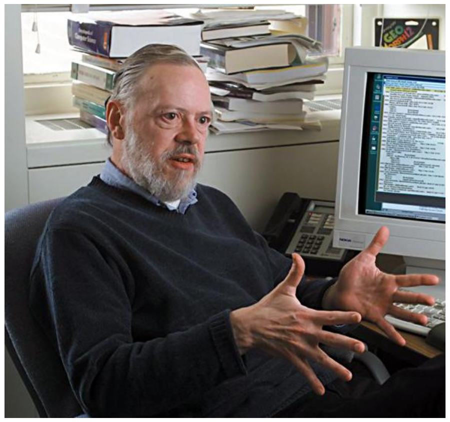
VM Internals

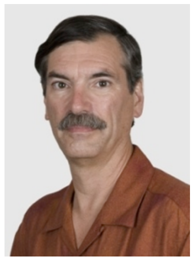
- Topics
  - Hardware support for virtual memory
- Learning Objectives:
  - Specify what the hardware must provide to enable virtual memory.
  - Discuss the hardware mechanisms that make virtual memory perform well.

OS Designer Humor

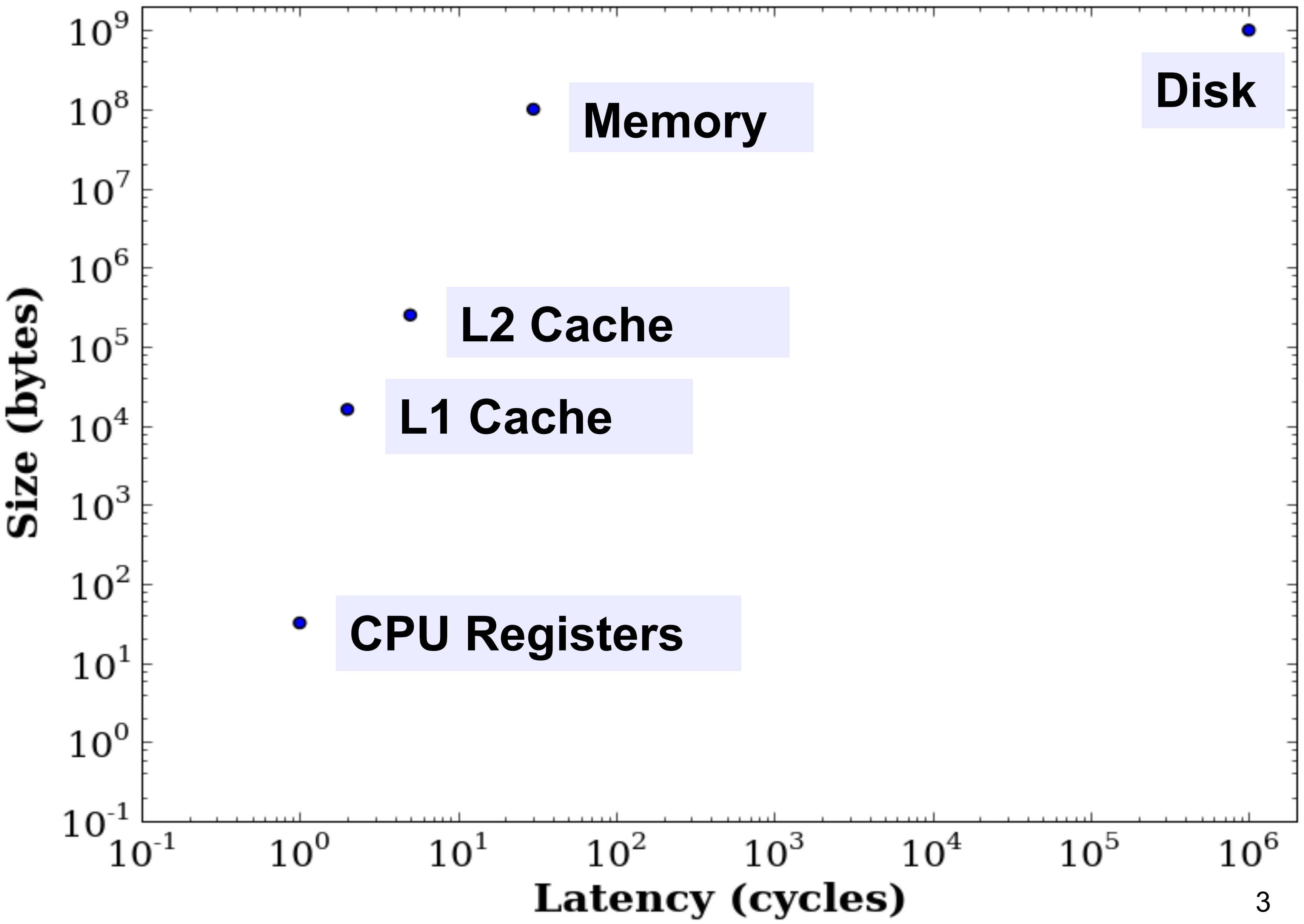
What do we usually do to improve performance?



