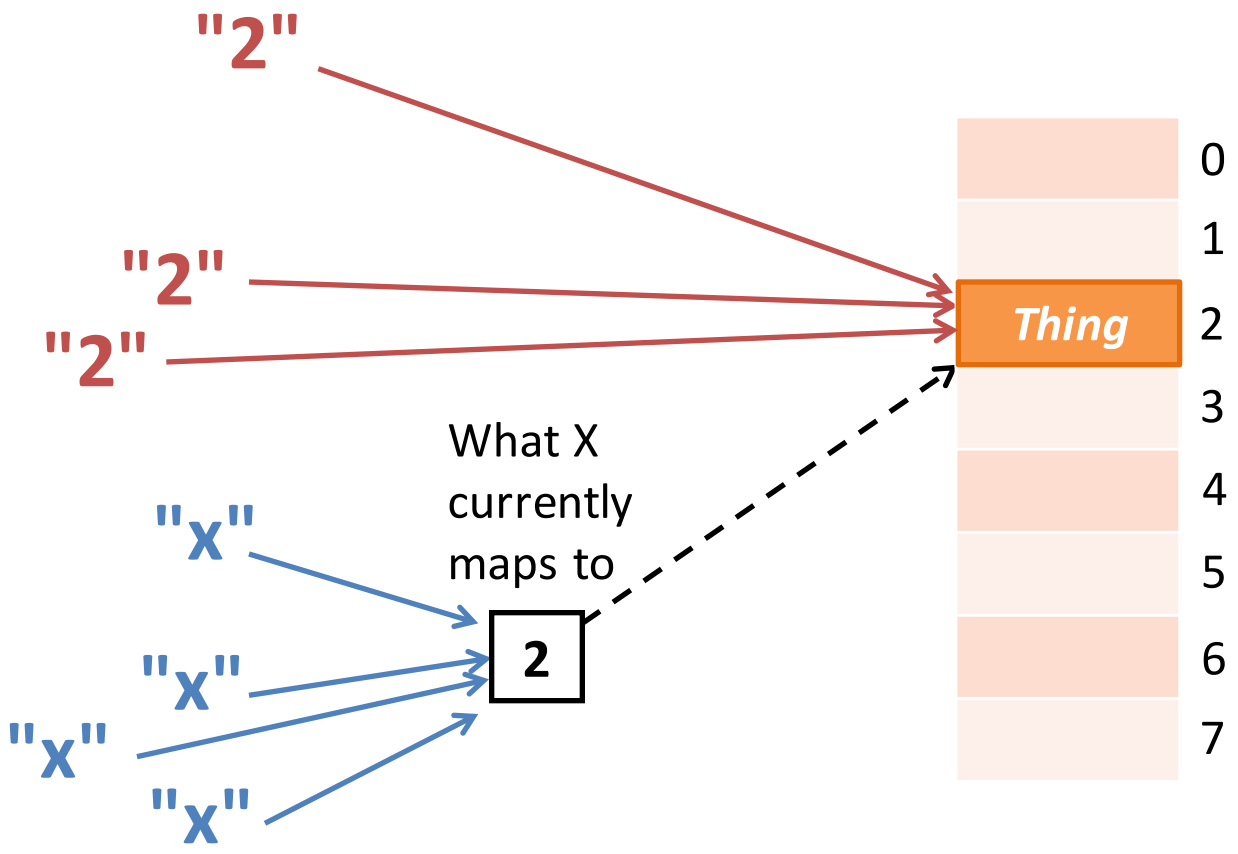
Single physical address space managed by OS/hardware.

Indirection

(it's everywhere!)

Direct naming

Indirect naming



What if we move ***Thing***?

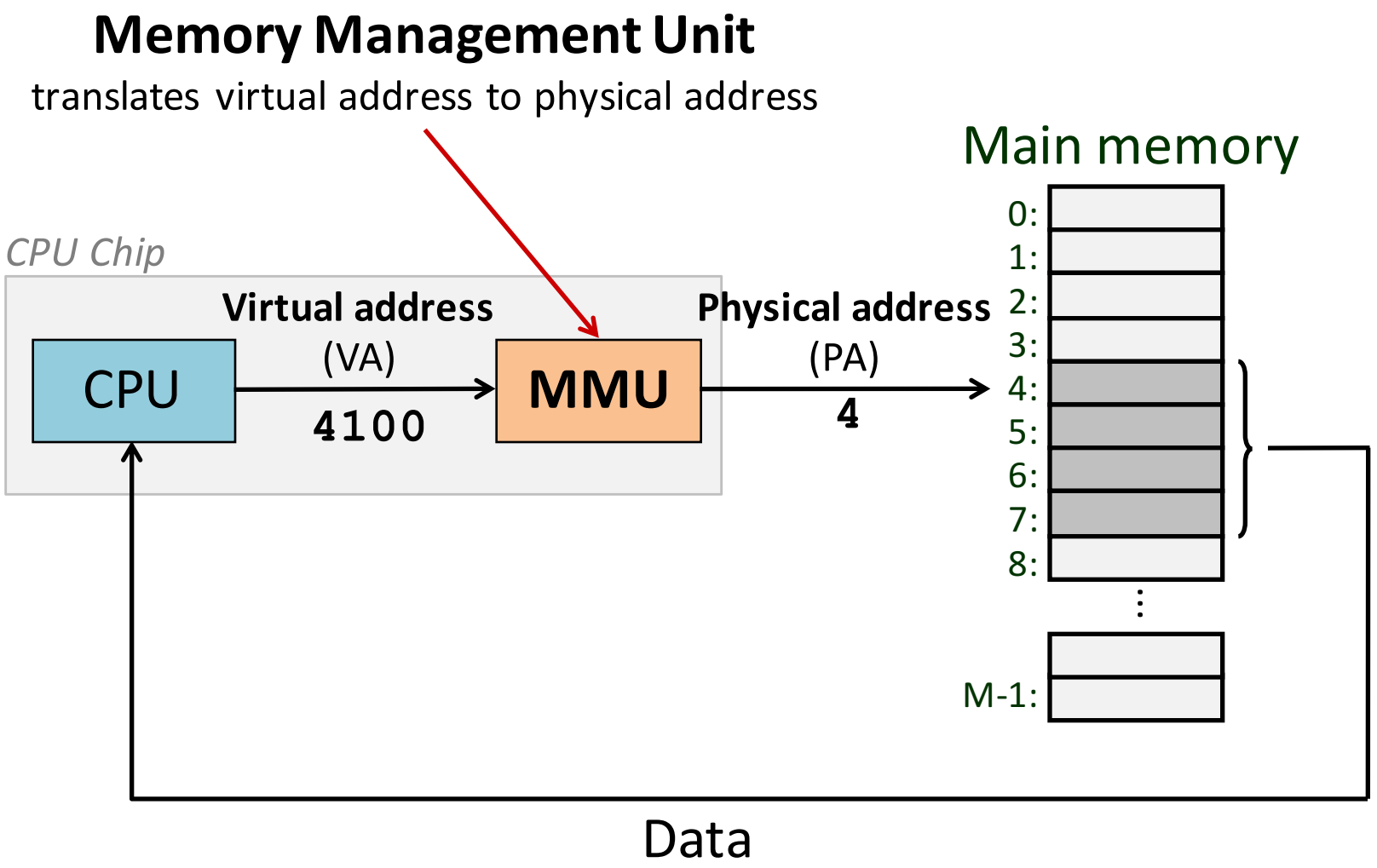
Tangent: **Indirection everywhere**

- Pointers
- Constants
- Procedural abstraction
- Domain Name Service (DNS)
- Dynamic Host Configuration Protocol (DHCP)
- Phone numbers
- 911
- Call centers
- Snail mail forwarding
- …

"Any problem in computer science can be solved by adding another level of indirection."
–David Wheeler, inventor of the subroutine, or Butler Lampson

Another Wheeler quote? "Compatibility means deliberately repeating other people's mistakes."

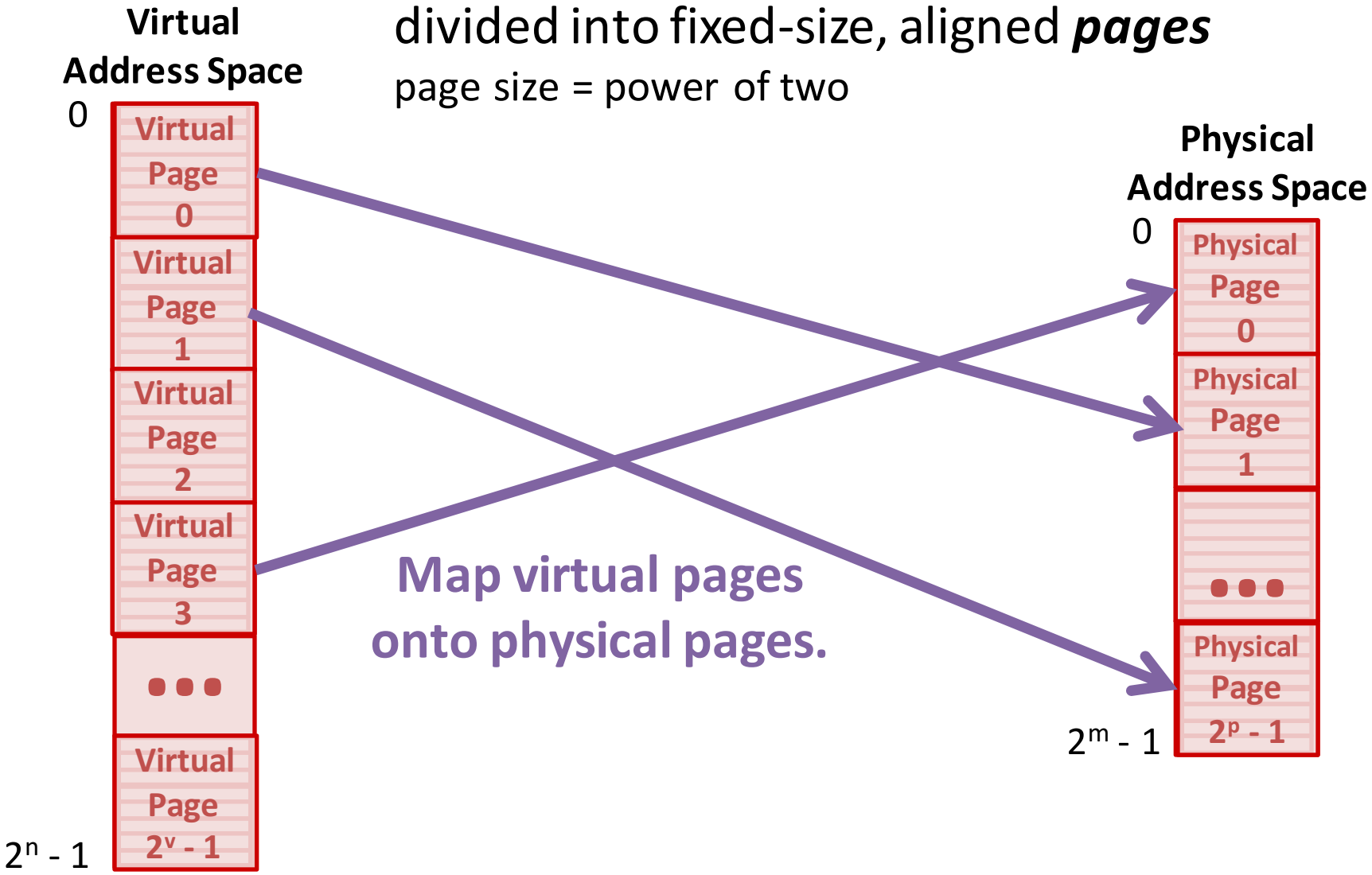
Virtual Addressing and Address Translation



