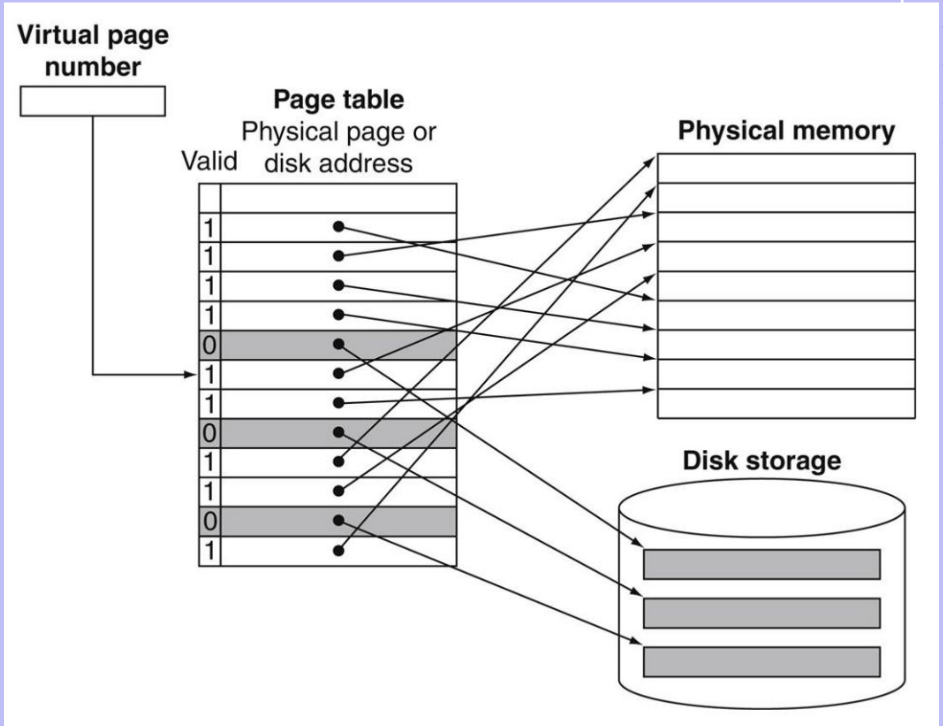
TLBs

? ? ?

How long does a memory access take on a system with virtual memory vs. one without?

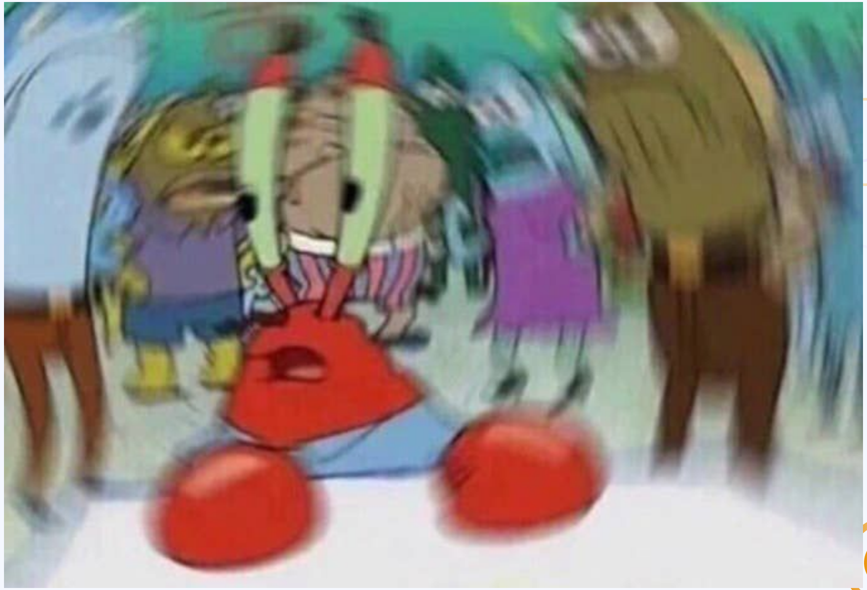


(Translation Lookaside Buffer)