Well, we could try a cache…



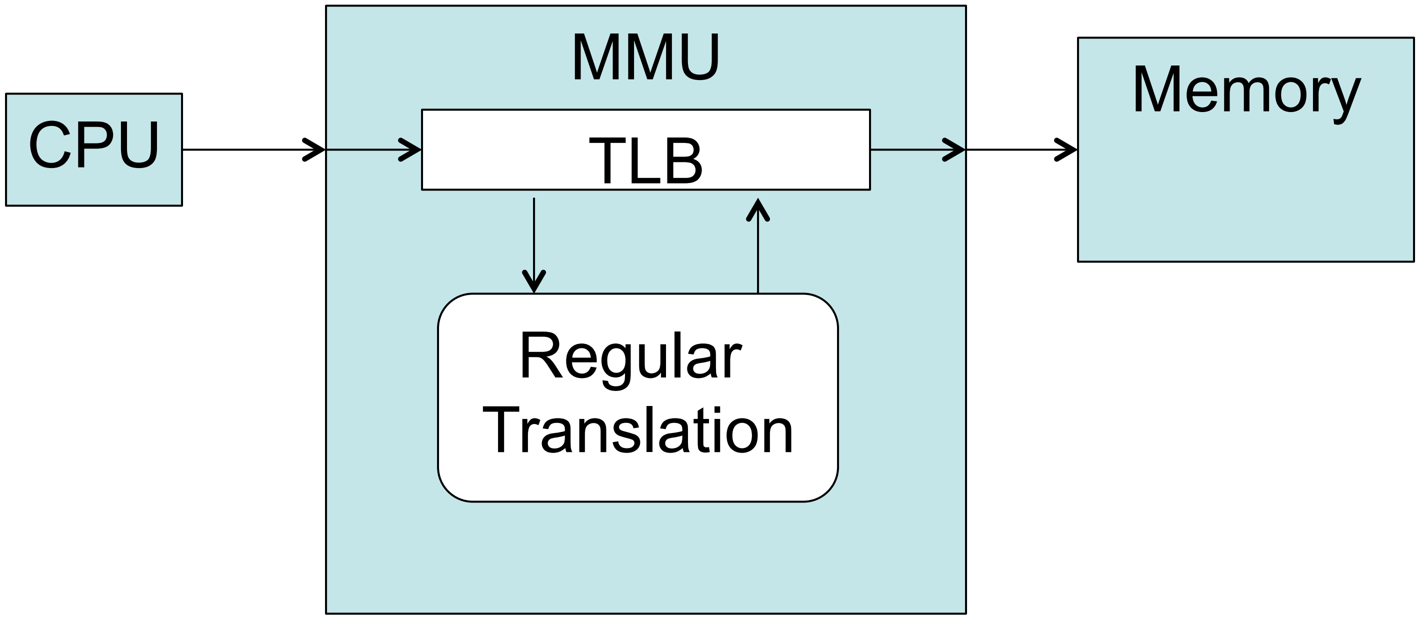
Everything's a Cache



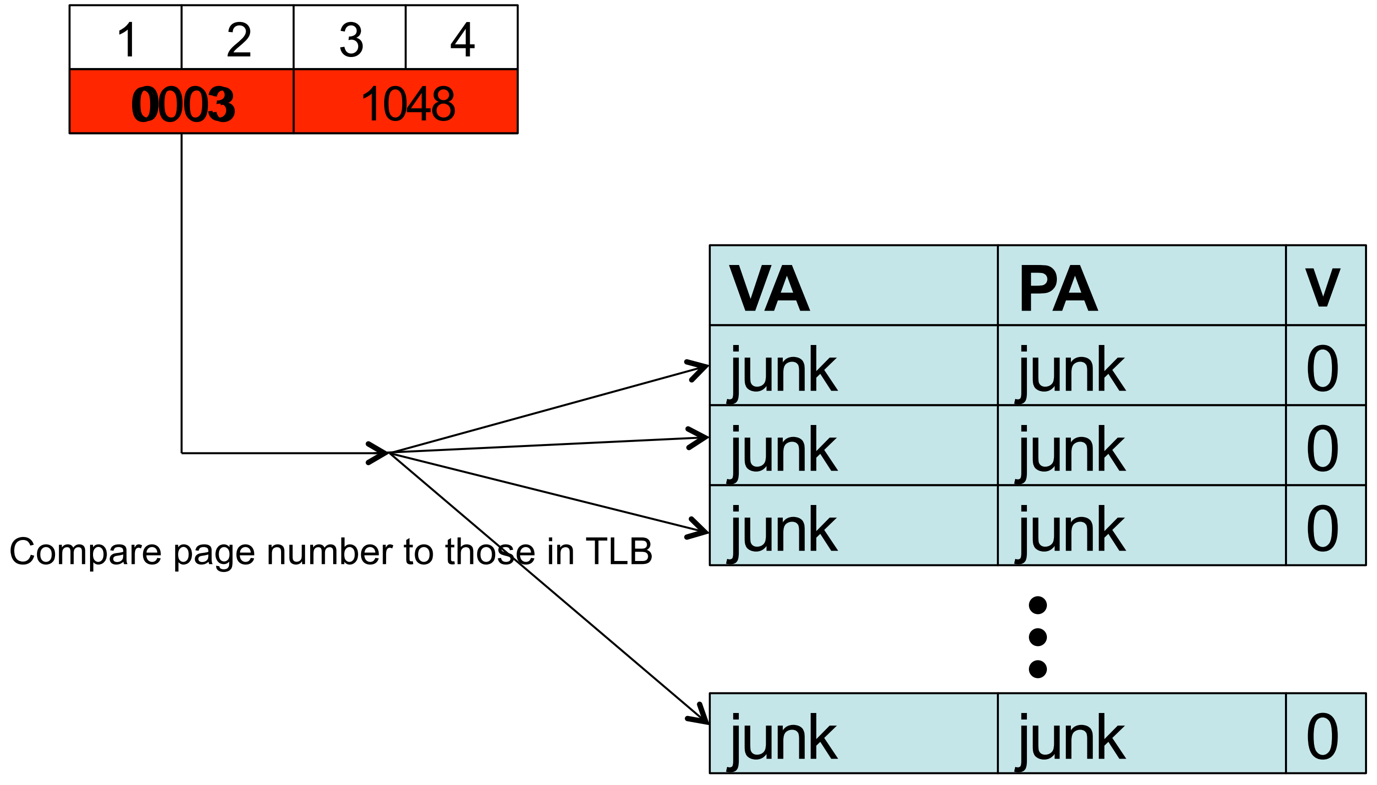
A Virtual Address Cache

- A translation lookaside buffer (TLB) is a cache of virtual to physical address translation.
  - Implemented in hardware
  - It contains a few tens to a few hundreds of mappings between virtual and physical addresses.
- But a process may have many, many more mappings, how can such a small number help us?