*Physical addresses are **invisible** to programs.*

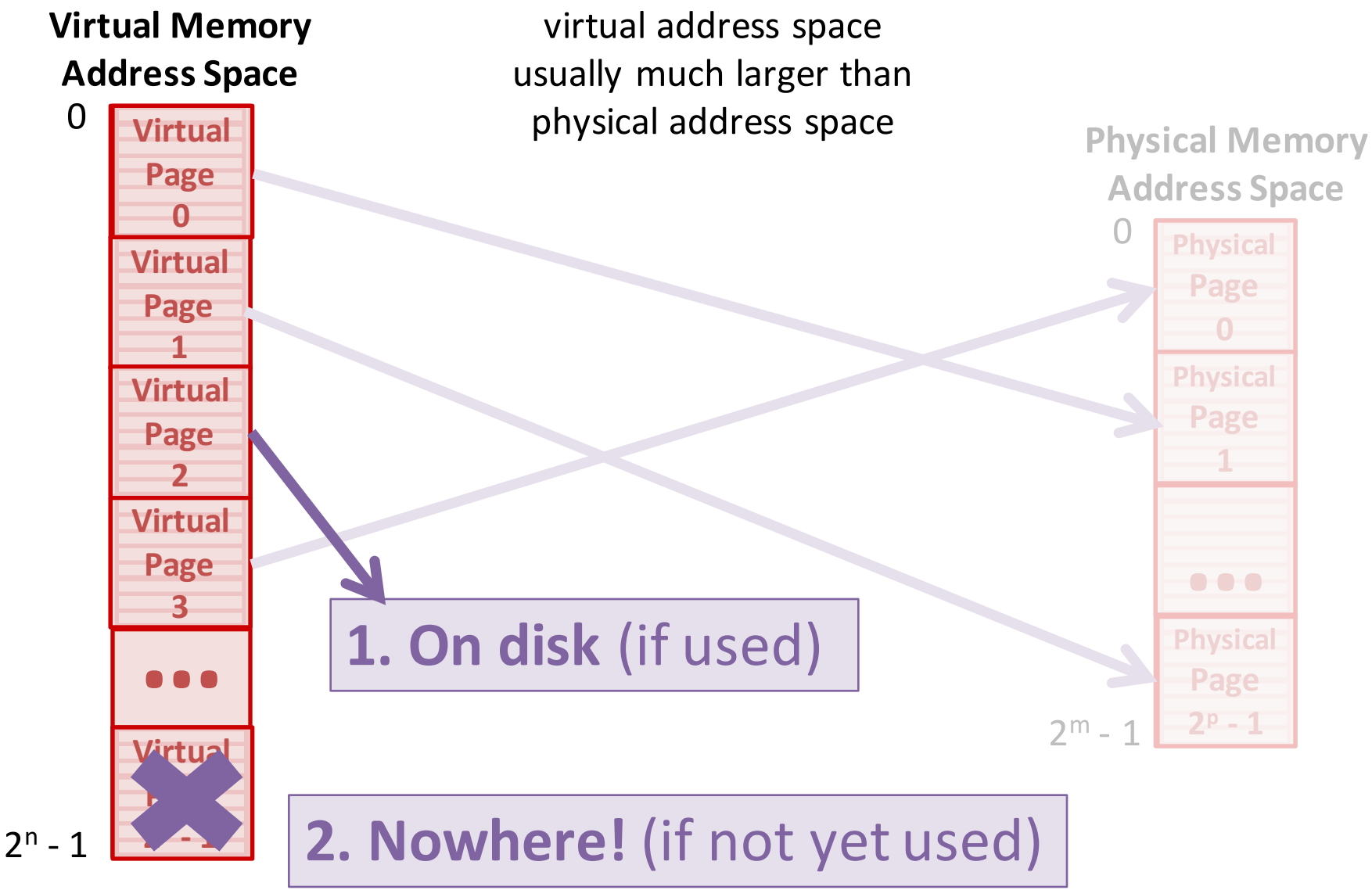
Page-based Mapping

both address spaces
divided into fixed-size, aligned ***pages***
page size = power of two



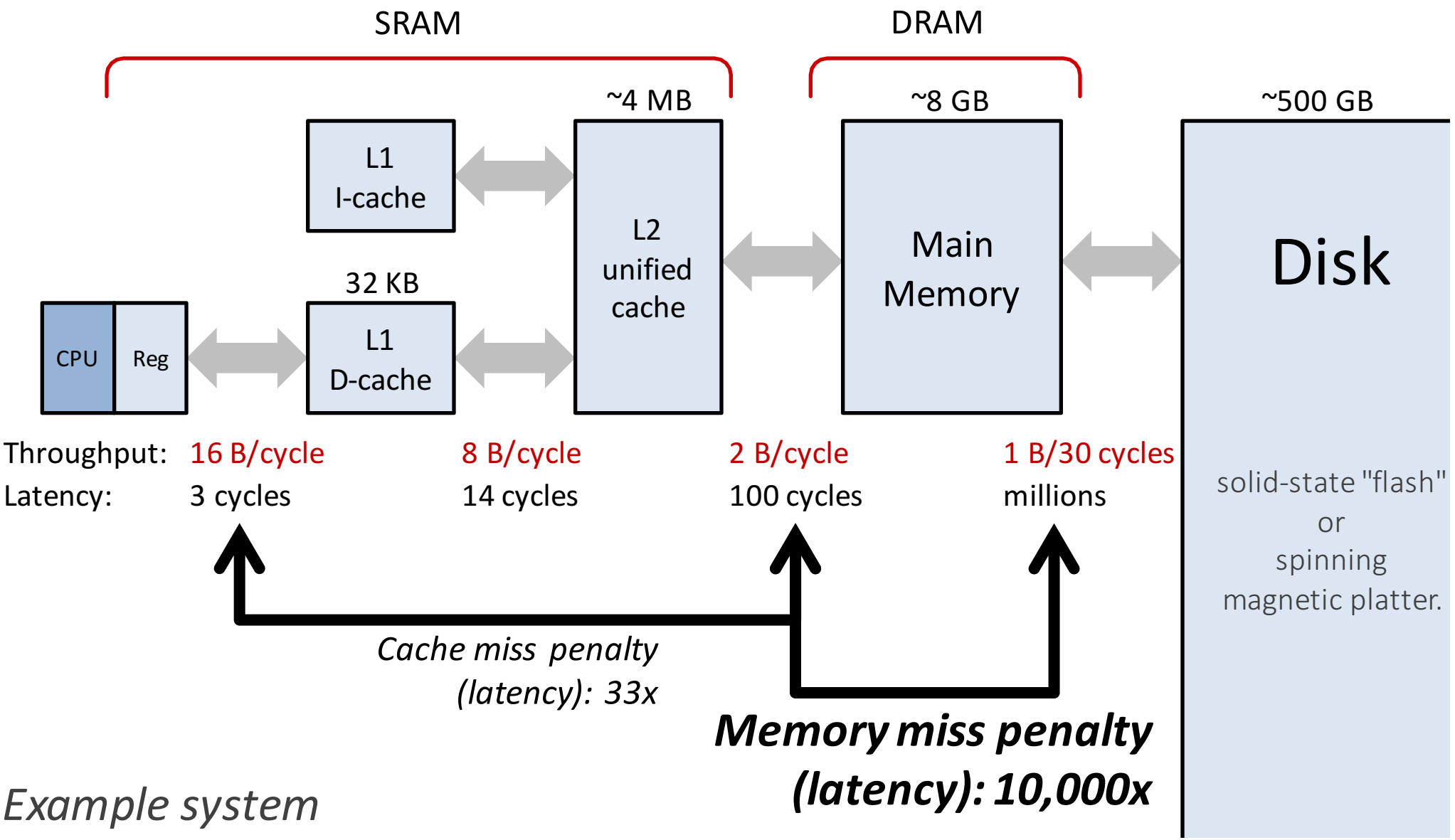
Some virtual pages do not fit! Where are they stored?

Some virtual pages do not fit! Where are they stored?



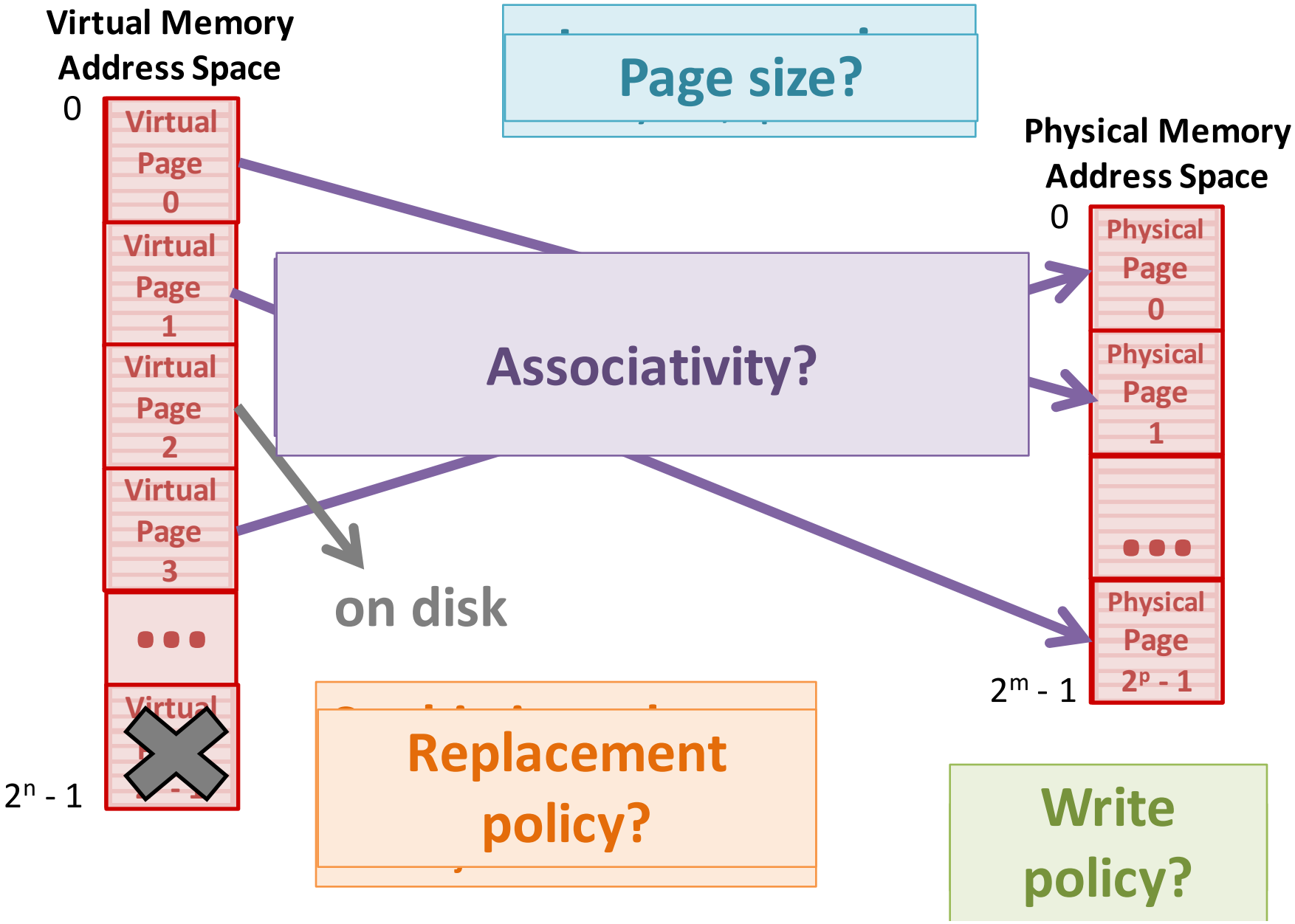
Not drawn to scale

Virtual Memory: cache for disk?

