

Virtual Memory

Hakim Weatherspoon

CS 3410, Spring 2012

Computer Science

Cornell University

P & H Chapter 5.4 (up to TLBs)

Goals for Today

Virtual Memory

- Address Translation
 - Pages, page tables, and memory mgmt unit
- Paging
- Role of Operating System
 - Context switches, working set, shared memory
- Performance
 - How slow is it
 - Making virtual memory fast
 - Translation lookaside buffer (TLB)
- Virtual Memory Meets Caching

Virtual Memory

Big Picture: Multiple Processes

How to Run multiple processes?

Time-multiplex a single CPU core (**multi-tasking**)

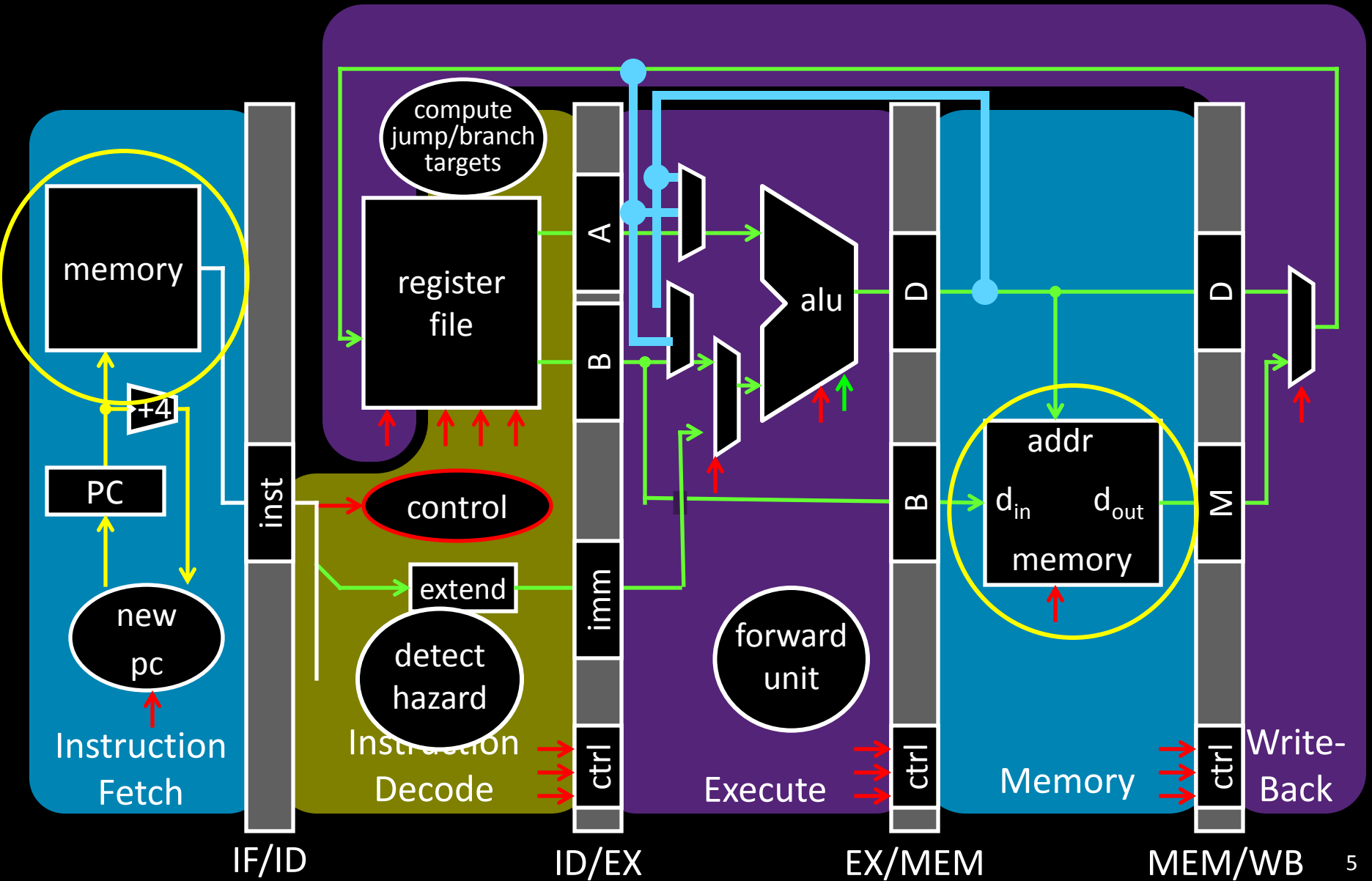
- Web browser, skype, office, ... all must co-exist

Many cores per processor (**multi-core**)
or many processors (**multi-processor**)