The MMU

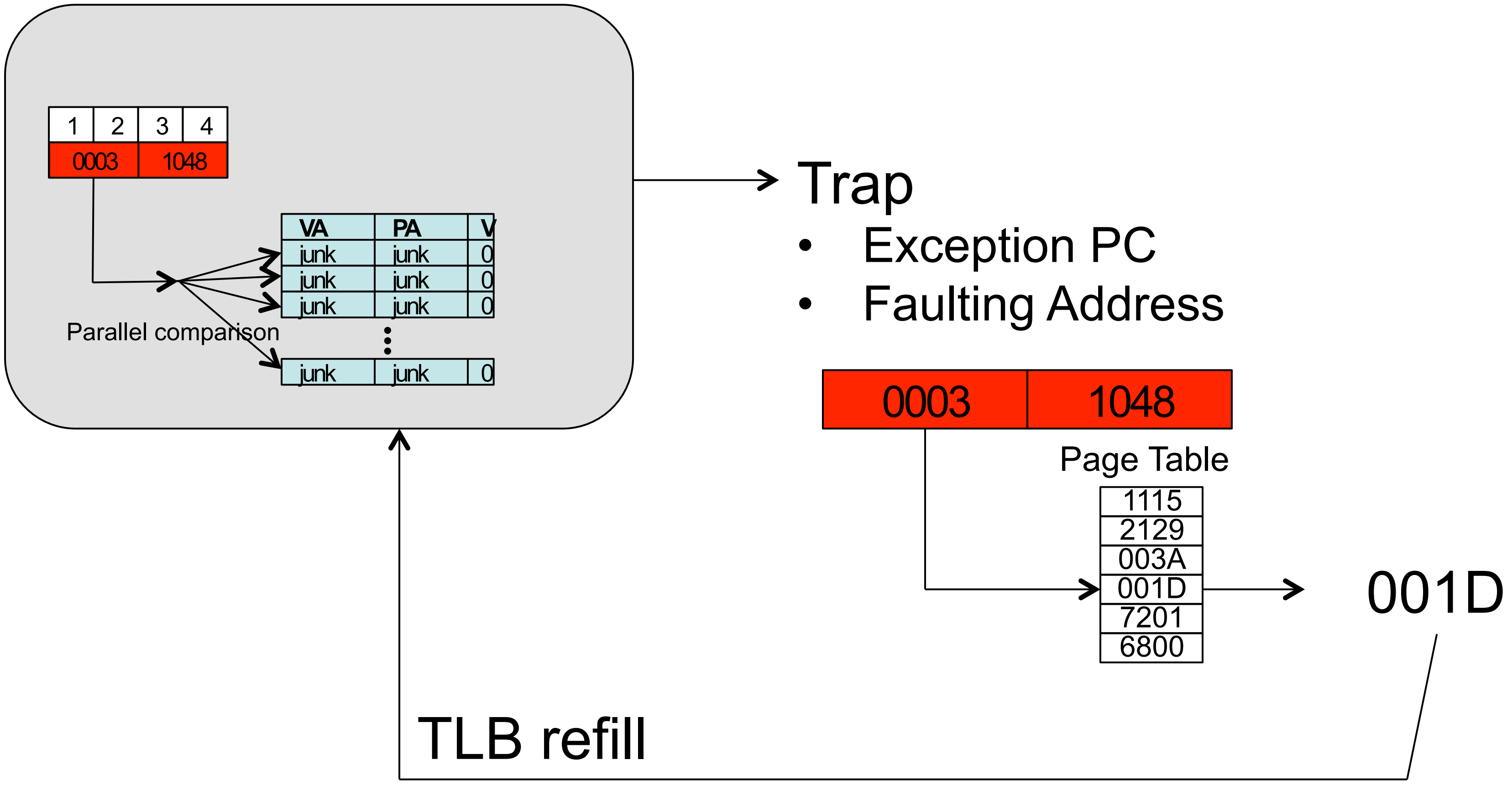


The TLB

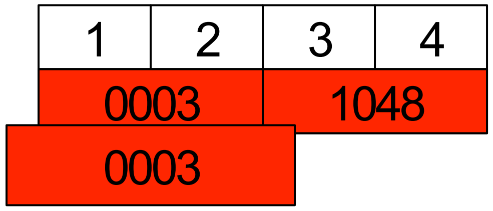


Miss! $\longrightarrow$ Trap!

A TLB Miss (in software)