TLBs: a cache for the page table

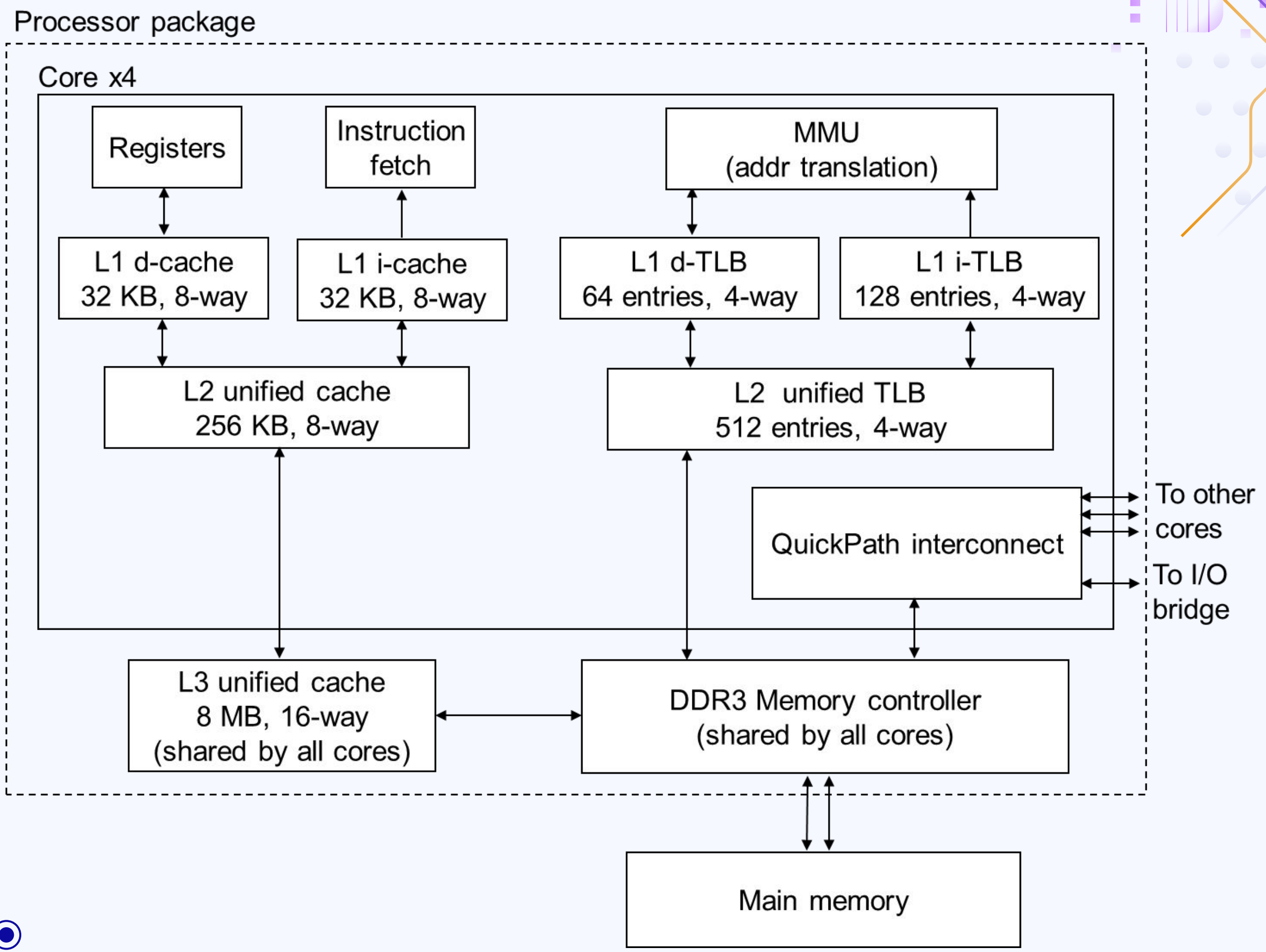
For those counting: we have

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

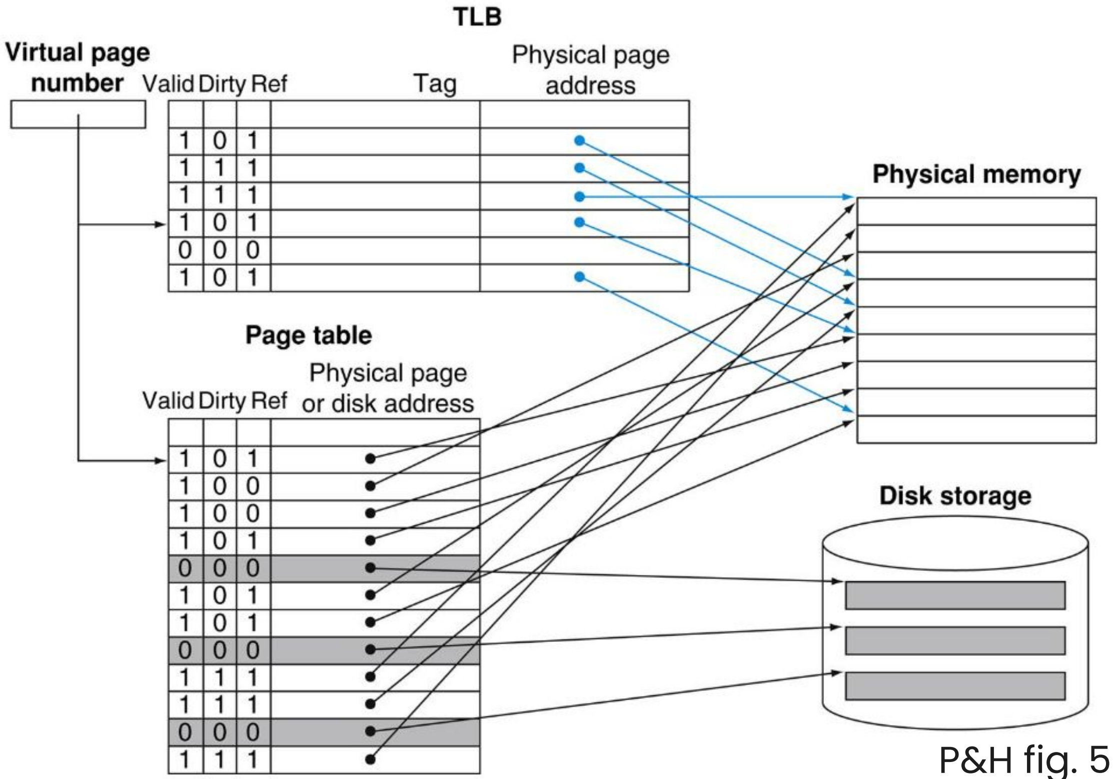


Intel i7

Source
(Bryant & O’Hallaron)


Processor package
Core x4
Registers
Instruction fetch
MMU (addr translation)
L1 d-cache 32 KB, 8-way
L1 i-cache 32 KB, 8-way
L1 d-TLB 64 entries, 4-way
L1 i-TLB 128 entries, 4-way
L2 unified cache 256 KB, 8-way
L2 unified TLB 512 entries, 4-way
QuickPath interconnect
To other cores
To I/O bridge
L3 unified cache 8 MB, 16-way (shared by all cores)
DDR3 Memory controller (shared by all cores)
Main memory

TLBs: does this clear it up?



TLB

Valid	Dirty	Ref	Tag	Physical page address
1	0	1		●
1	1	1		●
1	1	1		●