- Multiple programs run *simultaneously*

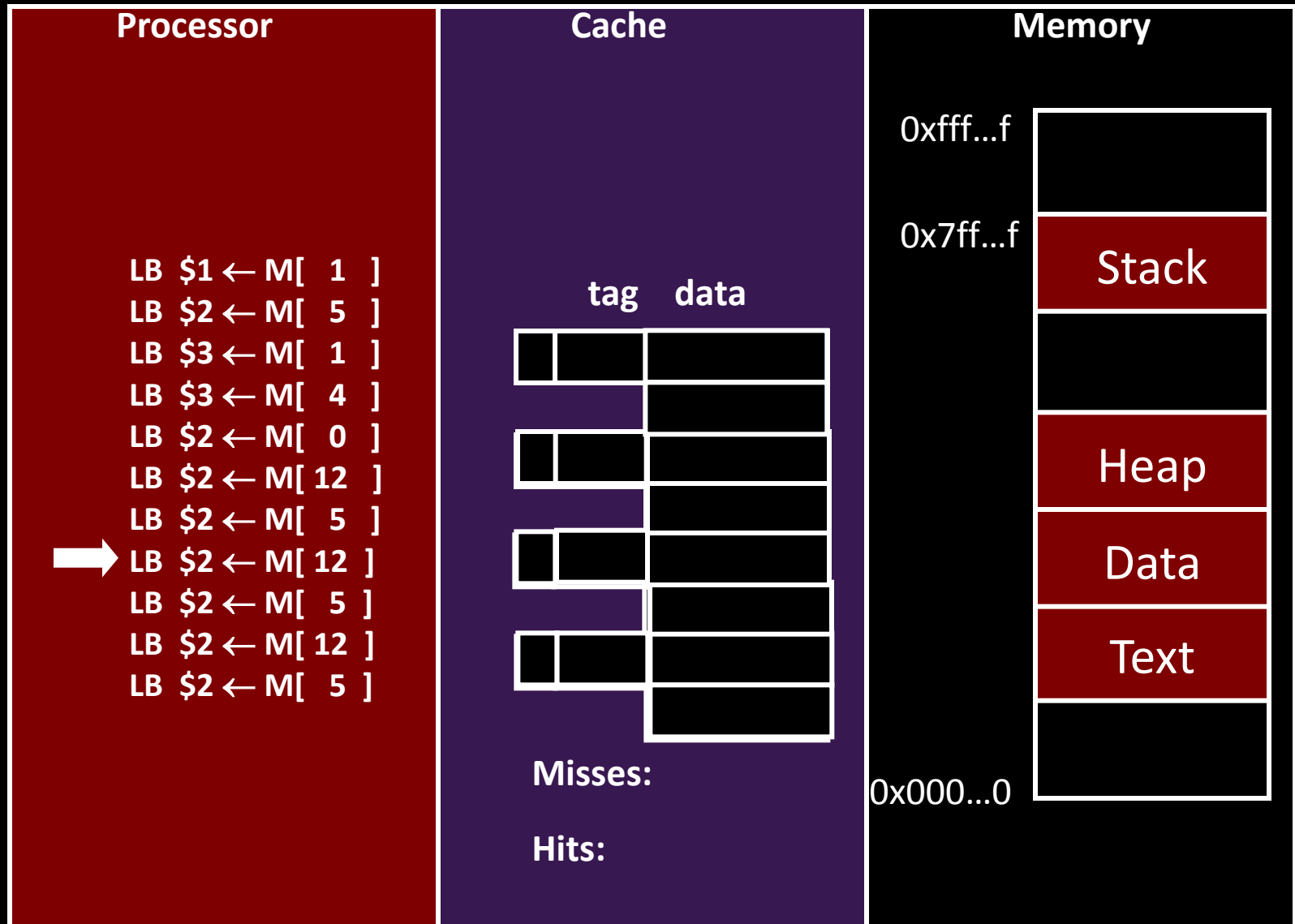
Big Picture: (Virtual) Memory

Memory: big & slow vs Caches: small & fast



Big Picture: (Virtual) Memory

Memory: big & slow vs Caches: small & fast



Processor & Memory

CPU address/data bus...

... routed through caches

... to main memory

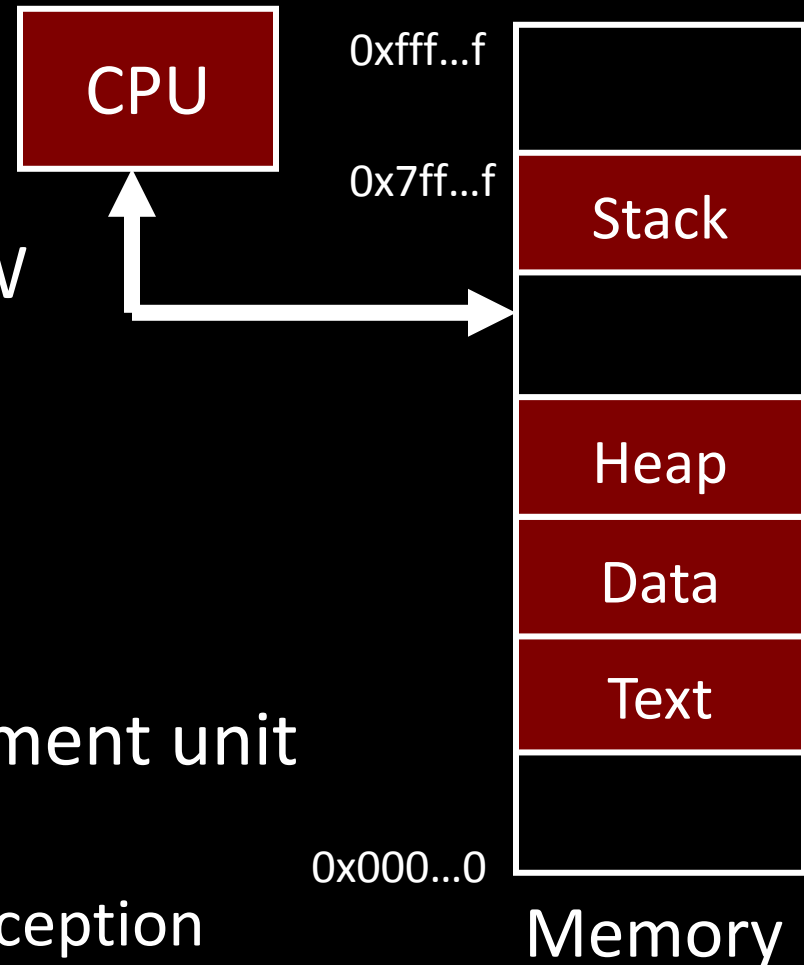
- Simple, fast, but...

Q: What happens for LW/SW to an invalid location?

- 0x00000000 (NULL)
- uninitialized pointer

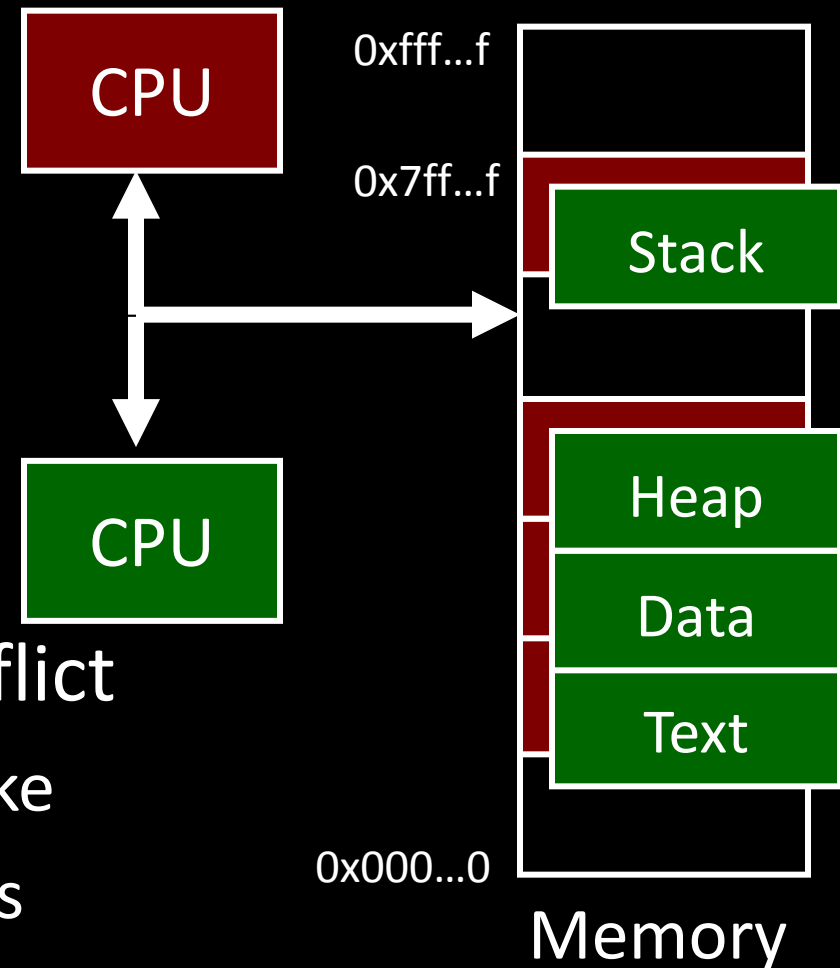
A: Need a memory management unit (MMU)

- Throw (and/or handle) an exception



Multiple Processes

Q: What happens when another program is executed concurrently on **another** processor?