TLB Refill

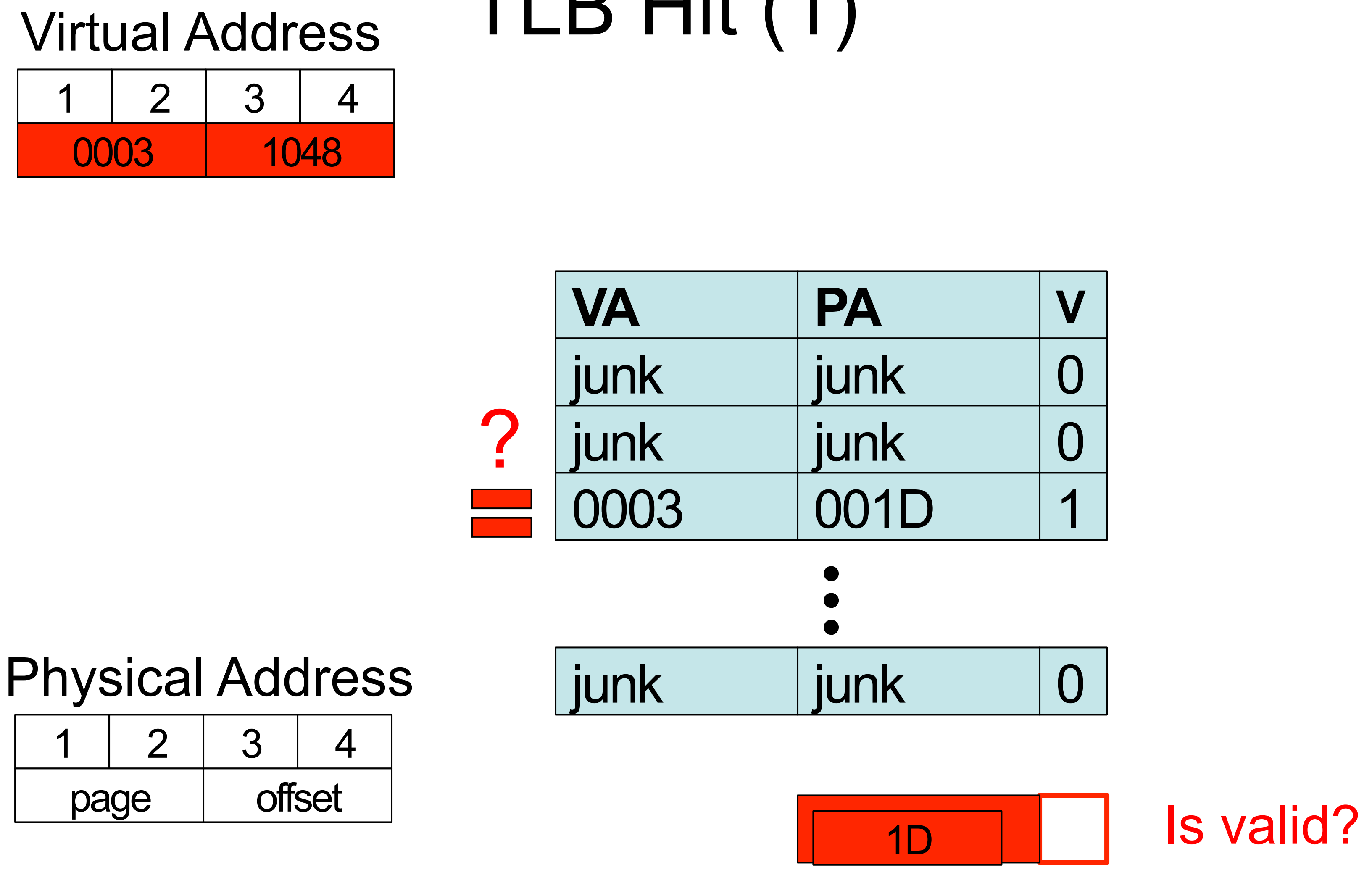


VA	PA	V
junk	junk	0
junk	junk	0
junk	junk	1
⋮		
junk	junk	0

From mapping

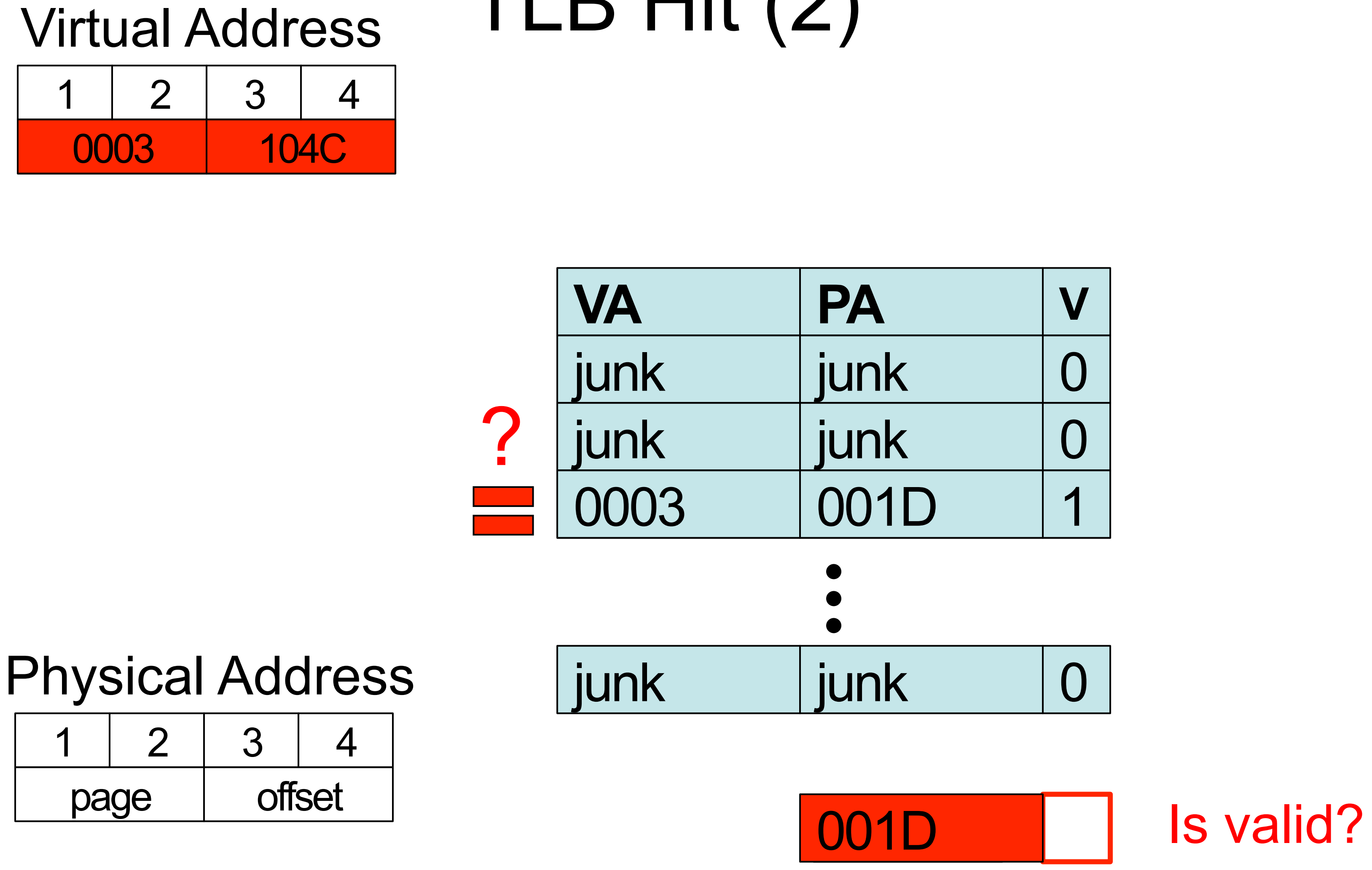
001D

TLB Hit (1)

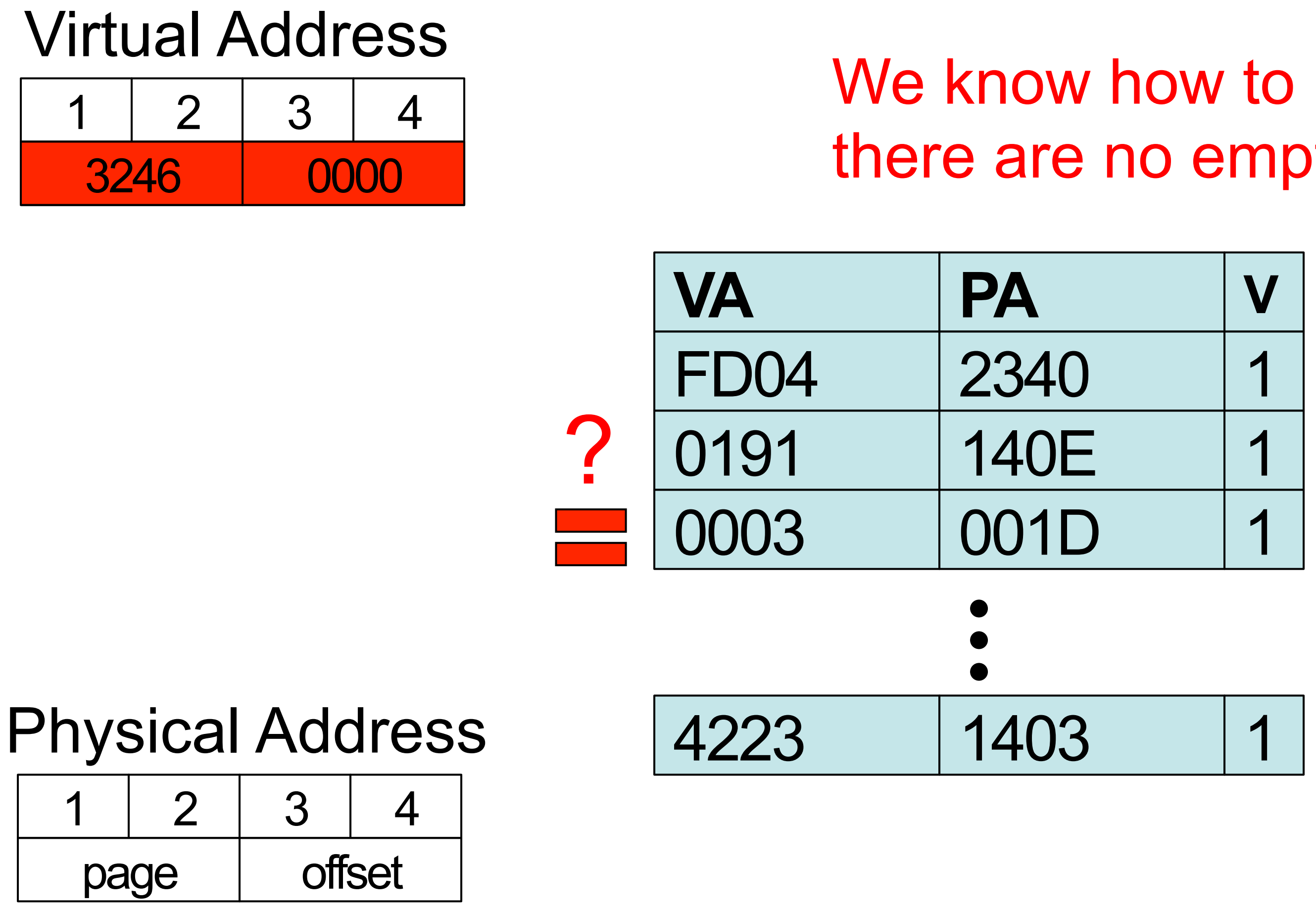


VA	PA	V
junk	junk	0
junk	junk	0
0003	001D	1
⋮		
junk	junk	0

TLB Hit (2)


Virtual Address
1
2
3
4
0003
104C
?
=
VA
PA
V
junk
junk
0
junk
junk
0
0003
001D
1
junk
junk
0
Physical Address
1
2
3
4
page
offset
001D
Is valid?

When your TLB Fills …



We know how to refill, but there are no empty spots!

VA	PA	V
FD04	2340	1
0191	140E	1
0003	001D	1
⋮		
4223	1403	1

TLB Eviction

- How do we decide which entry to evict?
  - Goal is to evict something you are unlikely to need soon.
    - Thoughts?

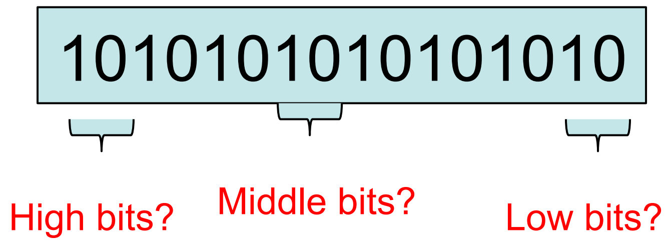
- We may also be constrained by where we are allowed to place an entry …

TLB Searching

- How do we quickly look up a virtual page number?
  - Linear search?