1	0	1		●
0	0	0		
1	0	1		●

Page table

Valid	Dirty	Ref	Physical page or disk address
1	0	1	●
1	0	0	●
1	0	0	●
1	0	1	●
0	0	0	●
1	0	1	●
1	0	1	●
0	0	0	●
1	1	1	●
1	1	1	●
0	0	0	●
1	1	1	●

P&H fig. 5.30

? ? ?

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

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

Cache addressing

PIPT (physical address) caches come at page translation cost (*maybe OK now that we have TLBs??*)

VIVT (virtual address) caches cause aliasing issues

Homonyms: two different processes using the same virtual address

(solution: match address signifier – ASID – along with tag)

Synonyms: two virtual addresses mapping to same physical address

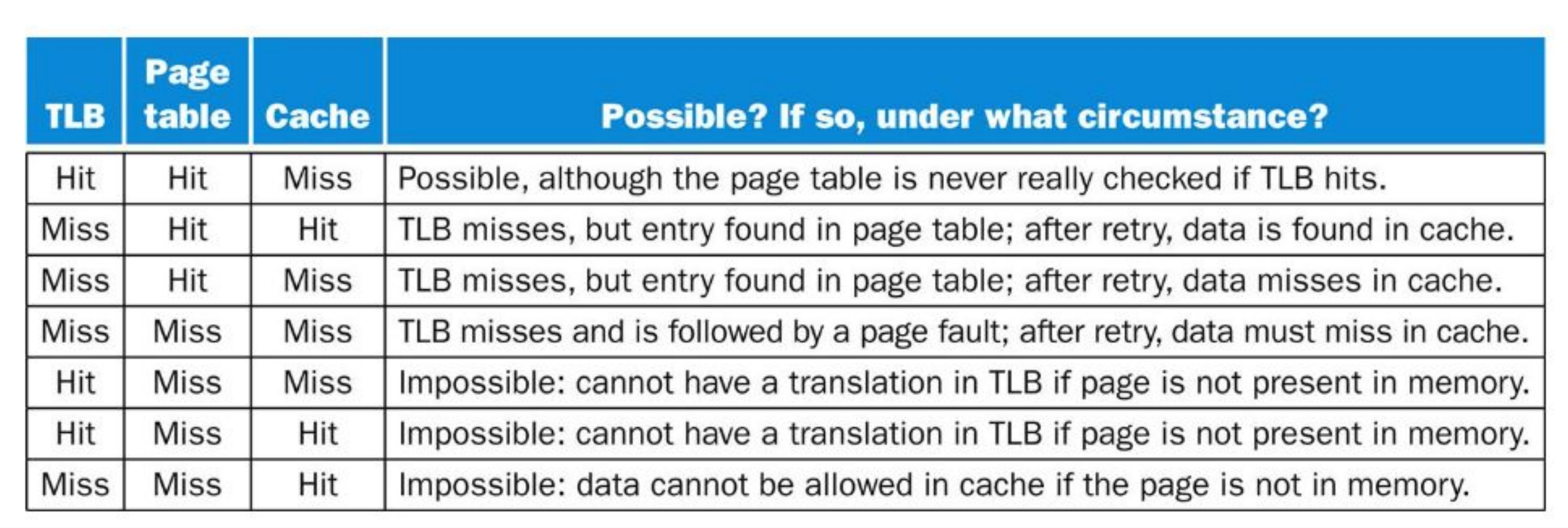
(solution: hardware detection)