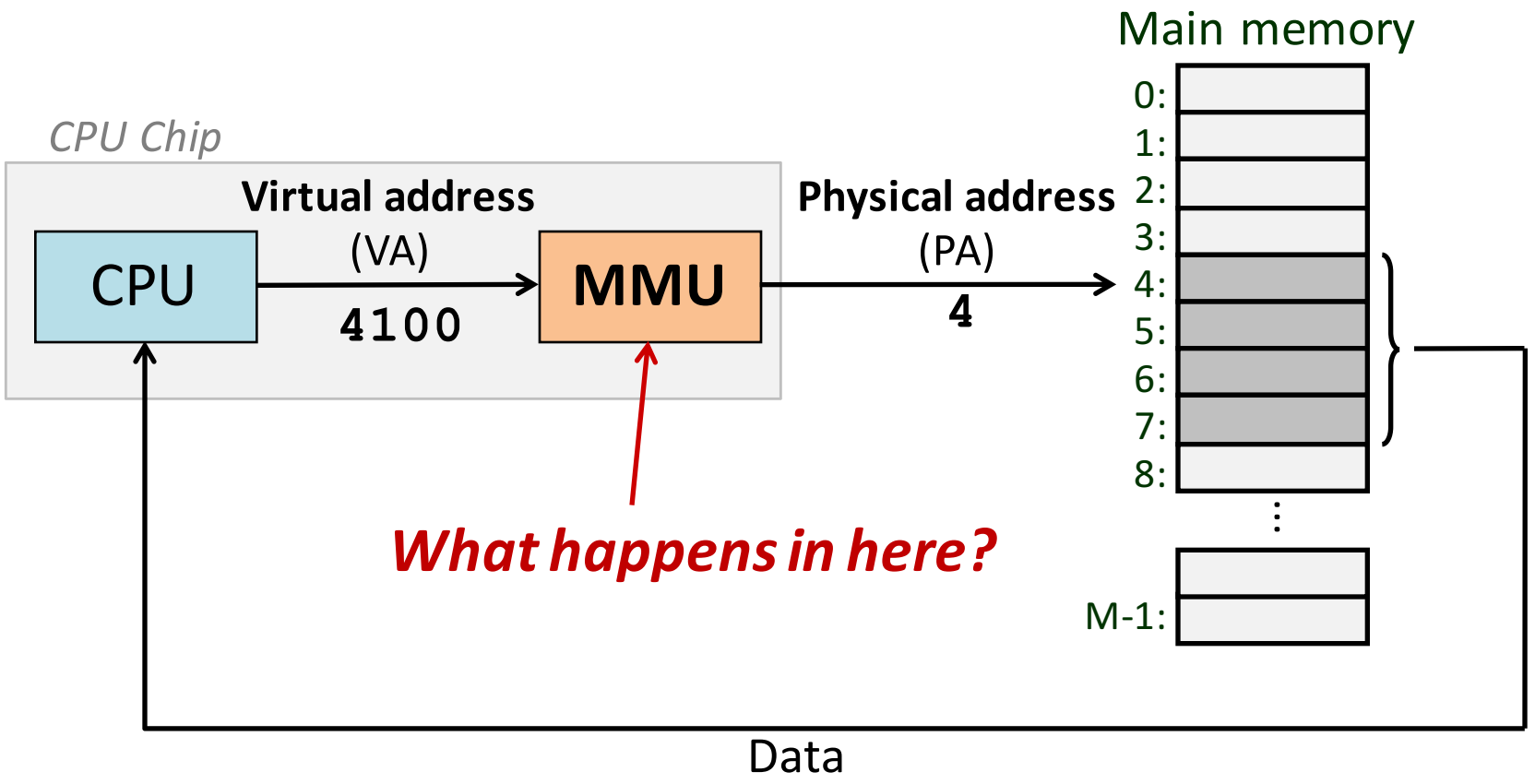


Example system

Design for a Slow Disk: Exploit Locality



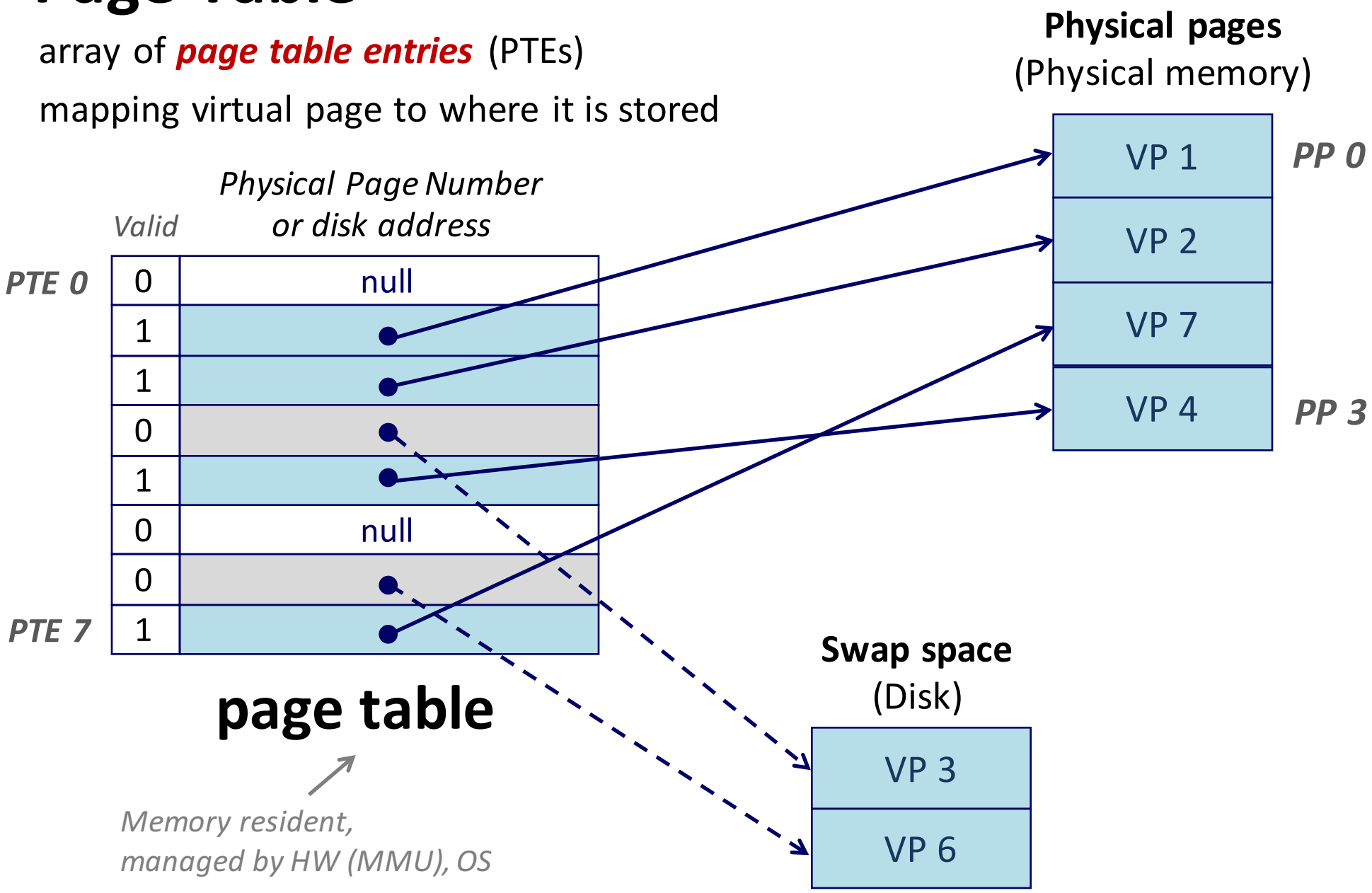
Address Translation



Page Table

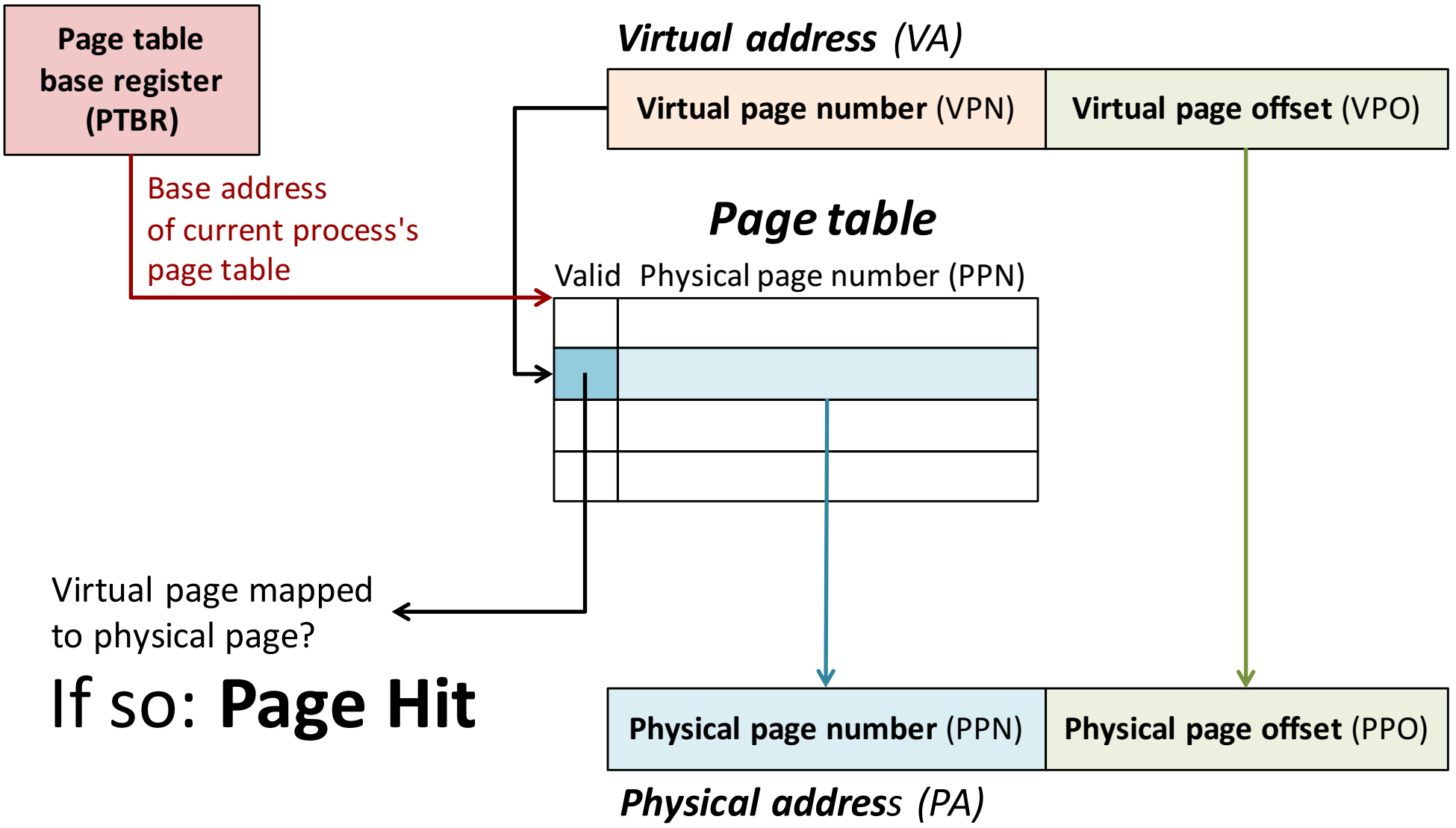
array of ***page table entries*** (PTEs)