A: The addresses will conflict

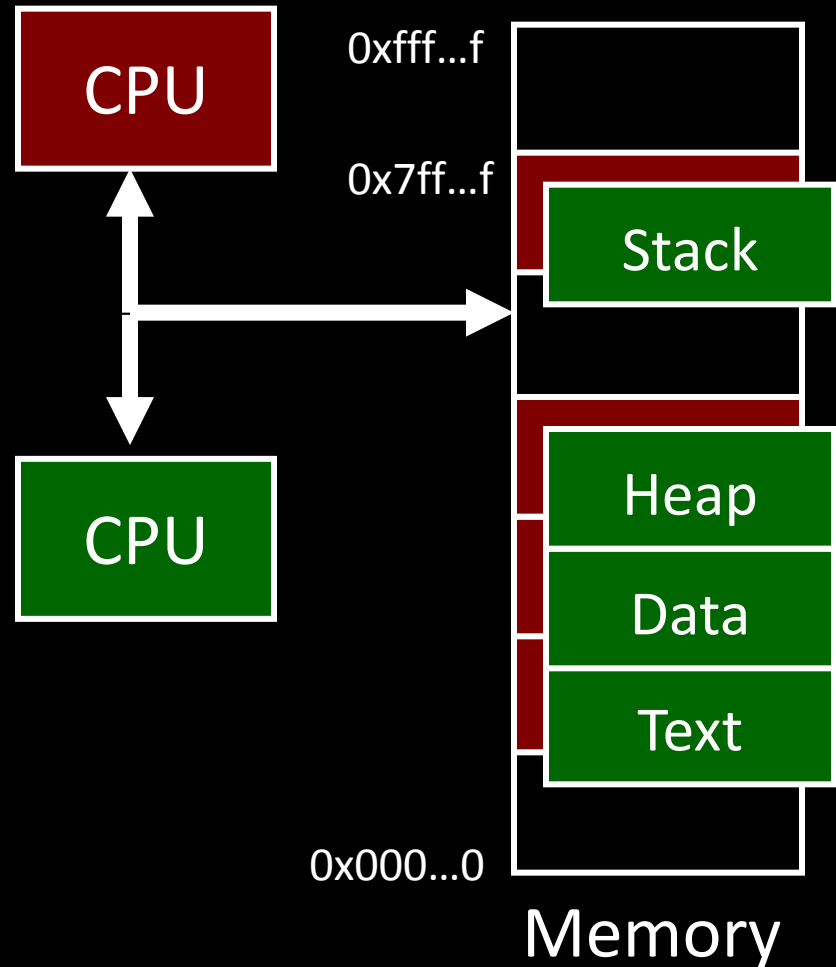
- Even though, CPUs may take turns using memory bus

Multiple Processes

Q: Can we relocate second program?

virtual \Rightarrow phys

map

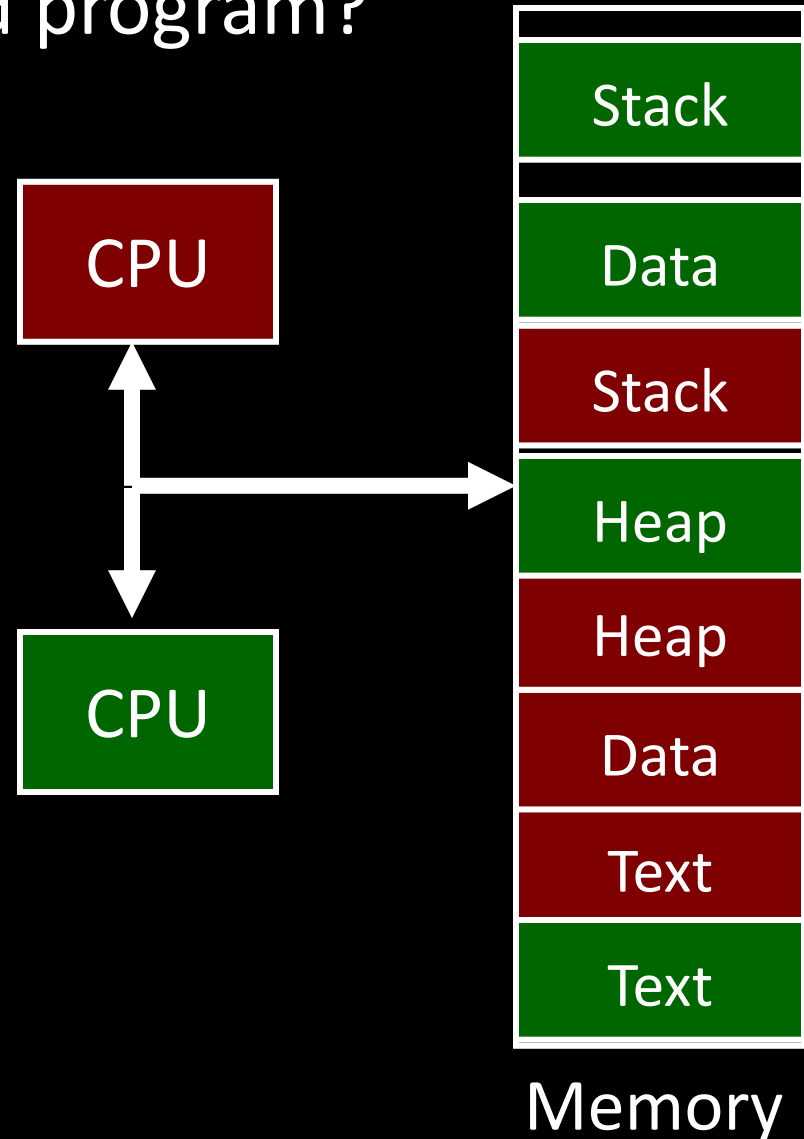


Solution? Multiple processes/processors

Q: Can we relocate second program?

A: Yes, **but...**

- What if they don't fit?
- What if not contiguous?
- Need to recompile/relink?
- ...



All problems in computer science can be solved by another level of indirection.

** map*

*virtual Addr
(generated by CPU)
to a phys addr
in mem*

- David Wheeler*
- or, Butler Lampson*
- or, Leslie Lamport*
- or, Steve Bellovin*

Virtual Memory

Virtual Memory: A Solution for All Problems

$0, \dots, 2^N$ ($N=32$)