TLB Searching

- How do we quickly look up a virtual page number?
  - Linear search?
    - Very simple!
    - Likely to be really slow!
  - Direct mapped
    - Let each page number map to a particular slot in the TLB.
    - Typically select the slot be extracting a few bits from the page number:



VA	PA	V
FD04	2340	1
0191	140E	1
0003	001D	1
1244	1403	1

Direct Mapping: Which bits?

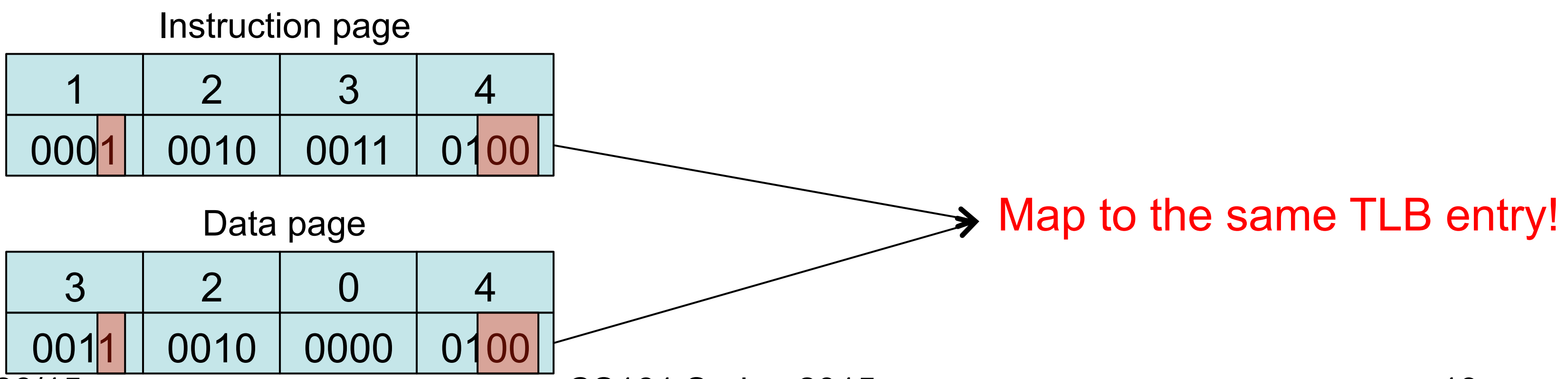
- High bits:
- Low bits:
- Middle bits:
- Mixture of high and low bits:

Direct Mapping: Thrashing (1)

- One of the problems with direct mapped caches is that they are subject to thrashing: that is repeated missing and eviction of the same pages.
- Consider a sequence of instructions whose instruction addresses and data addresses map to the same entry in the TLB.
- For example:

```
for (i = 0; i < 1024; i++)
     sum += array[i]
```

  - Let's assume that this loop starts at PC 12340000 and the array resides at 32040000.
  - Our page numbers are: 1234 and 3204



Direct Mapping: Thrashing (2)

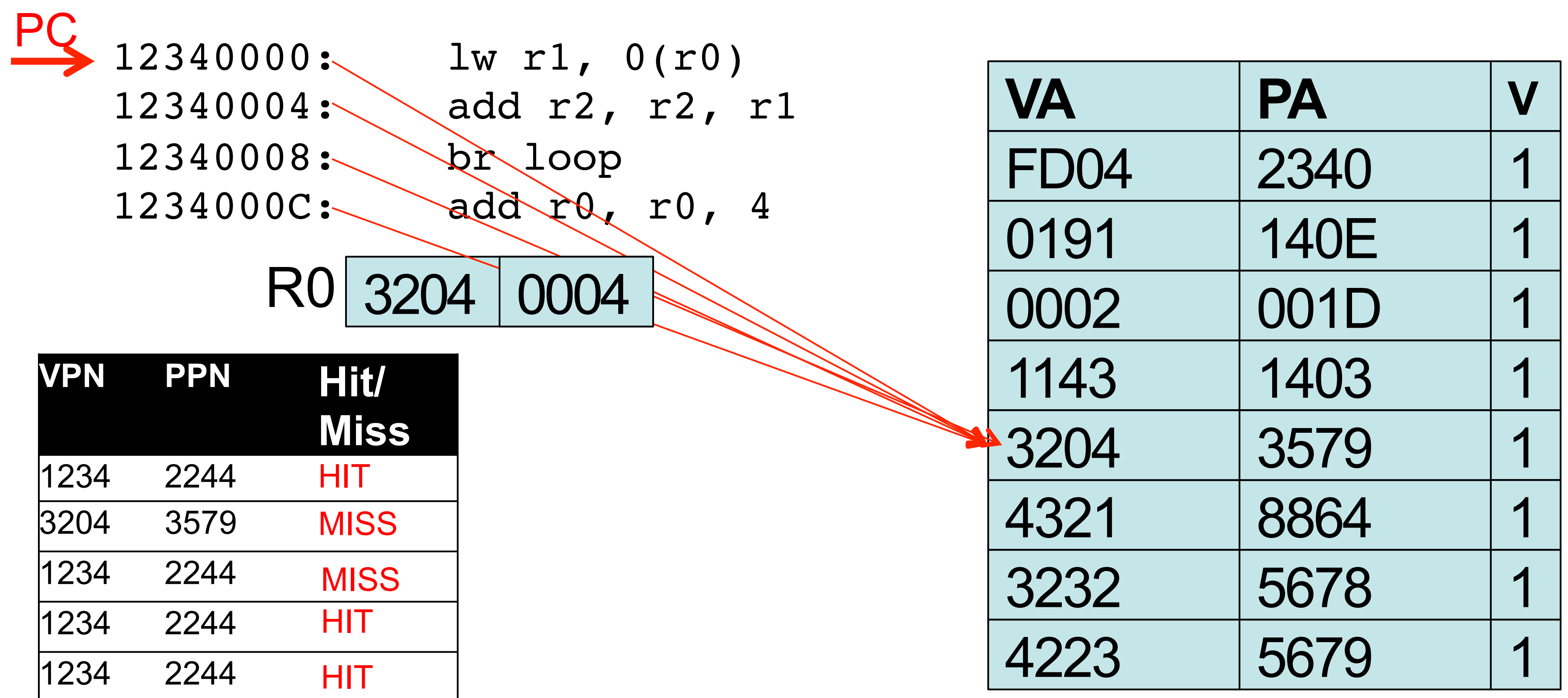
```
PC →
12340000:     lw r1, 0(r0)
12340004:     add r2, r2, r1
12340008:     br loop
1234000C:     add r0, r0, 4
```

R0 | 3204 | 0000

VPN	PPN	Hit/ Miss
1234	2244	HIT

VA	PA	V
FD04	2340	1
0191	140E	1
0002	001D	1
1143	1403	1
1234	2244	1
4321	8864	1
3232	5678	1
4223	5679	1

Direct Mapping: Thrashing (2)



```
12340000:     lw r1, 0(r0)
12340004:     add r2, r2, r1
12340008:     br loop
1234000C:     add r0, r0, 4
```

R0		
R0	3204	0004

VA	PA	V
FD04	2340	1
0191	140E	1
0002	001D	1
1143	1403	1
3204	3579	1
4321	8864	1
3232	5678	1
4223	5679	1

VPN	PPN	Hit/ Miss
1234	2244	HIT
3204	3579	MISS
1234	2244	MISS
1234	2244	HIT
1234	2244	HIT
1234	2244	HIT
3204	3579	MISS
1234	2244	MISS
1234	2244	HIT

TLB Hit Ratios

- TLB performance or effectiveness is expressed in terms of a *hit rate*: the percentage of times that an access hits in the TLB.