mapping virtual page to where it is stored

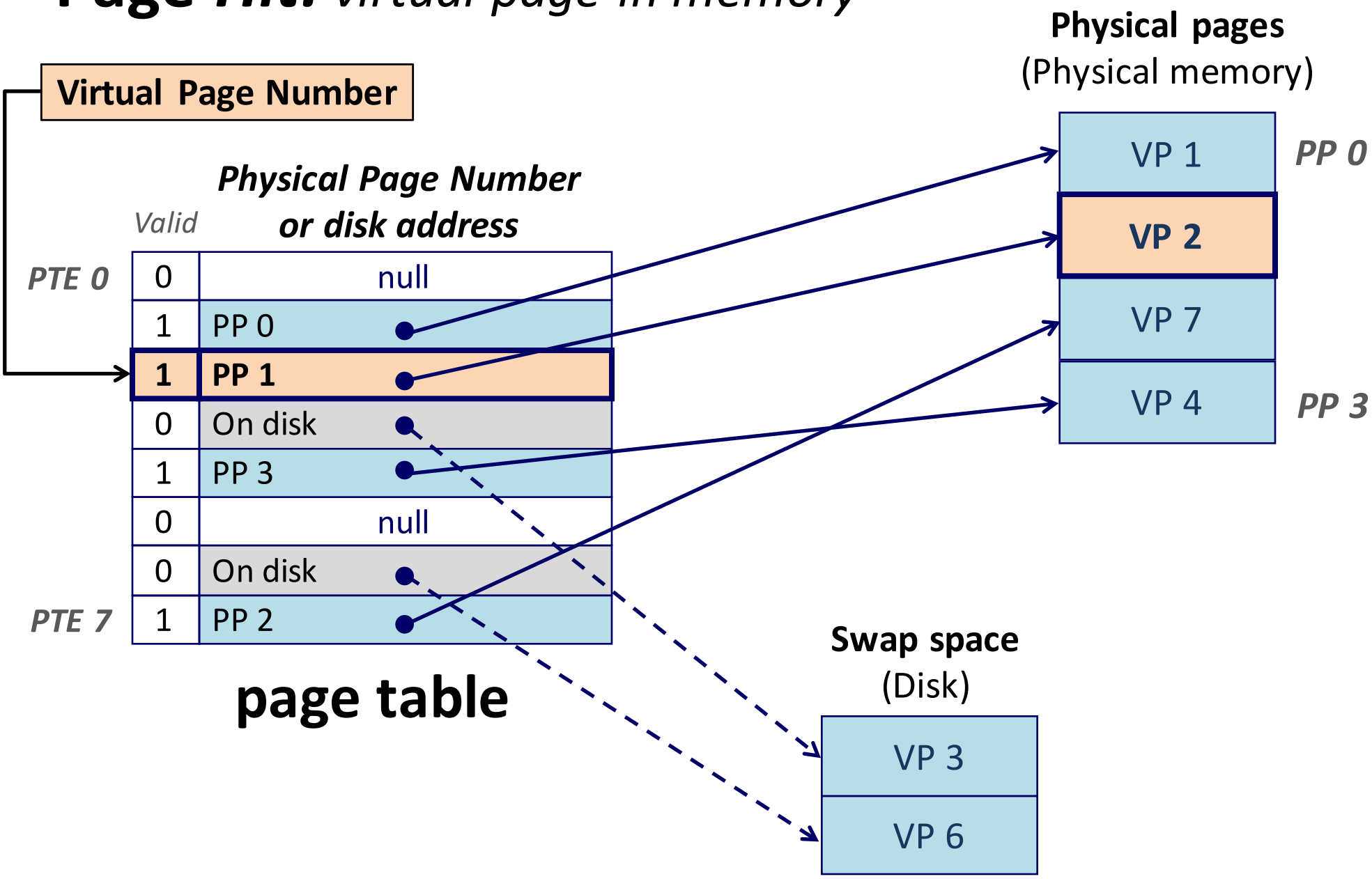


How many page tables are in the system?

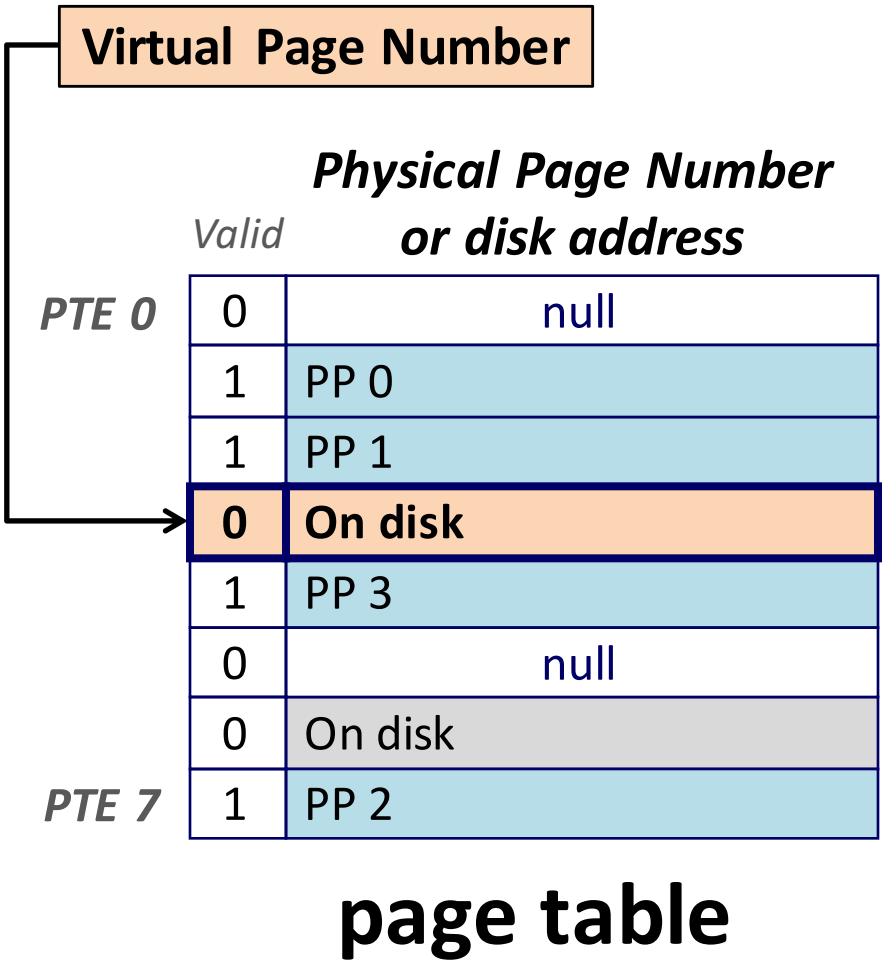
Address Translation with a **Page Table**



Page *Hit:* *virtual page in memory*

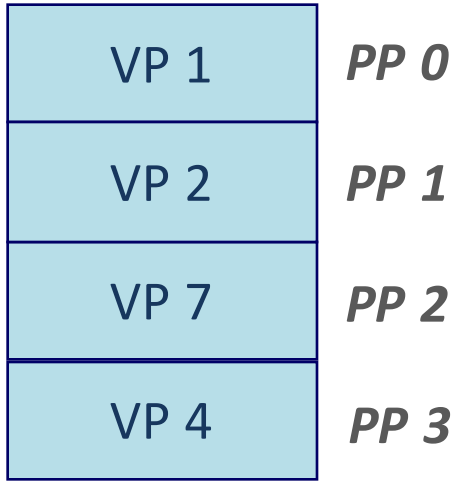


Page *Fault:*

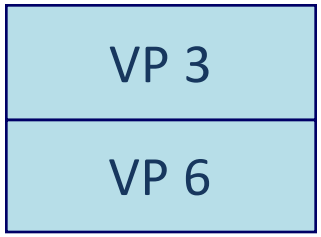


page table

Physical pages
(Physical memory)

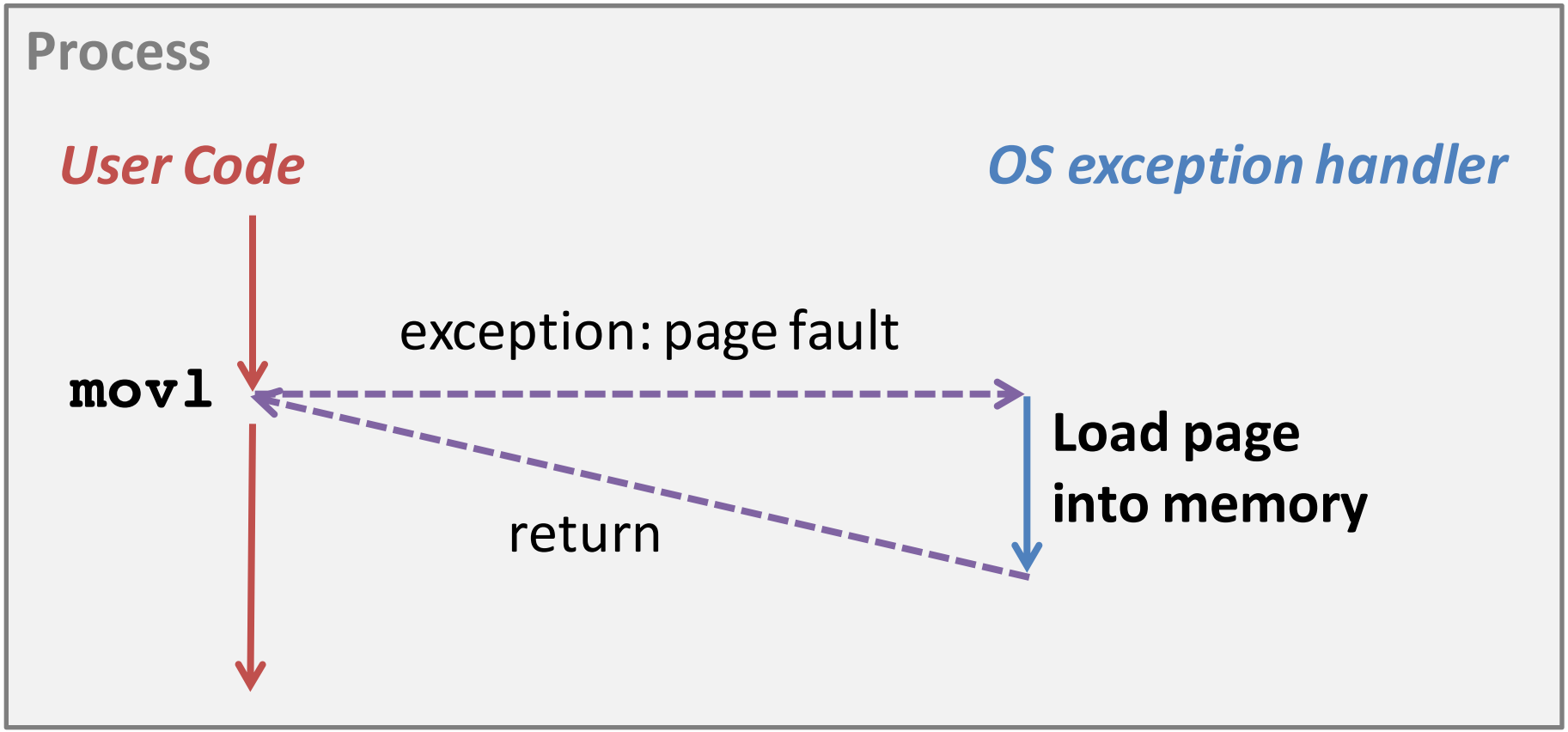


Swap space
(Disk)