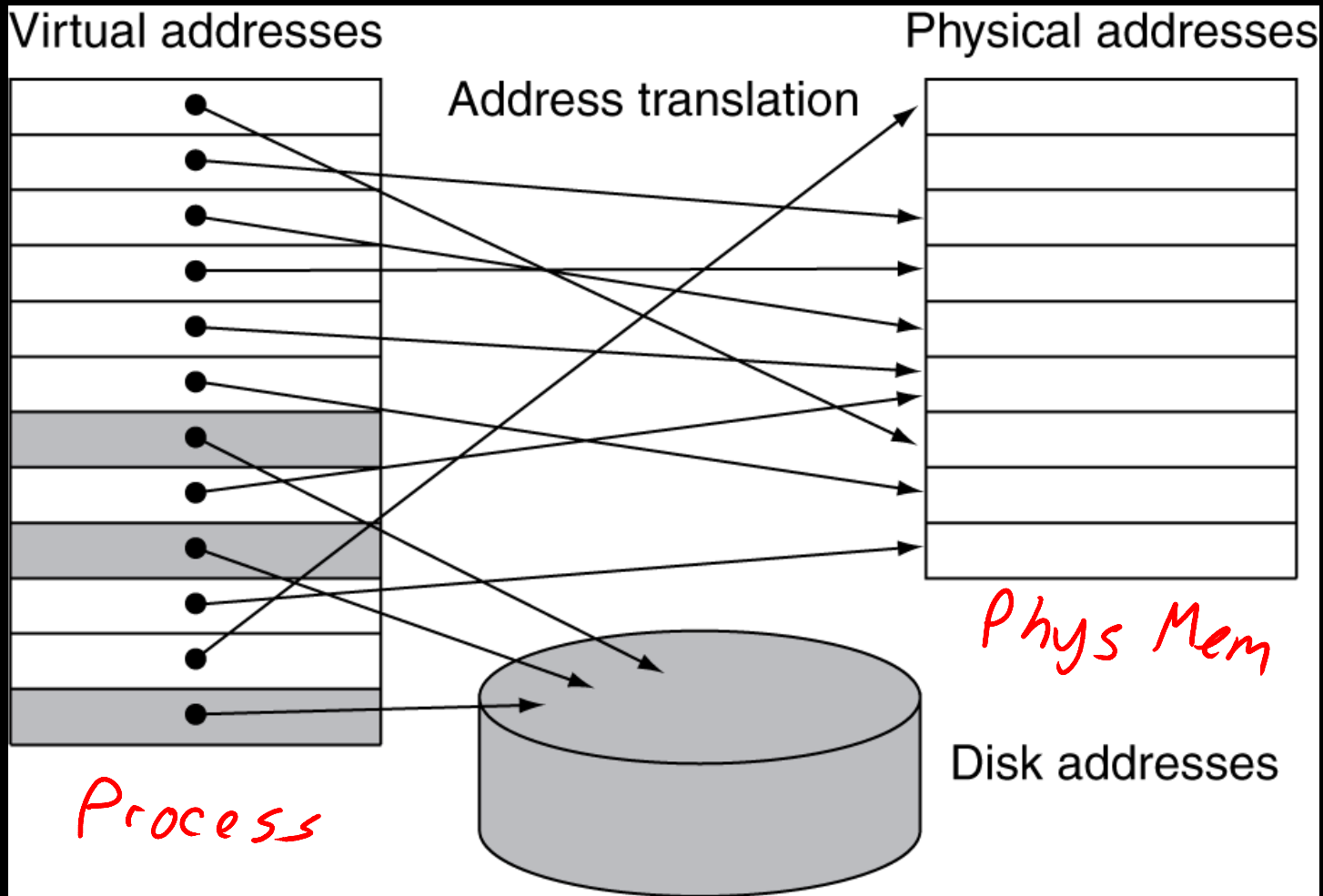
Each **process** has its own **virtual address space**

- Programmer can code as if they own all of memory

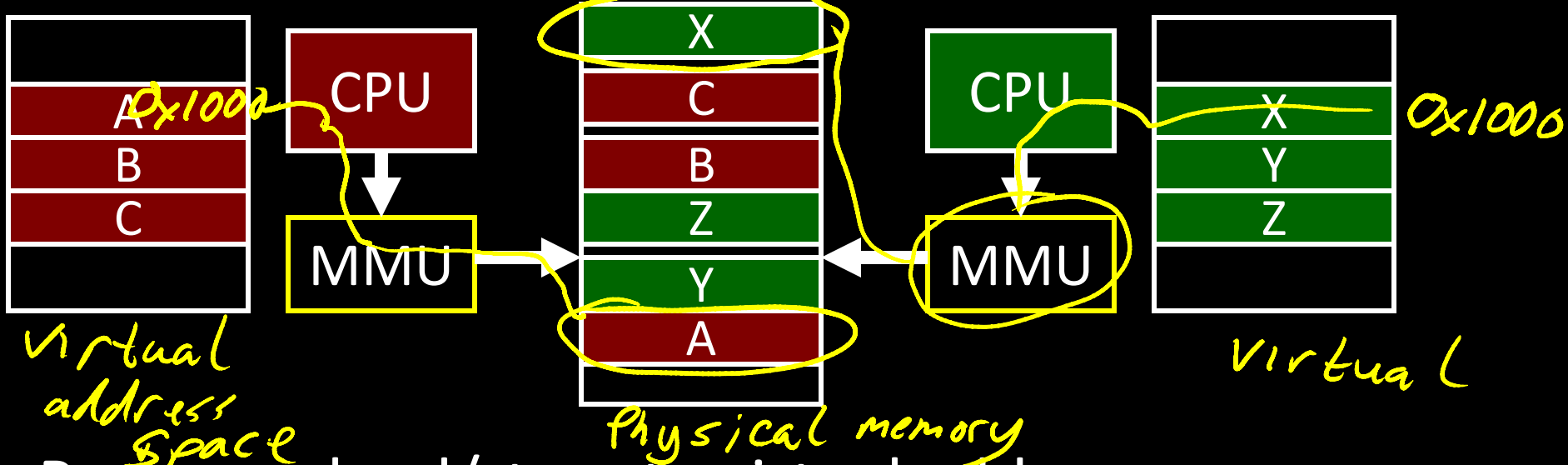
On-the-fly at runtime, for each memory access

- all access is *map* **indirect** through a virtual address
- **translate fake virtual address to a real physical address**
- ~~redirect load/store to the physical address~~

Address Space



Address Space



Programs load/store to virtual addresses

Actual memory uses physical addresses

Memory Management Unit (MMU)

- Responsible for translating on the fly
- Essentially, just a big array of integers:

```
paddr = PageTable[vaddr];
```

Virtual Memory Advantages

Advantages

Easy relocation

- Loader puts code anywhere in physical memory
- Creates **virtual mappings** to give illusion of correct layout

Higher memory utilization

- Provide illusion of contiguous memory
- Use all physical memory, even physical address 0x0

Easy sharing

- Different mappings for different programs / cores

And more to come...

Address Translation

Pages, Page Tables, and
the Memory Management Unit (MMU)

Address Translation

Attempt #1: How does MMU translate addresses?

`paddr = PageTable[vaddr];` ★

Granularity?