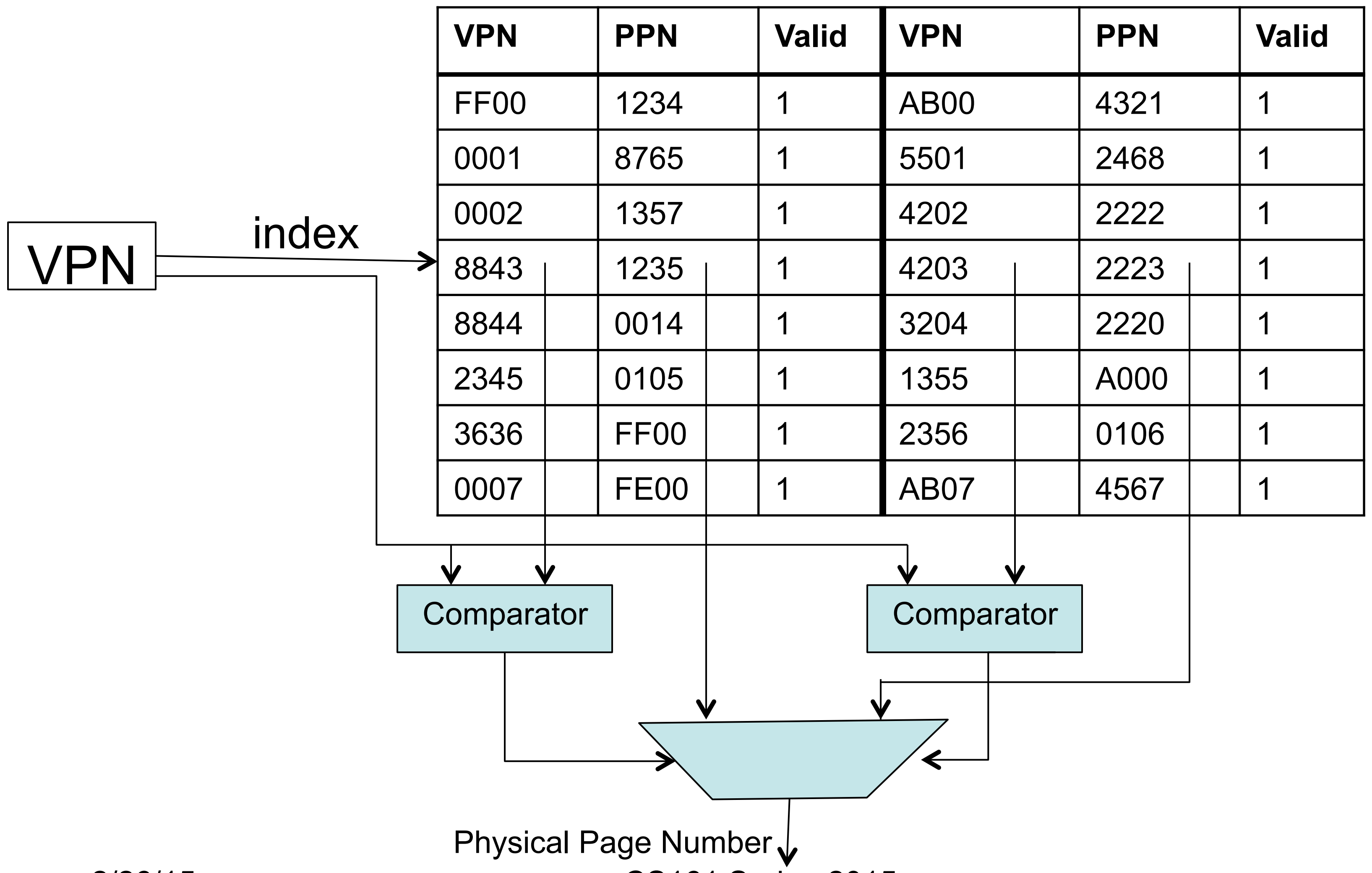
$$\text{TLB Hit Rate} = \frac{\text{\# TLB Hits}}{\text{\# TLB Accesses}}$$

- What is the hit rate on the previous slide?
- Even small TLBs typically achieve about a 98% hit rate; anything less is intolerable!
- How do they do it?

TLB: From Direct Mapped to Set Associative

- The fundamental problem we have is that any address can go in only one place.
- When more than one address that you need frequently maps to that one place, you are out of luck.
- Solution:
  - Provide more flexibility: let an address map to multiple locations
  - Search those locations in parallel.

2-Way Set Associative TLB



VPN	PPN	Valid	VPN	PPN	Valid
FF00	1234	1	AB00	4321	1
0001	8765	1	5501	2468	1
0002	1357	1	4202	2222	1
8843	1235	1	4203	2223	1
8844	0014	1	3204	2220	1
2345	0105	1	1355	A000	1
3636	FF00	1	2356	0106	1
0007	FE00	1	AB07	4567	1

Higher Associativity?

- If two-way is good, wouldn't four way be better?
- How about 8-way?
- How about fully associative?
  - Any entry can go in any location.
  - Older, tiny TLBs were, in fact, fully associative.
  - Implemented with a content-addressable memory (CAM).
  - CAMs are very expensive and become slower as you increase their size.
- In practice, most processors use a direct mapped or two-way set associative TLB.
- Once you have a few hundred entries, it doesn't seem to matter.