Not used very much these days

What's with the weird acronyms??

Interaction of TLB and cache (PIPT)

P&H fig. 5.33



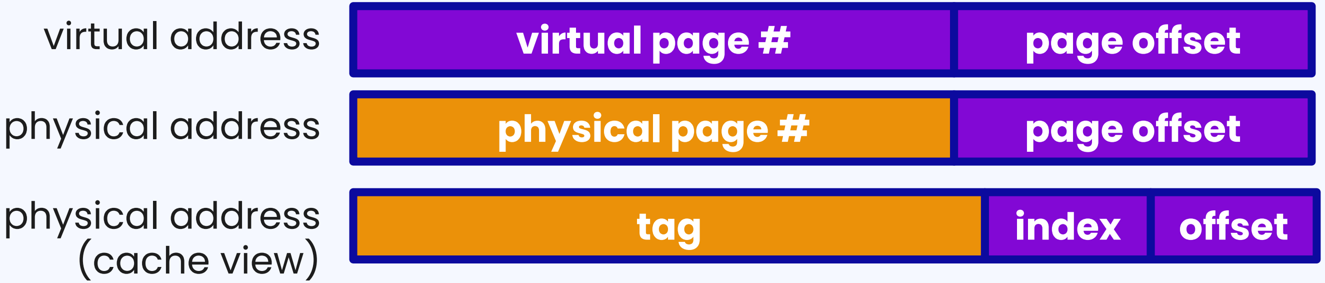
TLB	Page table	Cache	Possible? If so, under what circumstance?
Hit	Hit	Miss	Possible, although the page table is never really checked if TLB hits.
Miss	Hit	Hit	TLB misses, but entry found in page table; after retry, data is found in cache.
Miss	Hit	Miss	TLB misses, but entry found in page table; after 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

Can also use fancy OS math (page coloring) if index bits are mixed virtual/physical

Virtually *indexed*, Physically *tagged*

Choose $ topology so index bits of address are within page offset (fully available from virtual address)