- Per word...
- Per block...
- Variable.....

$$2^{32} \Rightarrow 4GB$$

→ 1 billion word table
Each PT entry is 4 bytes
PT = 4GB

Typical:

- 4KB – 16KB pages — 14 bits
- 4MB – 256MB jumbo pages

32-bit addr

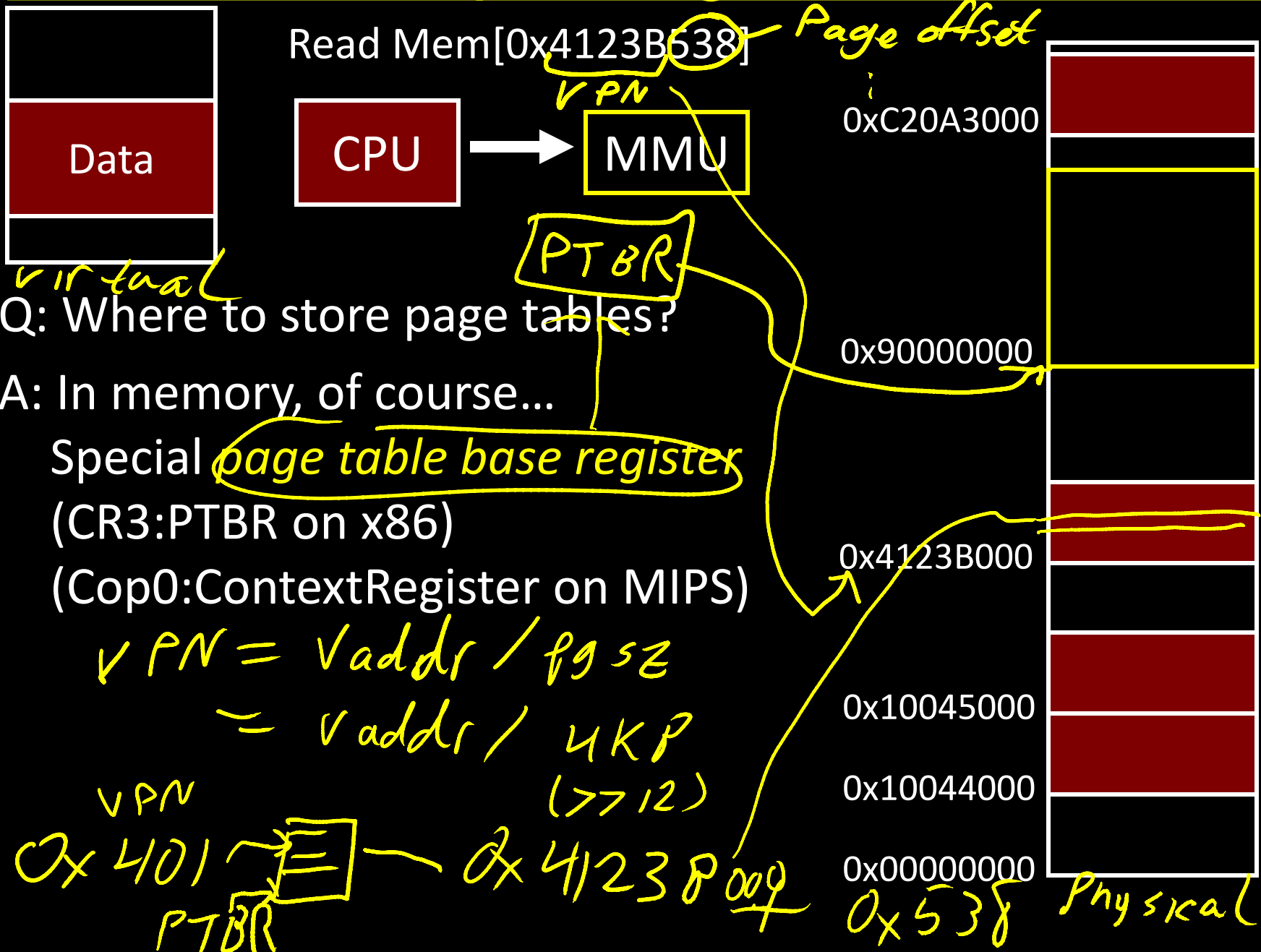
14-bits word w/n Page

$$32 - 14 = 18 \text{ bits}$$

2^{18} PT entries

⇒ PT = $2^{20} = 1MB$
each entry is 4 bytes
PT vaddr index

Simple PageTable



Simple PageTable

Physical Page Number

	0x10045 ●
	<i>0x777</i>
	0xC20A3 ●
	0x4123B ●
	0x10044 ●

0xC20A3000

0x90000000

0x4123B000

0x10045000

0x10044000

0x00000000

31

0

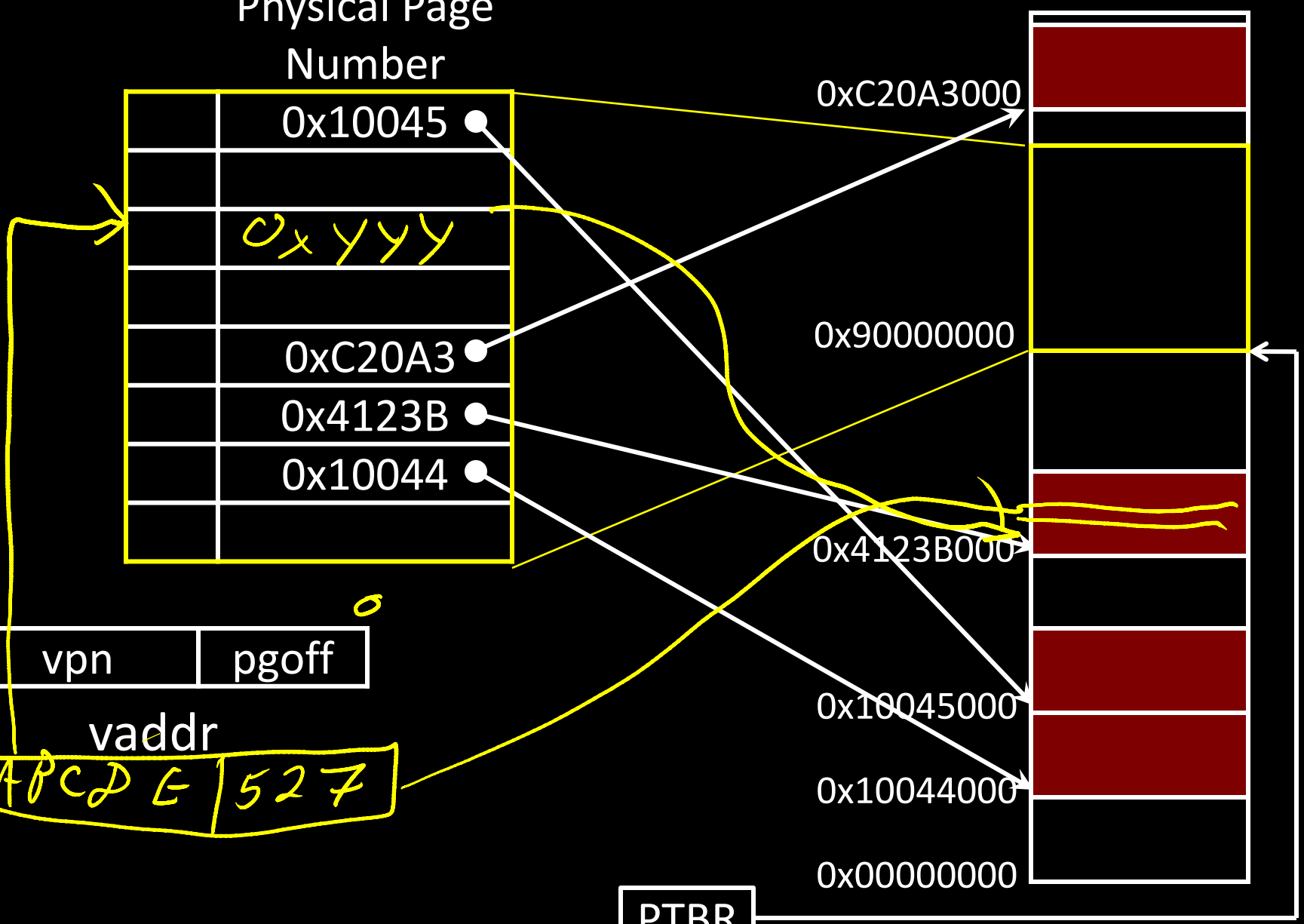
vpn	pgoff
-----	-------

vaddr

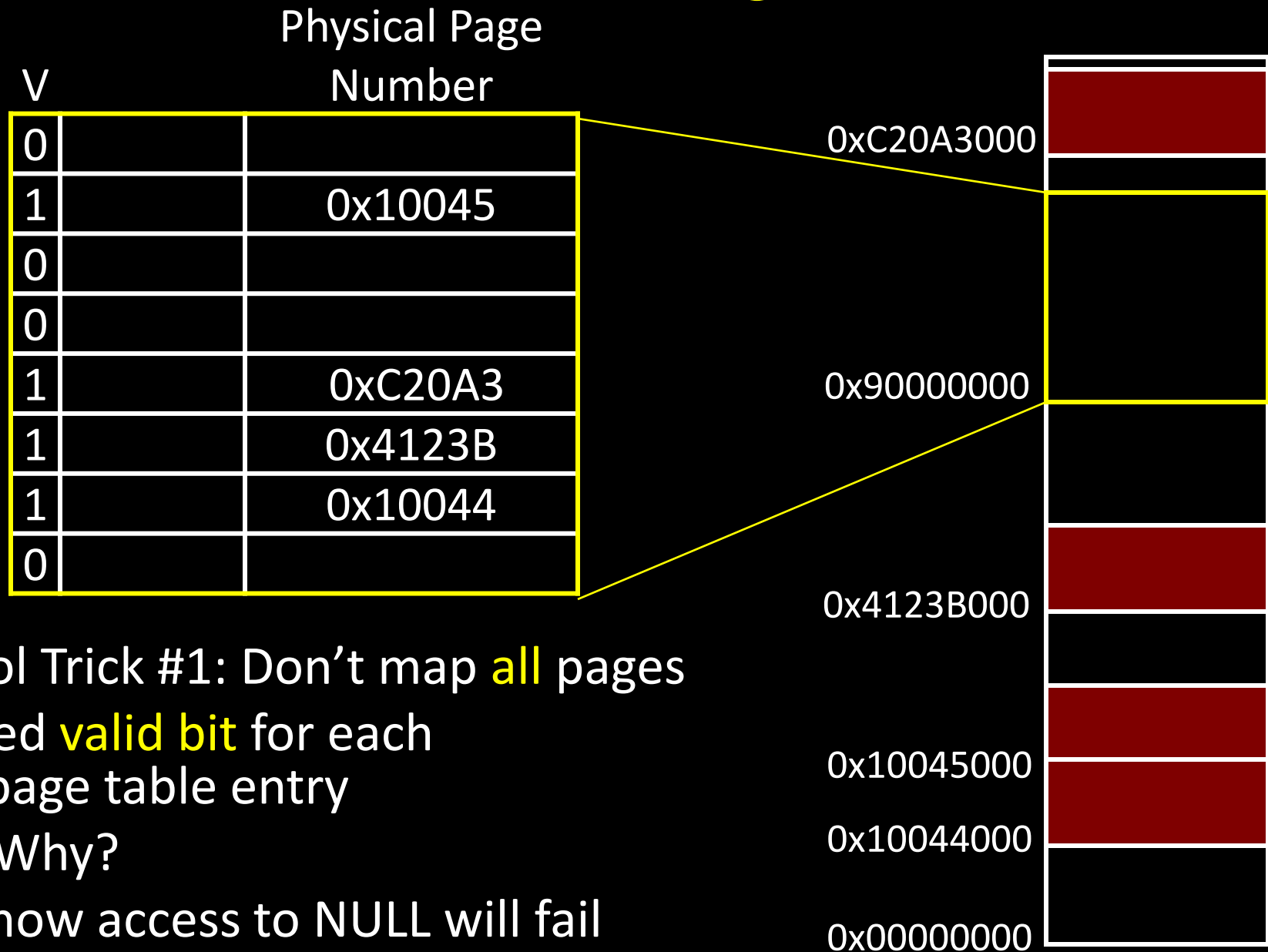
<i>0xABCDE</i>	<i>527</i>
----------------	------------

0

PTBR



Invalid Pages



Cool Trick #1: Don't map **all** pages

Need **valid bit** for each page table entry

Q: Why?

A: now access to NULL will fail

A: we might not have that much physical memory

Page Size Example

Overhead for VM Attempt #1 (example)

Virtual address space (for each process):

- total memory: 2^{32} bytes = 4GB
- page size: 2^{12} bytes = 4KB
- entries in PageTable?
- size of PageTable?

$$\frac{2^{32}}{2^{12}} = 2^{20}$$

2^{20} PTE

each entry is 4 bytes
(including valid)

$$2^{22} \text{ B} = 4 \text{ MB}$$

Physical address space:

- total memory: 2^{29} bytes = 512MB
- overhead for 10 processes?