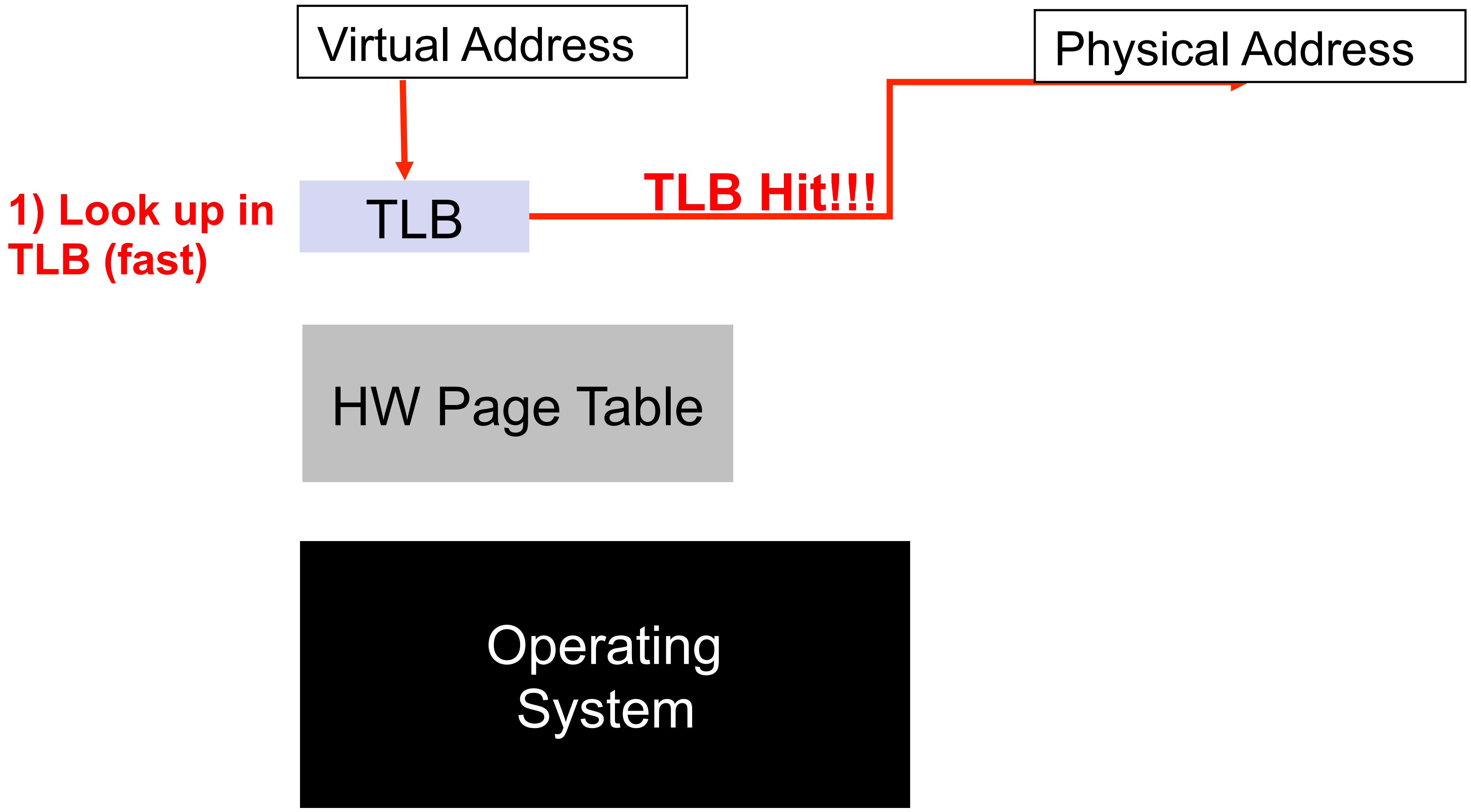
Back to TLB Eviction

- Recall (slide 13) that we started to look at how TLBs were arranged to answer the question, “How do we select an entry to evict when the TLB is full?”
- Answer:
  - You do not have many choices.
  - In a direct mapped TLB?
  - In a 2-way TLB?
  - What do you suppose you do?

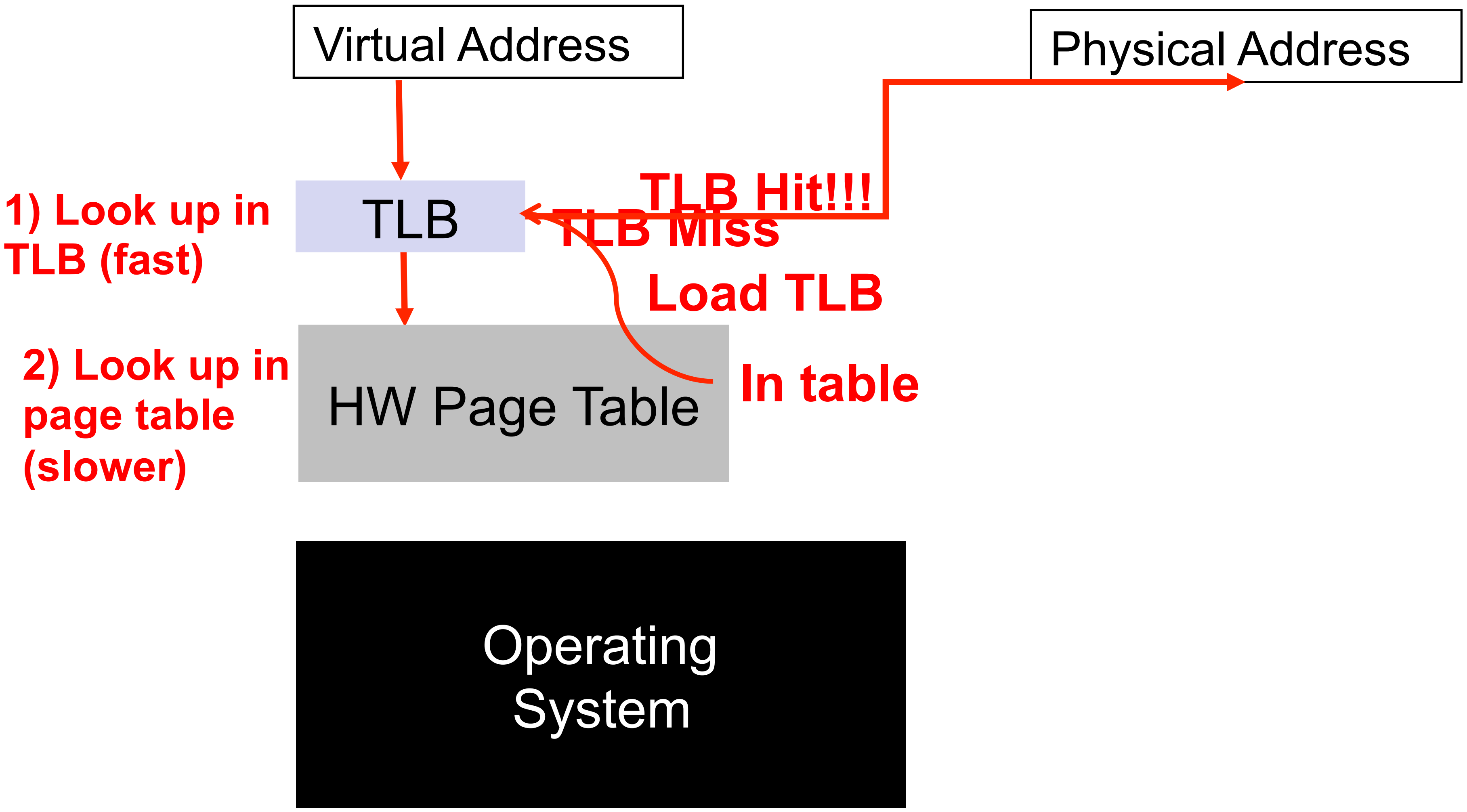
The MMU and Process Switching

- What happens to the state in the TLB on a process switch?
- How could you design a smarter TLB?
- We’ve seen that traps must do a lot of work in saving and restoring state. This makes process switching expensive. What are some of the other (hidden) costs of process switching?