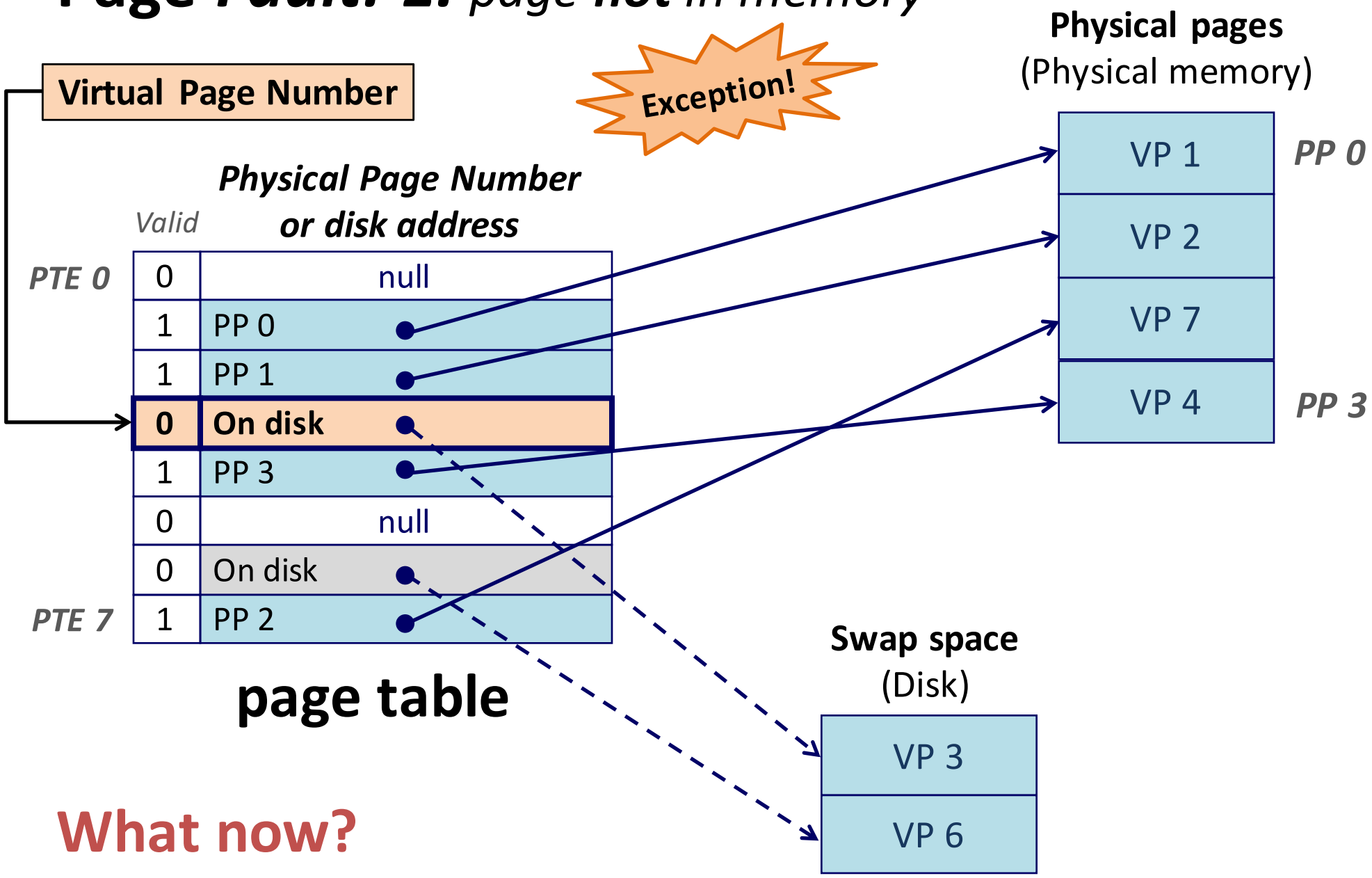
Page *Fault:* *exceptional control flow*

Process accessed virtual address in a page that is not in physical memory.



Returns to faulting instruction:
`movl` is executed *again*!

Page *Fault: 1.* *page* ***not*** *in memory*

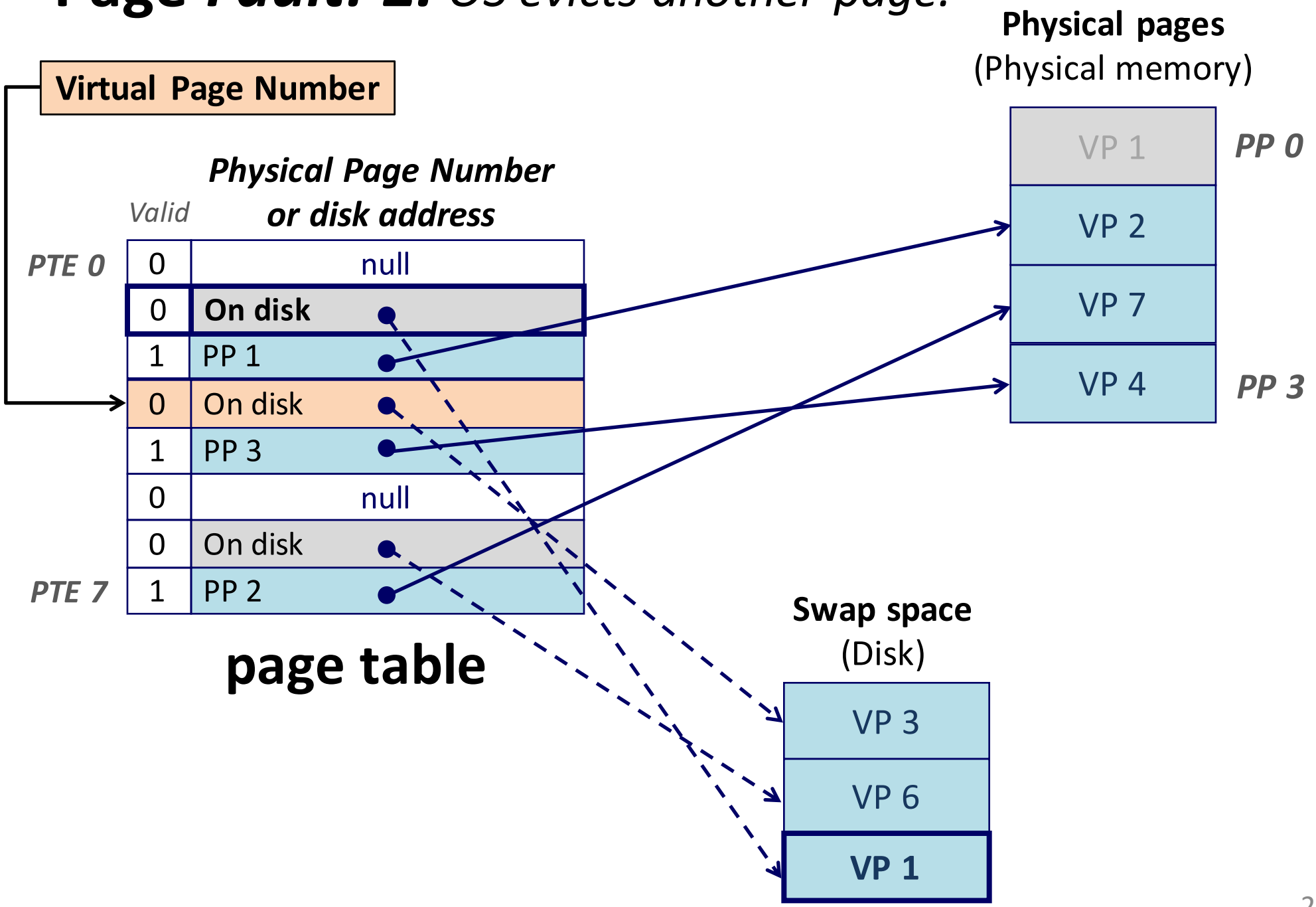


What now?

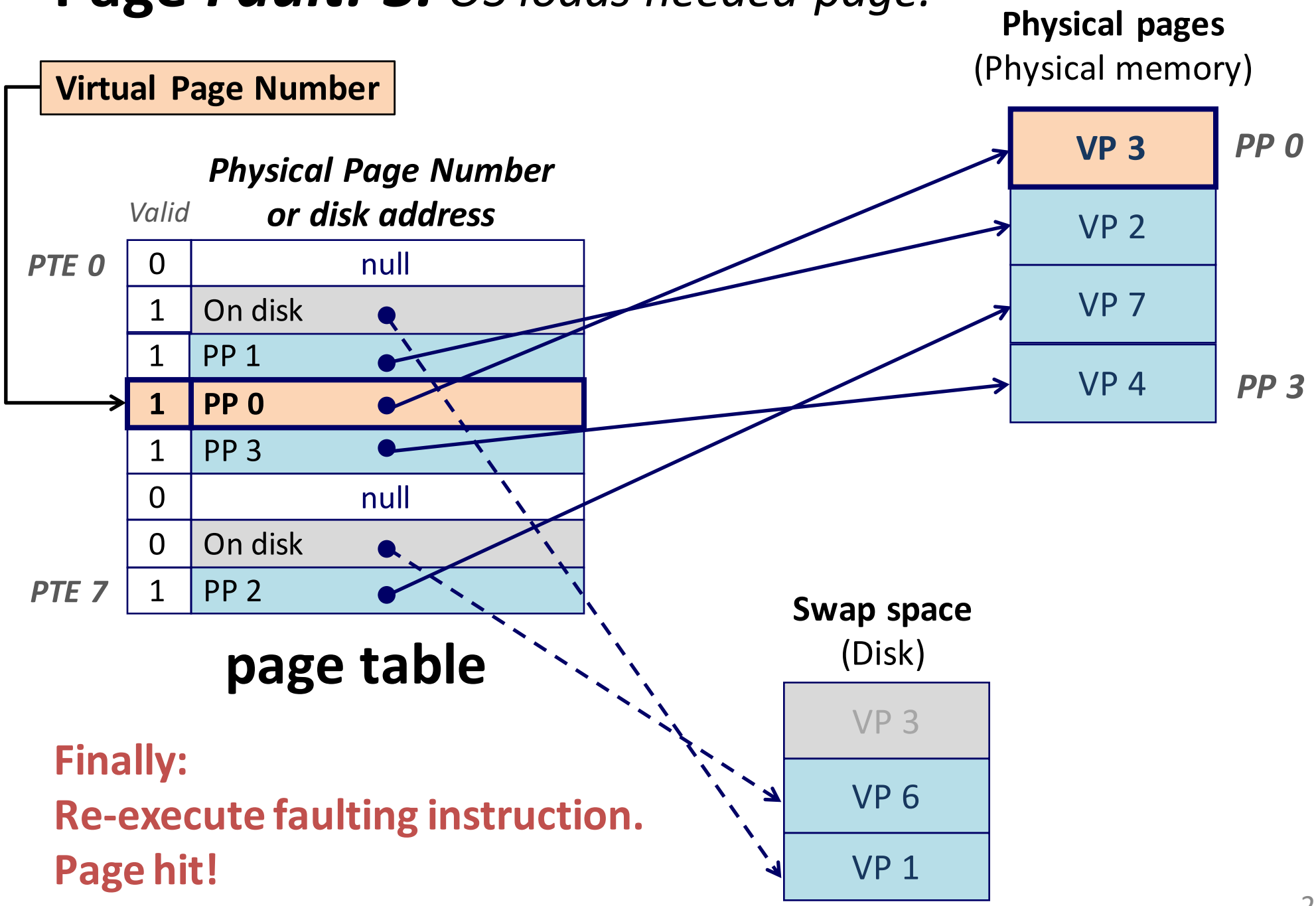
OS handles fault

"Page out"

Page *Fault: 2.* *OS evicts another page.*



Page *Fault: 3.* *OS loads needed page.*



Finally:
Re-execute faulting instruction.
Page hit!