if 10 processes

use 40 MB

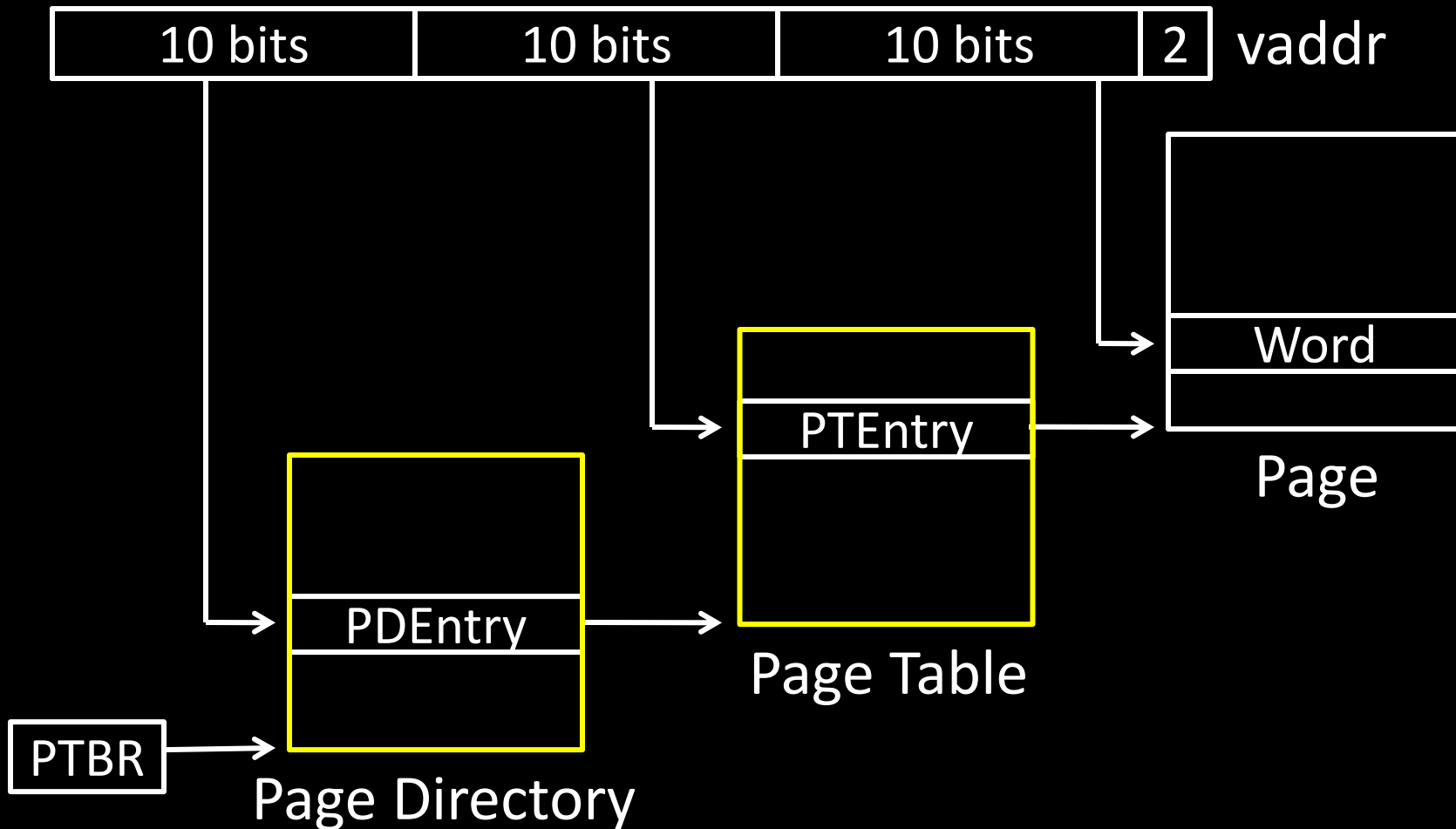
for PTEs

$$\frac{40}{512} = 8-9\%$$

Beyond Flat Page Tables

Assume most of PageTable is empty

How to translate addresses? Multi-level PageTable



* x86 does exactly this

Beyond Flat Page Tables

Assume most of PageTable is empty

How to translate addresses? Multi-level PageTable

Q: Benefits?

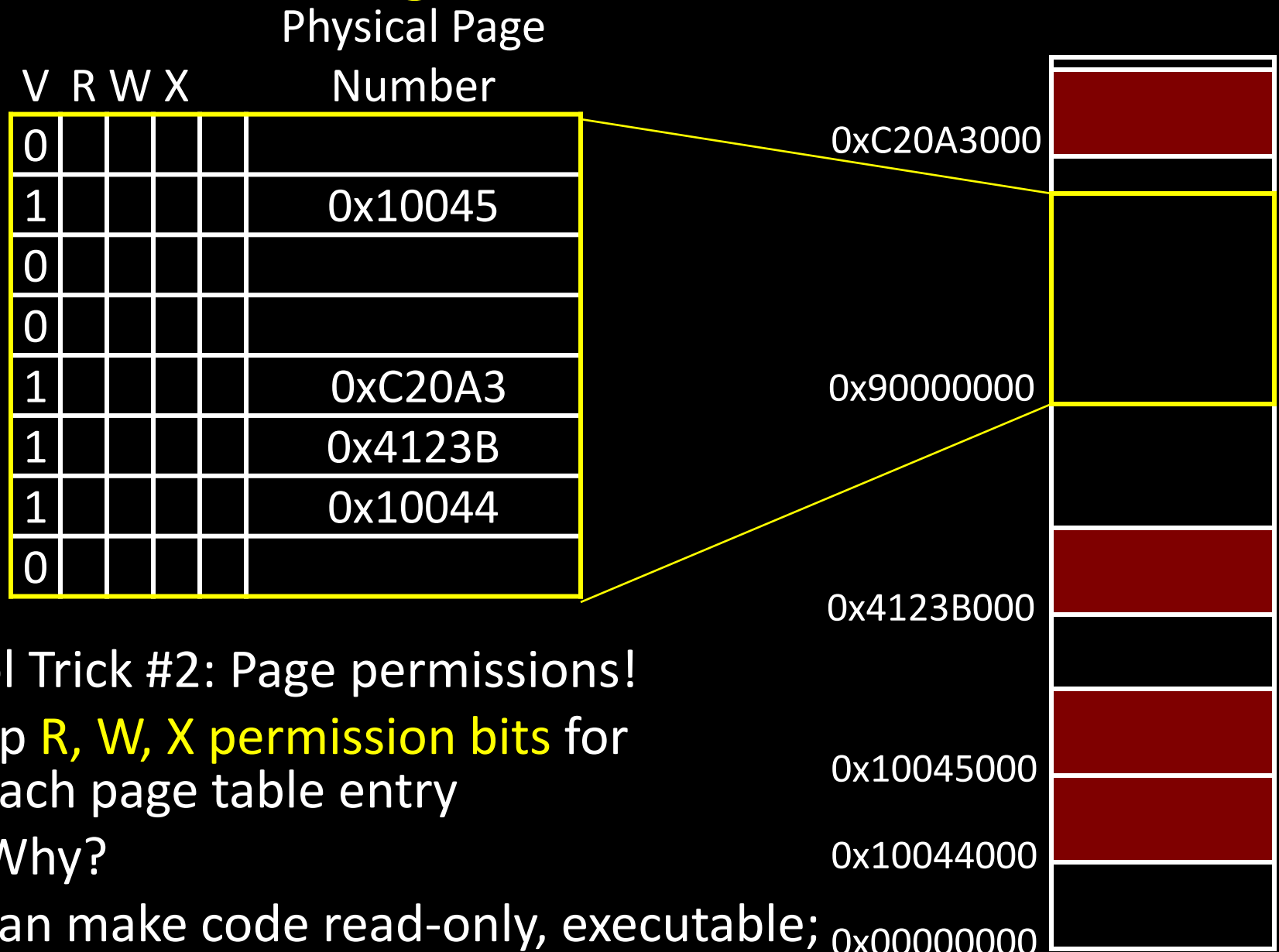
A: Don't need 4MB contiguous physical memory