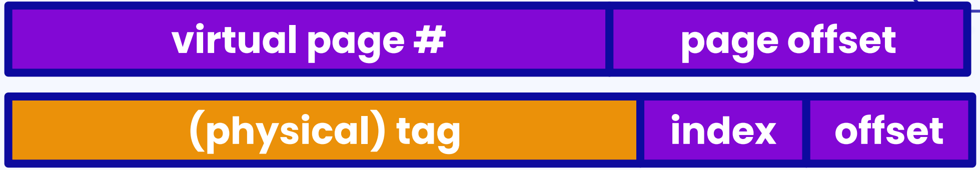


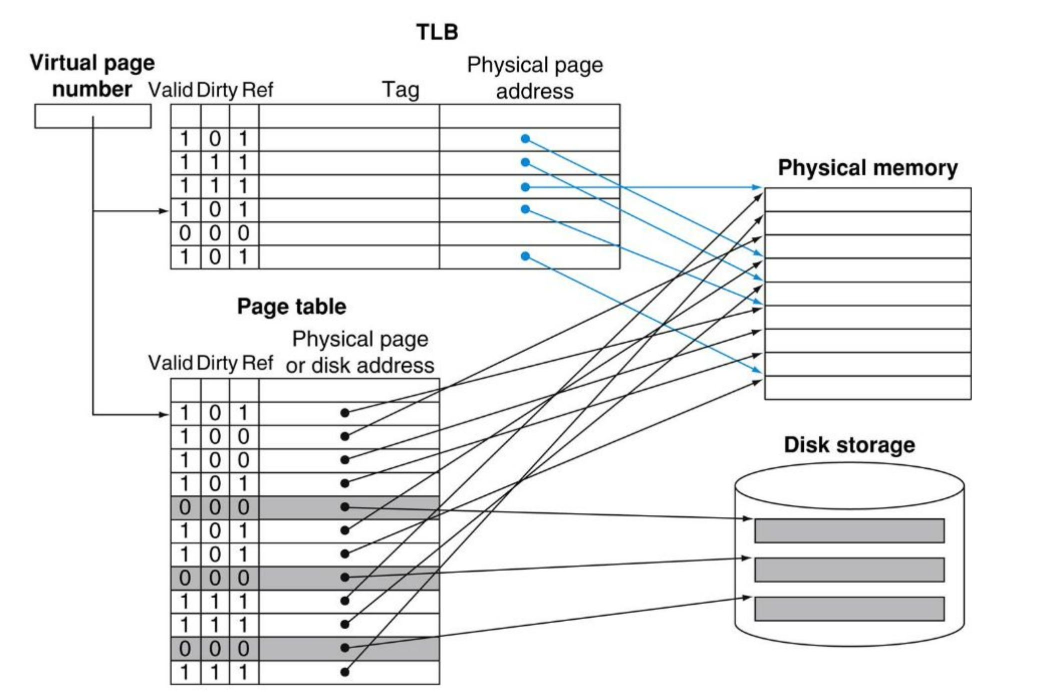
virtual address: virtual page # | page offset

physical address: physical page # | page offset

physical address (cache view): tag | index | offset

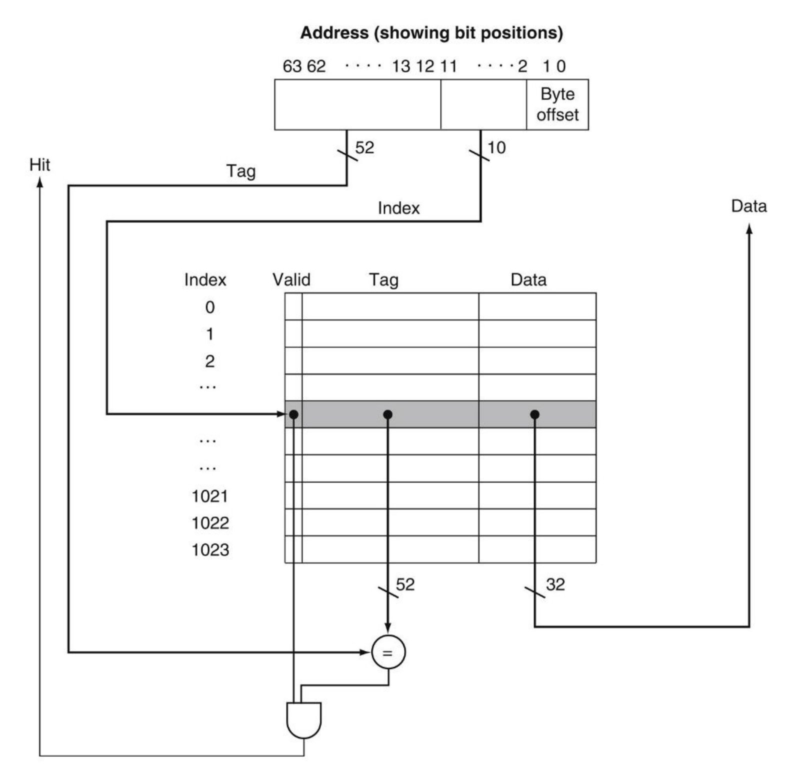
Don’t need to know physical address to find cache index, just to do tag comparison and can do **parallel lookup** of tag (in TLB) and index (in $)!!





Valid	Dirty	Ref	Tag	Physical page address
1	0	1		
1	1	1		
1	1	1		
1	0	1		
0	0	0		
1	0	1		

Valid	Dirty	Ref	Physical page or disk address
1	0	1	
1	0	0	
1	0	0	
1	0	1	
0	0	0	
1	0	1	
1	0	1	
0	0	0	
1	1	1	
1	1	1	
0	0	0	
1	1	1	


Address (showing bit positions)
63 62 · · · · 13 12 11 · · · · 2 1 0
Byte offset
52
10
Hit
Tag
Index
Data
Index
Valid
Tag
Data
0
1
2
…
…
…
1021
1022
1023
52
32
=