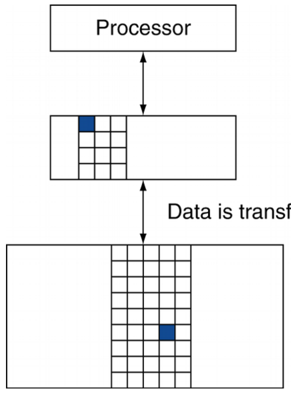
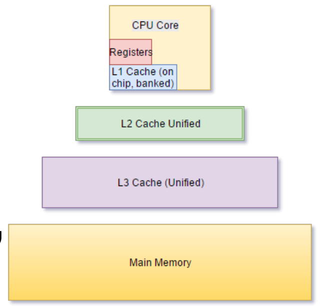
Where are we?

Memory Hierarchy



Improving Cache Performance

- Reduce miss rate (associative cache)
- Reduce miss (time) penalty (Hierarchy)
- Speed hit access time! (parallel search, Hierarchy)



1. Where can a block be placed? **(Q1)**
2. How is a block found? **(Q2)**
3. What block is replaced on a miss? **(Q3)**
4. How are writes handled? **(Q4)**

Virtual Memory

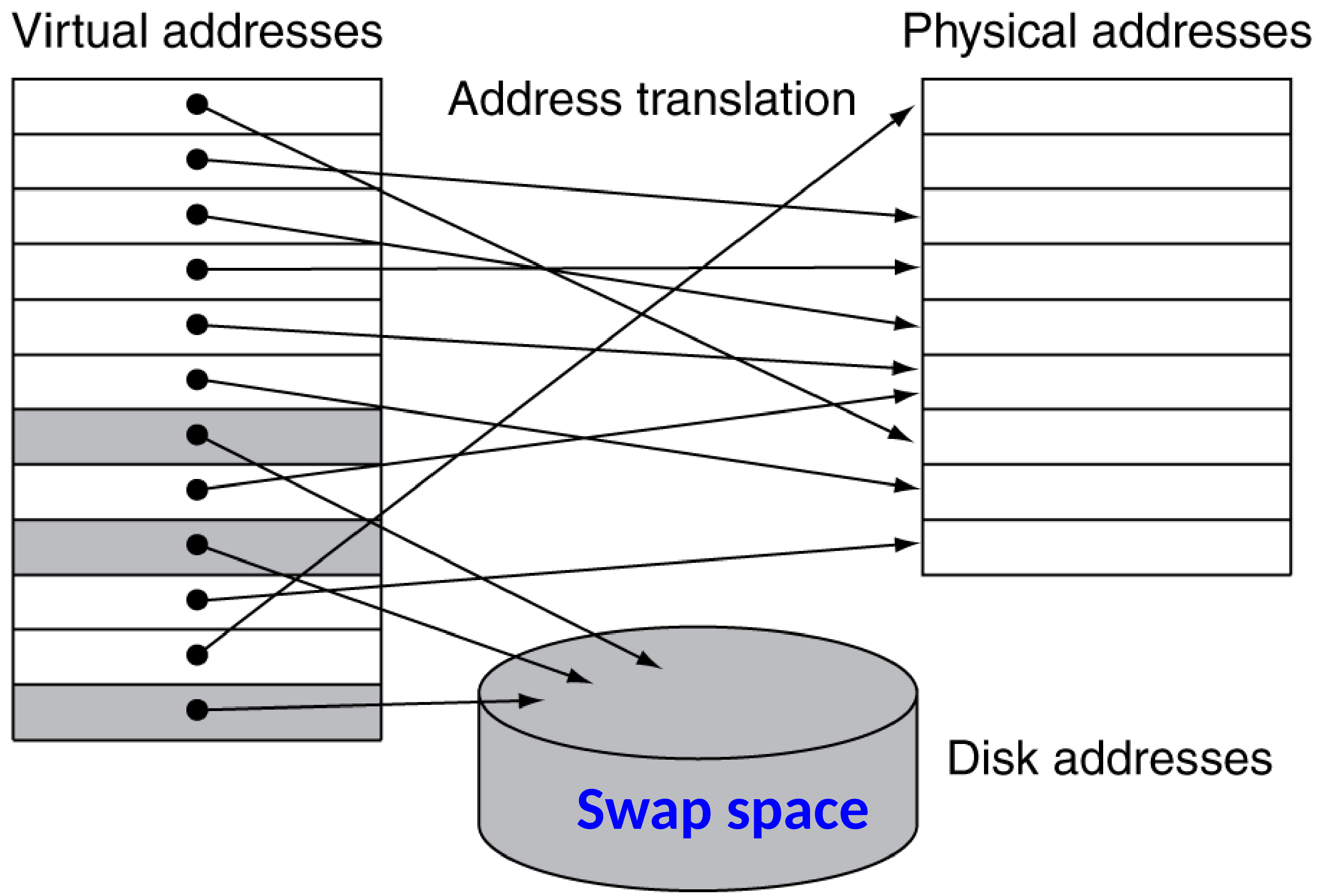
The large memory illusion

Virtual Memory

- Use main memory as a “cache” for secondary (disk) storage
  - Managed jointly by CPU hardware and the operating system (OS)
- Programs share main memory
  - Each gets a private virtual address space holding its frequently used code and data
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
  - Virtual memory “block” is called a page
  - Virtual memory translation “miss” is called a page fault