Terminology

context switch

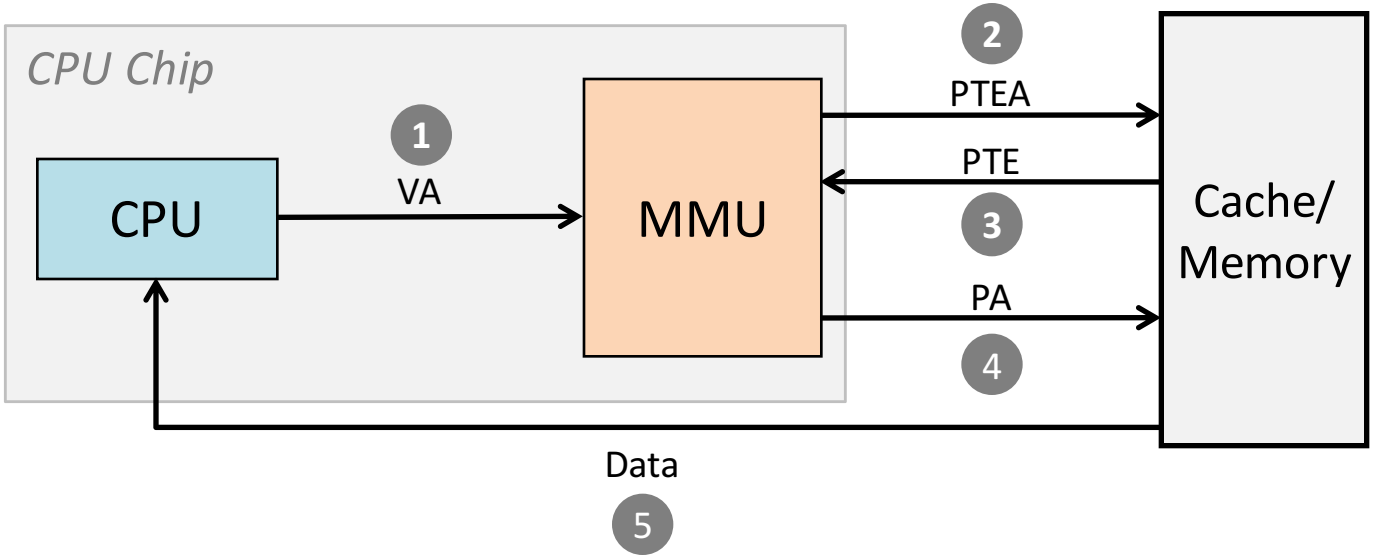
page in

page out

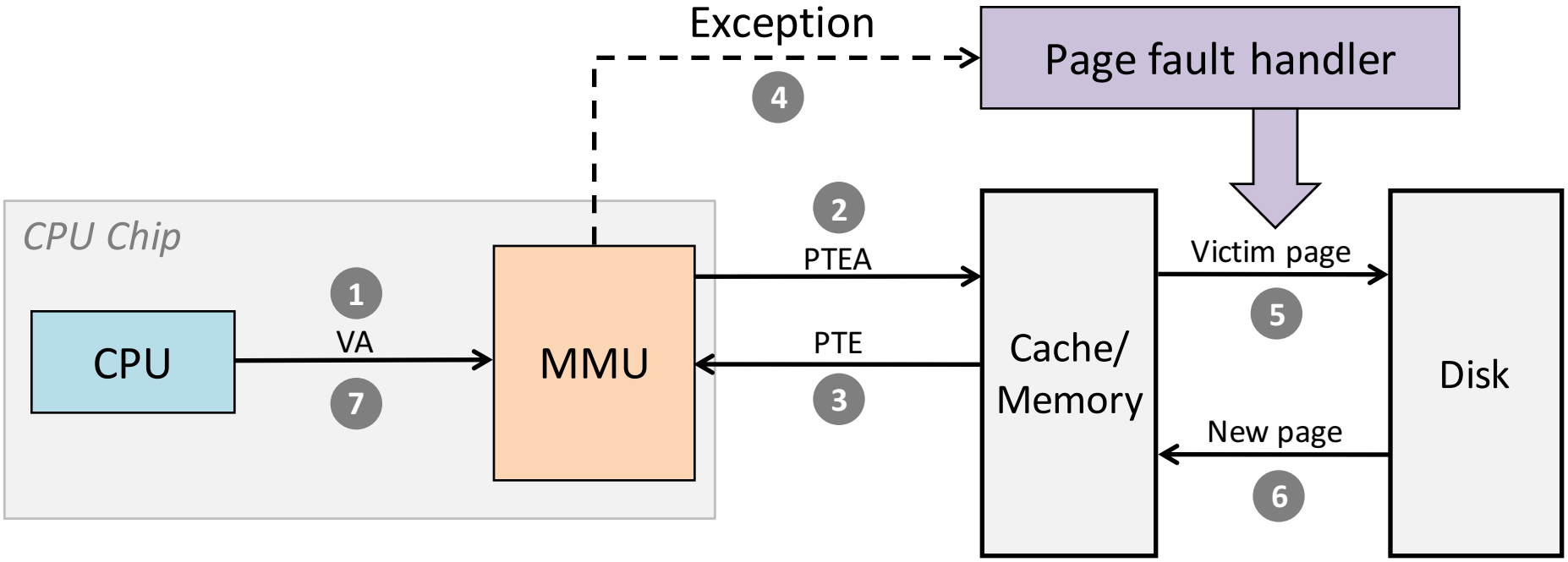
swap

thrash

Address Translation: Page *Hit*



Address Translation: Page *Fault*



How fast is translation?

How many physical memory accesses are required to complete one virtual memory access?

Translation Lookaside Buffer (TLB)

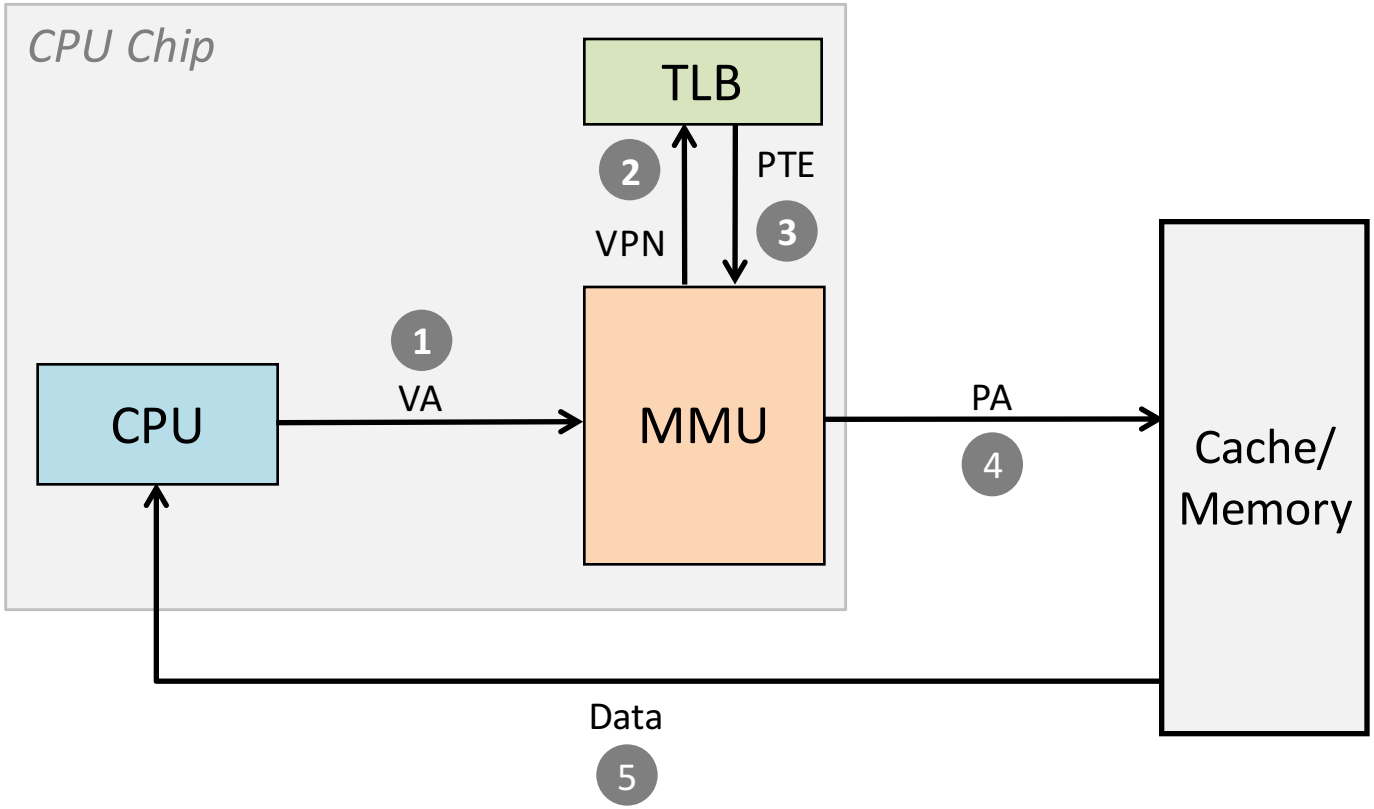
Small hardware cache in MMU just for page table entries

- e.g., 128 or 256 entries

Much faster than a page table lookup in memory.