A: Don't need to allocate every PageTable, only those containing valid PTEs

Q: Drawbacks

A: Performance: Longer lookups

Page Permissions



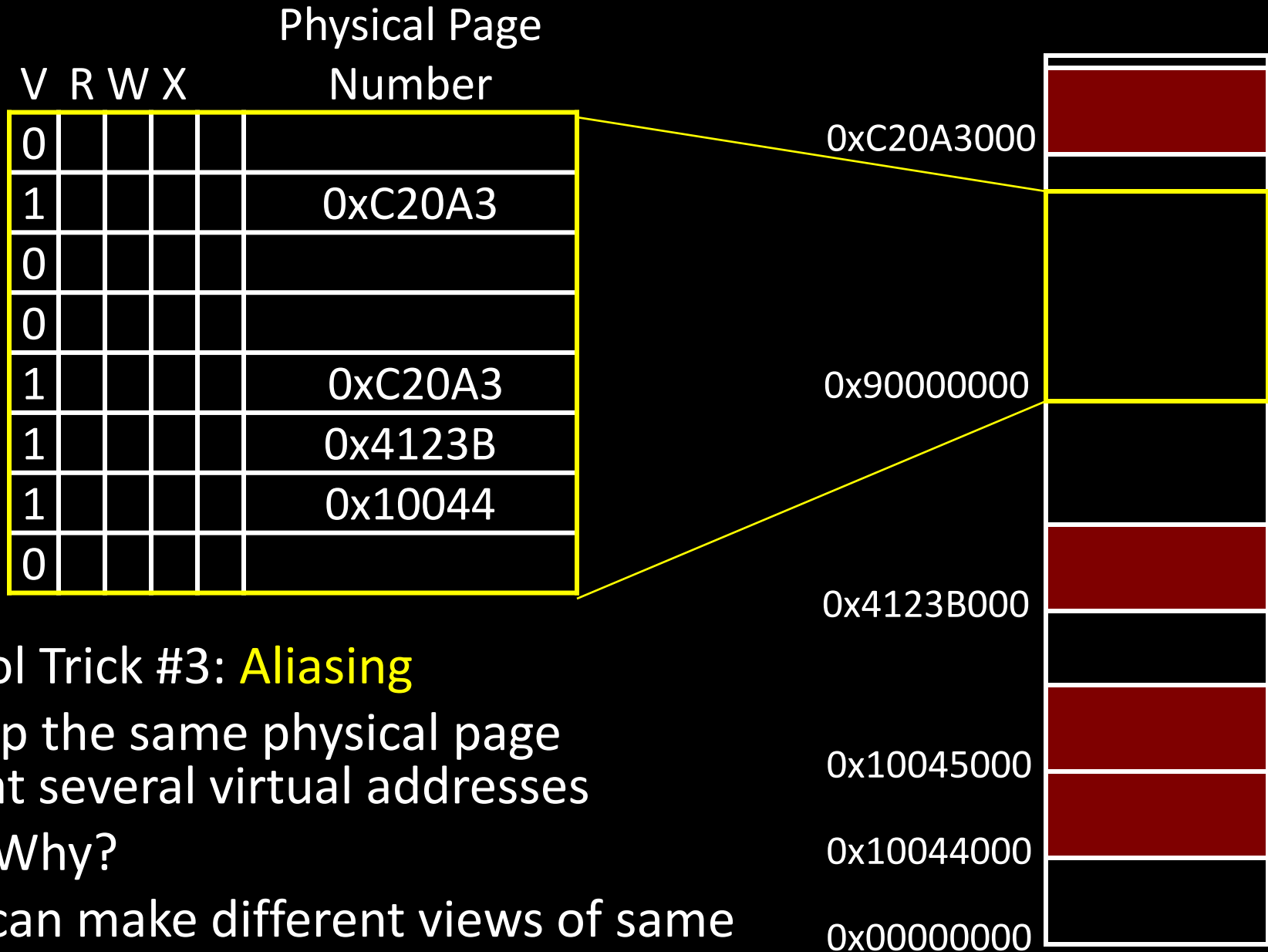
Cool Trick #2: Page permissions!

Keep **R, W, X permission bits** for each page table entry

Q: Why?

A: can make code read-only, executable; make data read-write but not executable; etc.

Aliasing



Cool Trick #3: **Aliasing**

Map the same physical page
at several virtual addresses

Q: Why?

A: can make different views of same
data with different permissions

Paging

Paging

Can we run process larger than physical memory?

- The “virtual” in “virtual memory”

View memory as a “cache” for secondary storage

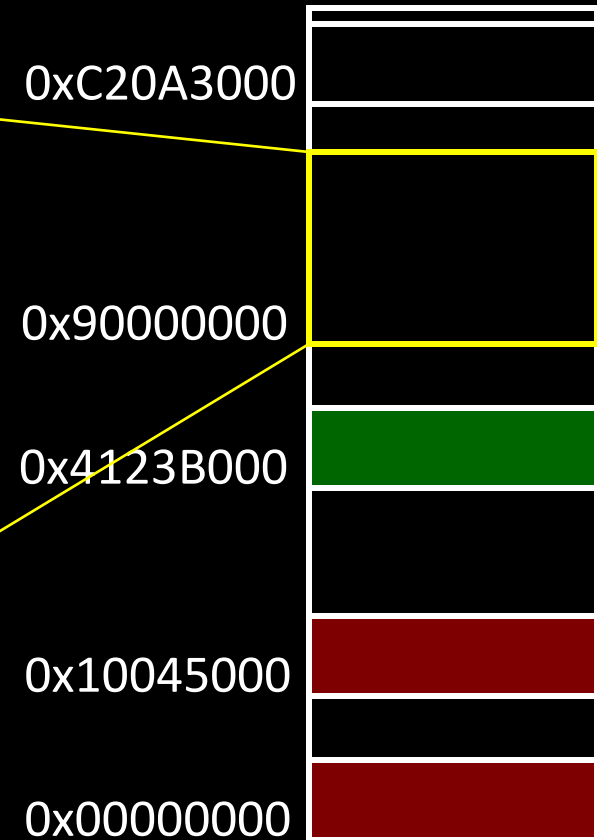
- **Swap** memory pages out to disk when not in use
- **Page** them back in when needed

Assumes Temporal/Spatial Locality

- Pages used recently most likely to be used again soon

Paging

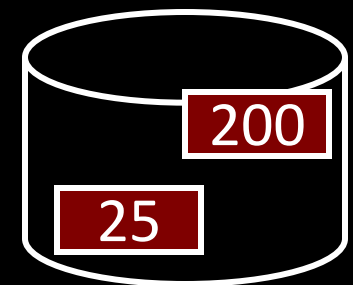
Physical Page					
V	R	W	X	D	Number
0					invalid
1				0	0x10045
0					invalid
0					invalid
0				0	disk sector 200
0				0	disk sector 25
1				1	0x00000
0					invalid



Cool Trick #4: **Paging/Swapping**

Need more bits:

Dirty, RecentlyUsed, ...



Summary

Virtual Memory

- Address Translation
 - Pages, page tables, and memory mgmt unit
- Paging

Next time

- Role of Operating System
 - Context switches, working set, shared memory
- Performance
 - How slow is it
 - Making virtual memory fast
 - Translation lookaside buffer (TLB)
- Virtual Memory Meets Caching

Administrivia

Lab3 is due next Monday

- Extra Lab Help session:
- Saturday, April 7th, 12:00-2:30pm in CSUGLab

HW5 is due *next* Tuesday

- Download updated version. Use updated version.
- Online Survey available later this evening