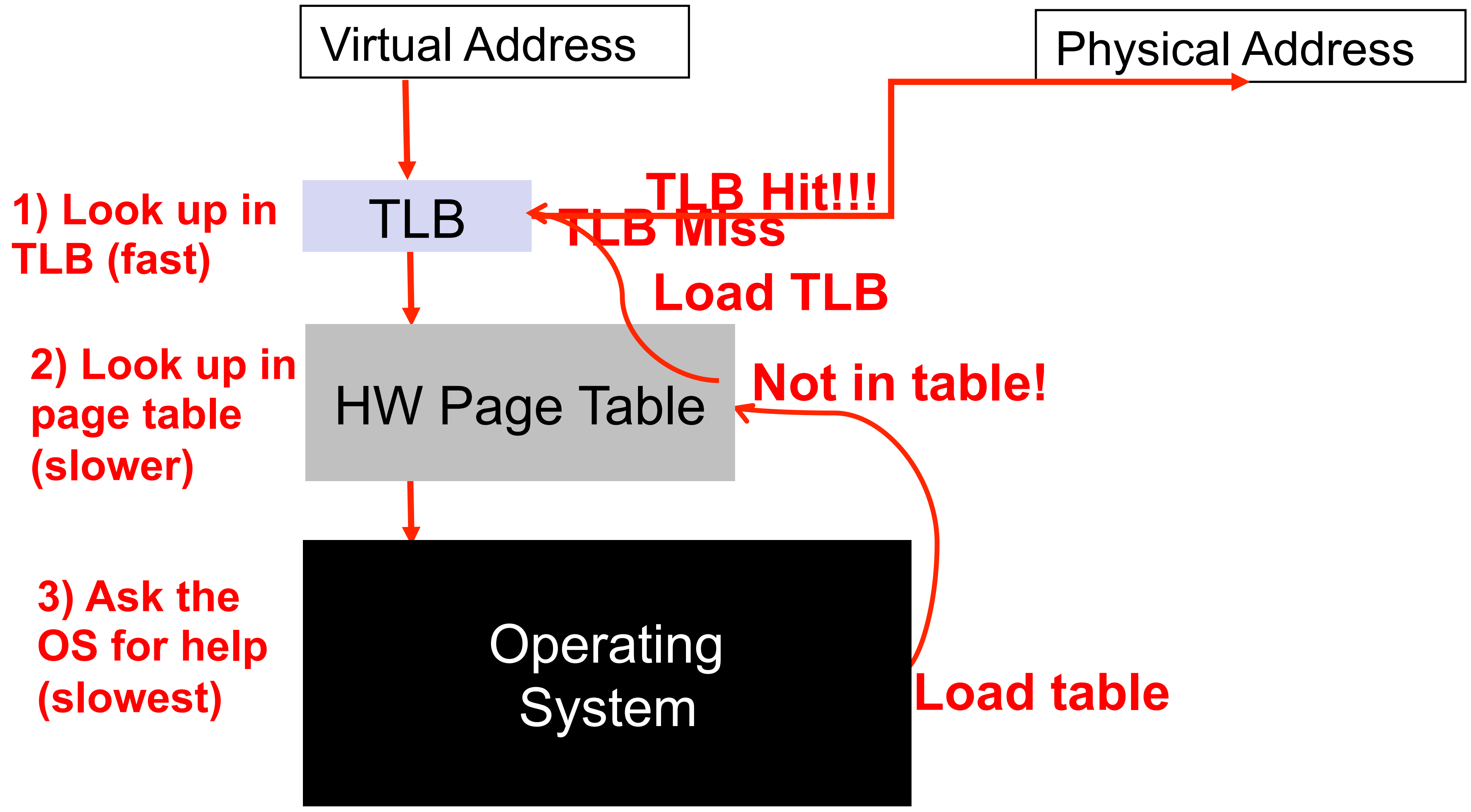
Address Translation Summary (1)



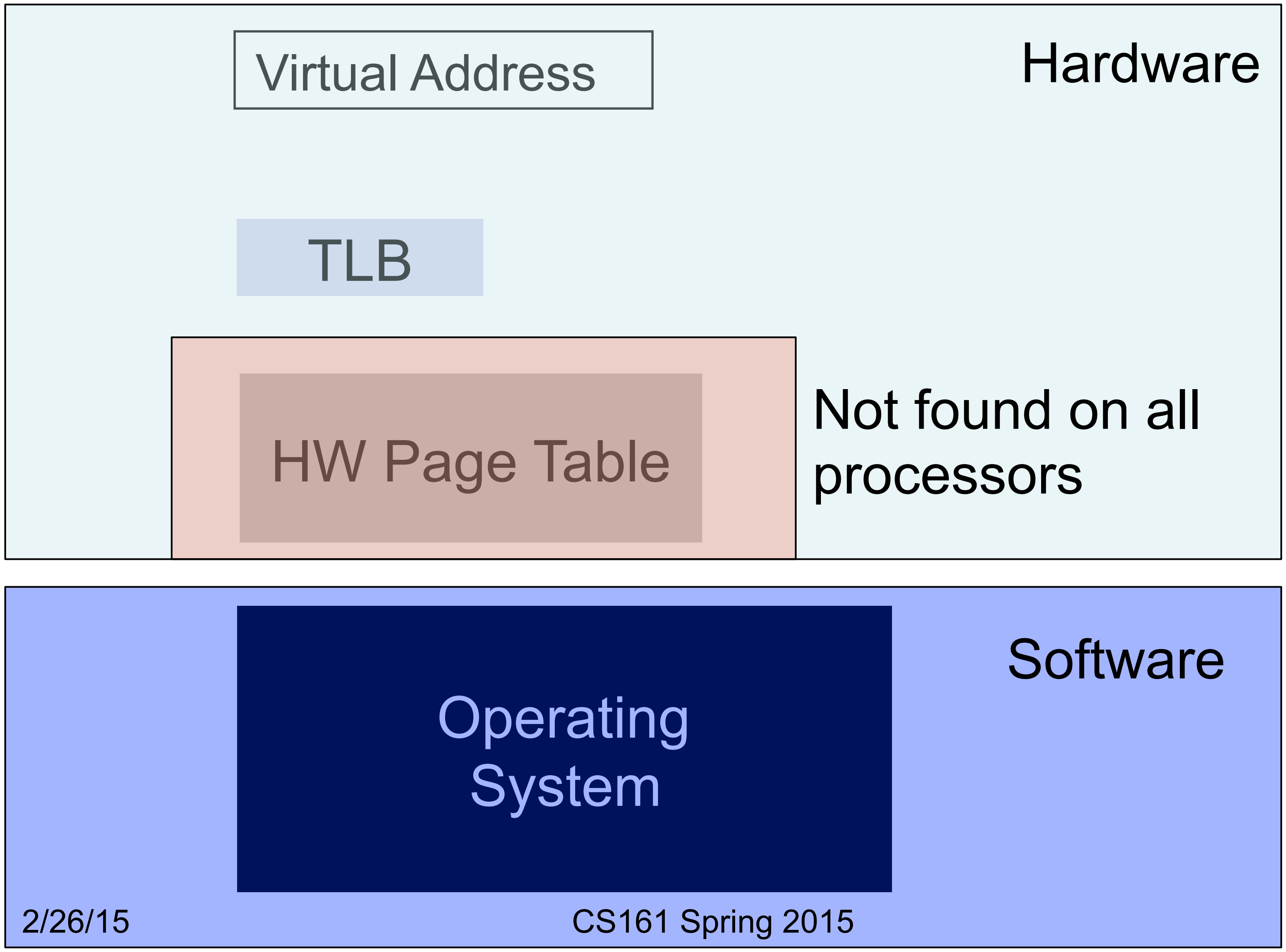
Address Translation Summary (2)



Address Translation Summary (3)



Hardware/Software Boundary



HW versus SW TLB Fault Handling

- Software (MIPS R2000):
  - Advantages
    - Simplicity (of hardware)
    - Flexibility
    - More able to monitor page accesses
    - Good for homework assignments in operating systems
  - Disadvantages:
    - Slow?
- Hardware (x86):
  - Advantages:
    - Speed
  - Disadvantages:
    - Less control
    - Hardware dictates page table structure