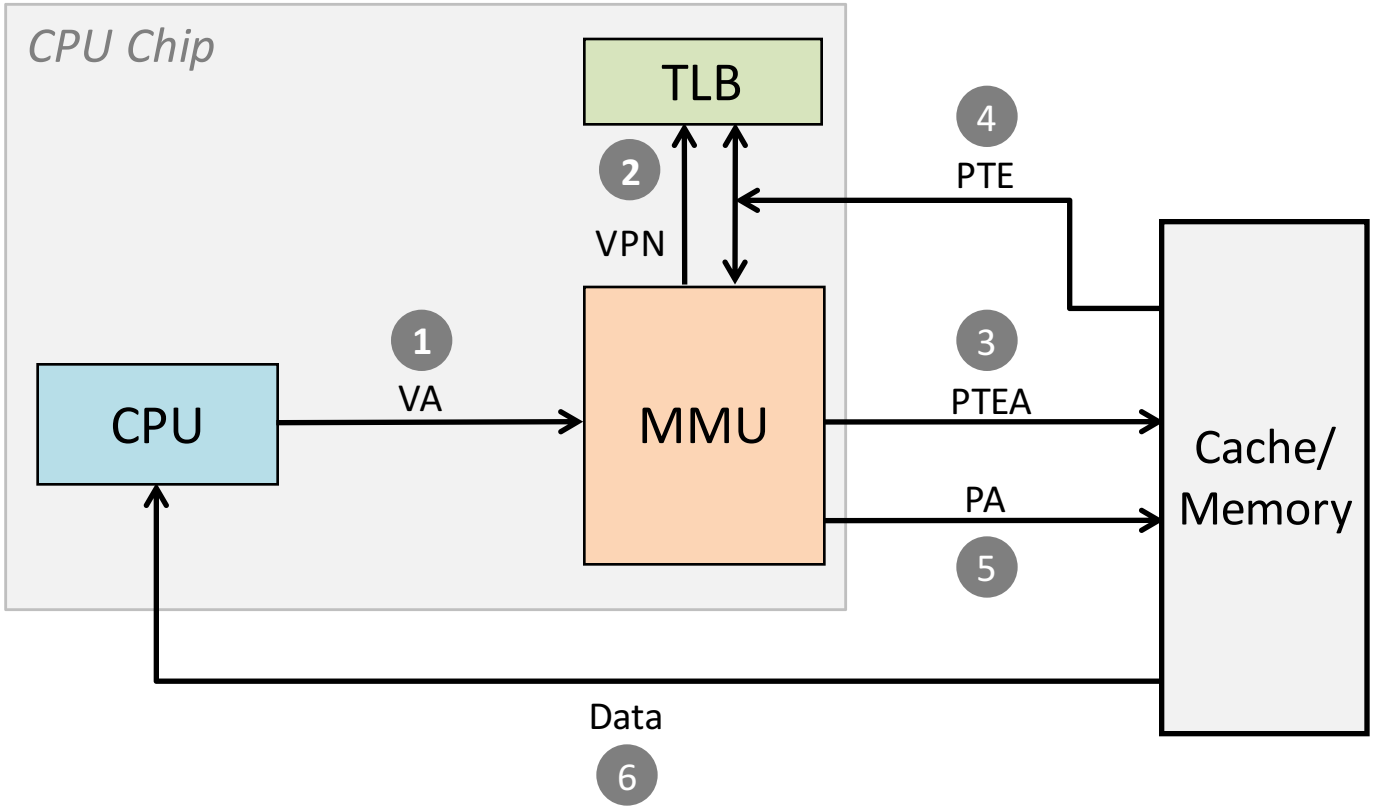
In the running for *"un/classiest name of a thing in CS"*

TLB Hit



A TLB hit eliminates a memory access

TLB Miss



A TLB miss incurs an additional memory access (the PTE)

Fortunately, TLB misses are rare. Does a TLB miss require disk access?

Simple Memory System Example (small)

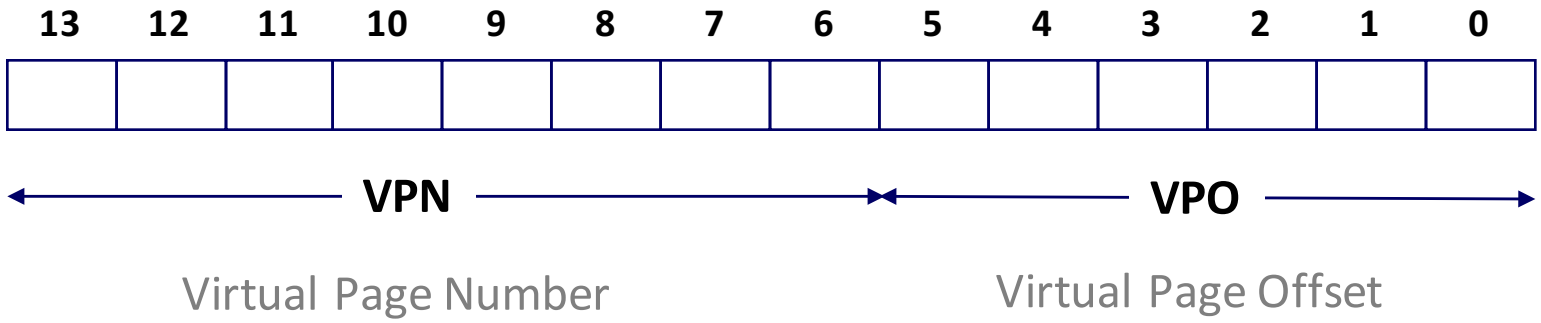
Addressing

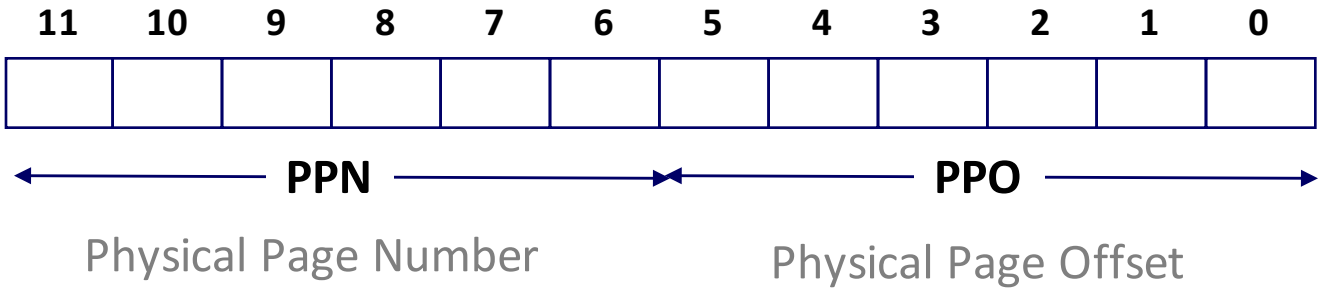
14-bit virtual addresses

12-bit physical address

Page size = 64 bytes

Simulate accessing these virtual addresses on the system: **`0x03D4, 0x0B8F, 0x0020`**





Simple Memory System Page Table

Only showing first 16 entries (out of 256 = 2^8)

virtual page #___ TLB index___ TLB tag ____ TLB Hit? __ Page Fault? __ physical page #: ____

VPN	*PPN*	*Valid*
00	28	1
01	–	0
02	33	1
03	02	1
04	–	0
05	16	1
06	–	0
07	–	0

VPN	*PPN*	*Valid*
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

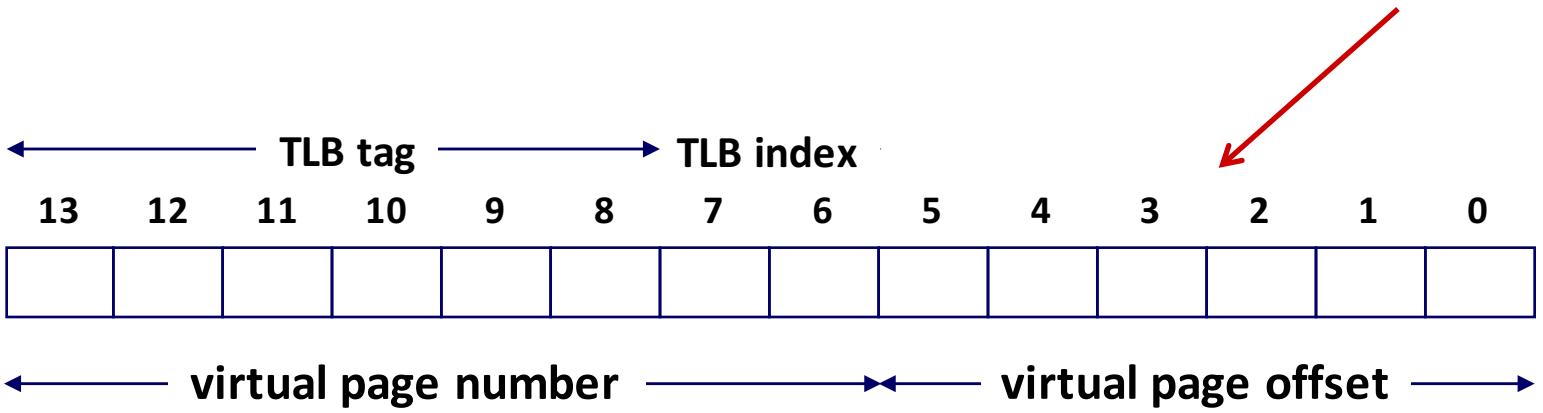
What about a real address space? Read more in the book…

Simple Memory System TLB

16 entries

4-way associative

TLB ignores page offset. Why?



virtual page #___ TLB index___ TLB tag ____ TLB Hit? __ Page Fault? __ physical page #: ____

Set	*Tag*	*PPN*	*Valid*	*Tag*	*PPN*	*Valid*	*Tag*	*PPN*	*Valid*	*Tag*	*PPN*	*Valid*
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

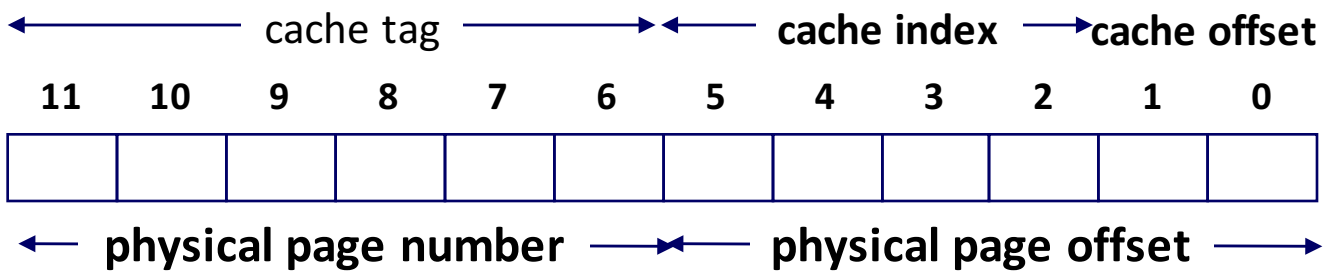
Simple Memory System Cache

16 lines

4-byte block size

Physically addressed

Direct mapped



cache offset___ cache index___ cache tag____ Hit? __ Byte: ____

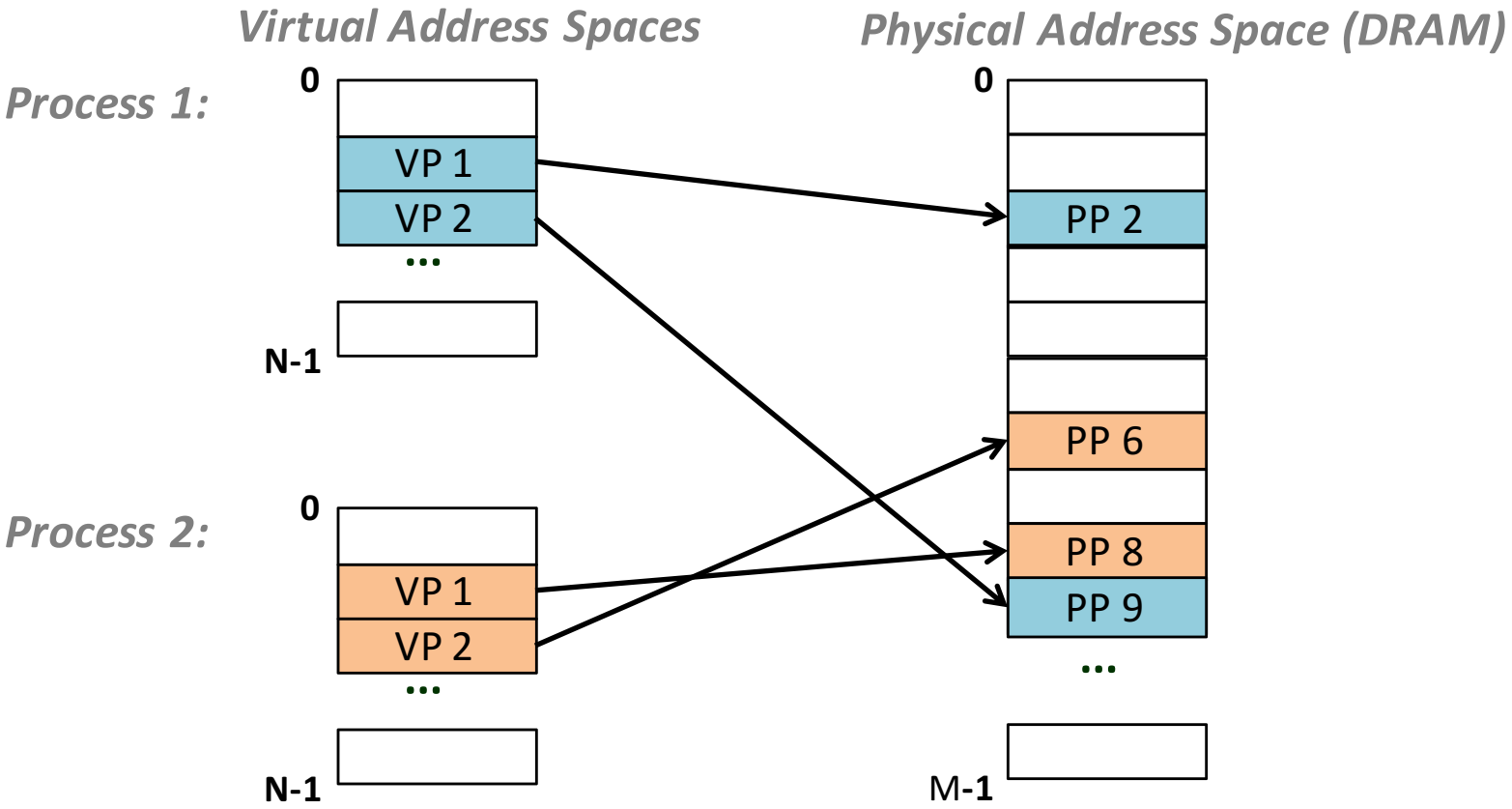
Idx	*Tag*	*Valid*	*B0*	*B1*	*B2*	*B3*
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03

Idx	*Tag*	*Valid*	*B0*	*B1*	*B2*	*B3*
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

Simple address space allocation

Process needs private ***contiguous*** address space.

Storage of virtual pages in physical pages is **fully associative.**



Simple cached access to storage > memory

Good locality, or least "small" working set = mostly page hits

All necessary
page table entries
fit in TLB

Working set pages
fit in physical memory

If combined working set > physical memory:

Thrashing: Performance meltdown. CPU always waiting or paging.

Full indirection quote:

"Every problem in computer science can be solved by adding another level of indirection, ***but that usually will create another problem***."

Freebies

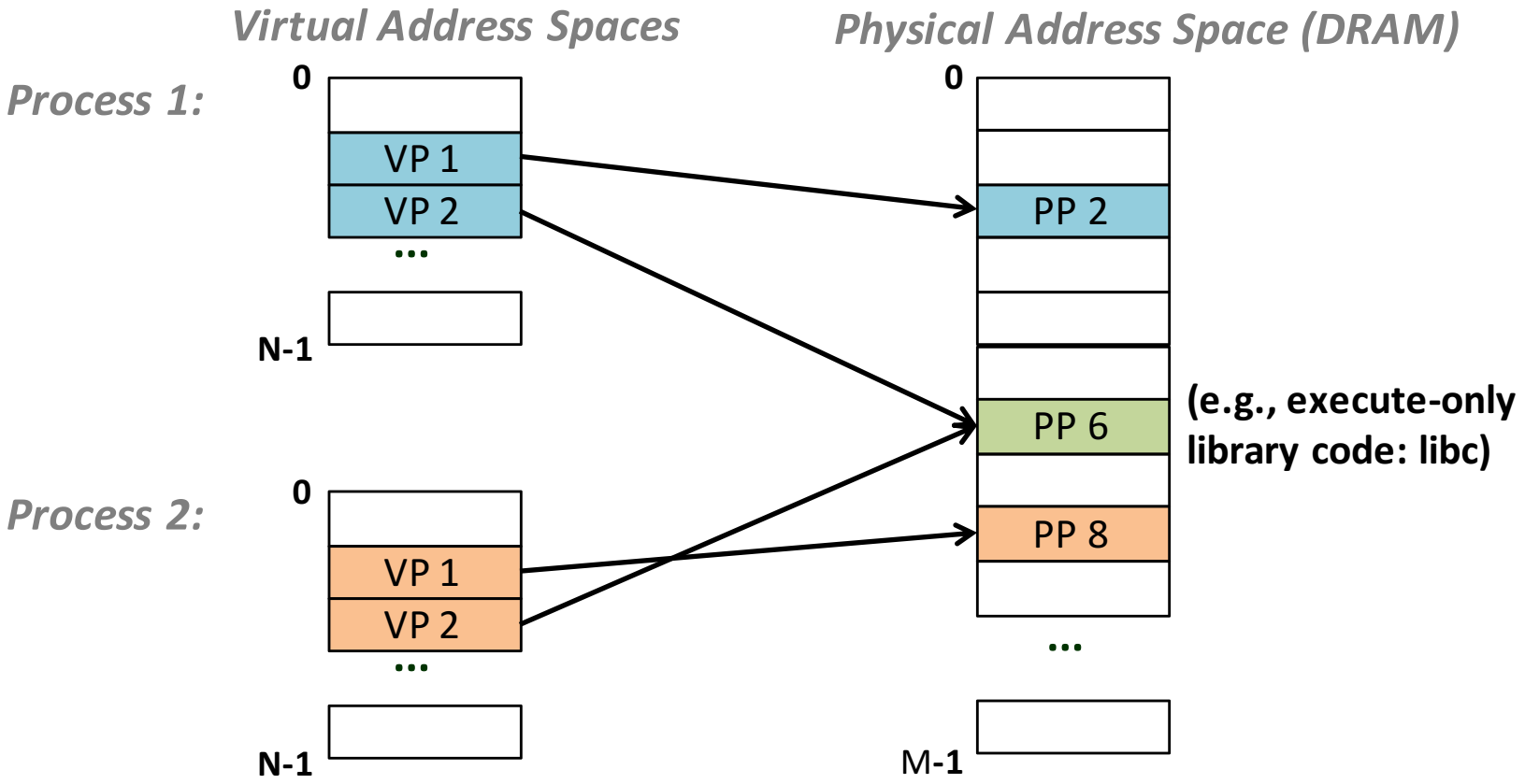
Protection:

All accesses go through translation.

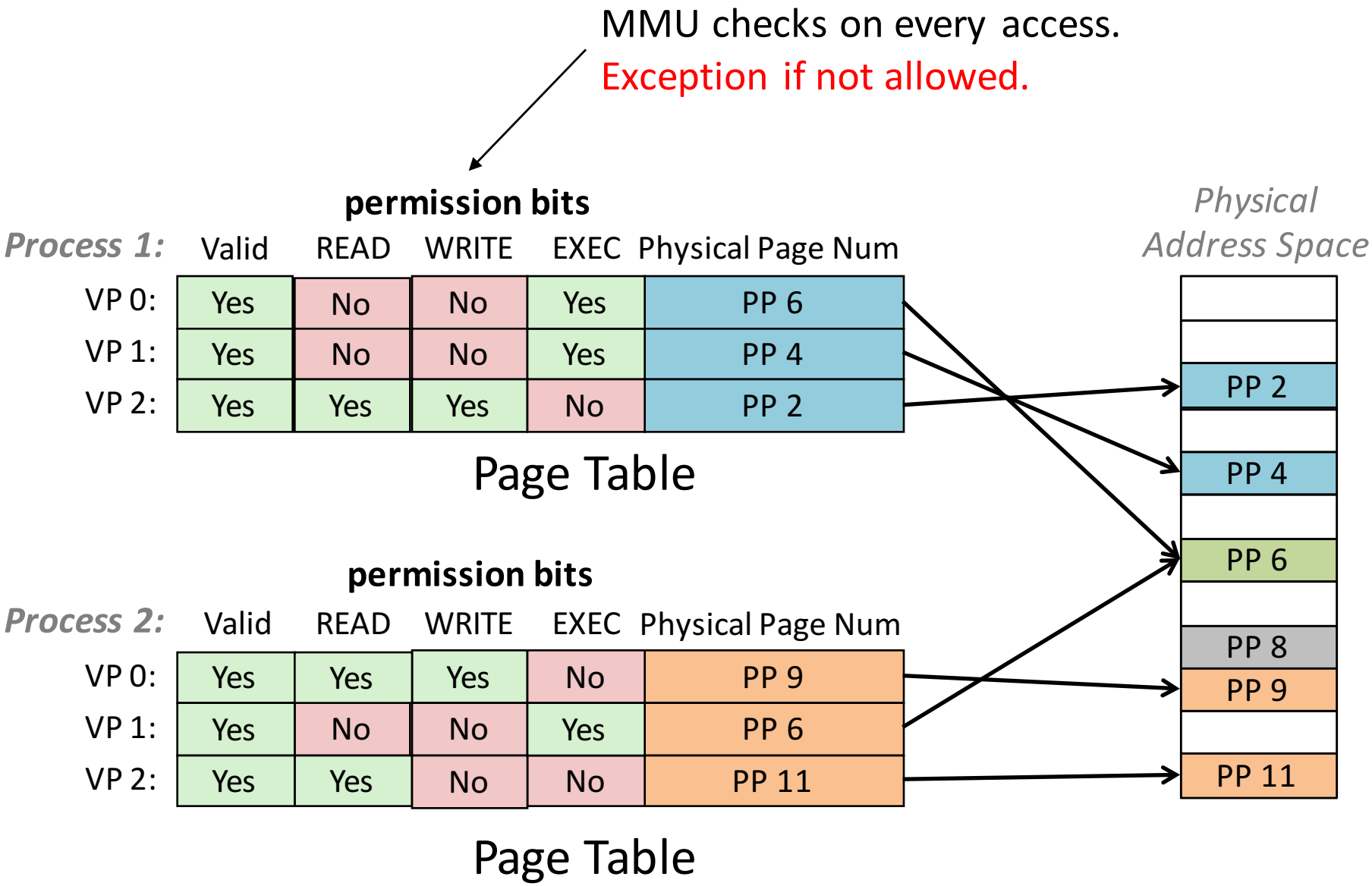
Impossible to access physical memory not mapped in virtual address space.

Sharing:

Map virtual pages in separate address spaces to same physical page *(PP 6).*



Memory permissions



How would you set permissions for the stack, heap, global variables, literals, code?

Summary: Virtual Memory

Programmer's view of virtual memory

System view of virtual memory

Summary: Memory Hierarchy

L1/L2/L3 Cache: Pure Hardware

Virtual Memory: Software-Hardware Co-design