Address Translation

- Fixed-size pages (e.g., 4K)



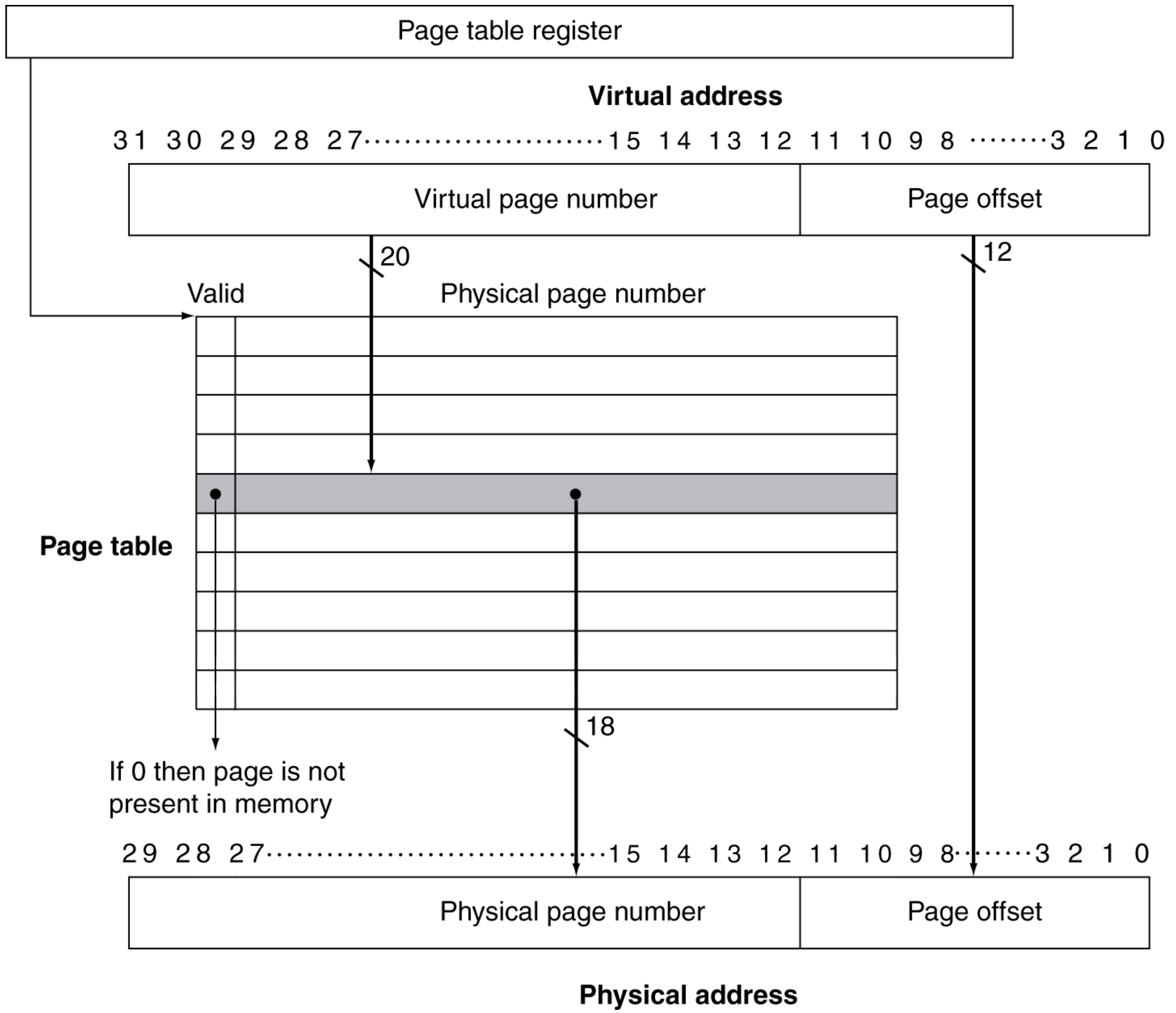
Page Fault Penalty

- On page fault $\rightarrow$ the page must be fetched from disk
    - Takes ***millions*** of clock cycles
- Page fault rate should be minimized
    - Fully associative placement
    - Smart replacement algorithms used by operating system to track placement

Page Tables

- Stores placement information
  - Array of page table entries, indexed by virtual page number
  - Each program has its own page table
  - hardware includes a register that points to the start of the page table [***page table register***]
- If page is present in memory
  - Page table entry stores the physical page number
  - Plus other status bits (referenced, dirty, …)
- If page is not present
  - Page table entry can refer to location in swap space on disk

Translation Using a Page Table



Replacement and Writes

- To reduce page fault rate, prefer least-recently used (LRU) replacement
  - Reference bit (aka use bit) in Page Table set to 1 on access to page
  - Periodically cleared to 0 by OS
  - A page with reference bit = 0 has not been used recently
  - Fully associative placement to reduce the miss penalty
- Disk writes take millions of cycles
  - Block at once, not individual locations
  - Write through is impractical
  - Use write-back when replacing a page
  - Dirty bit in Page Table set when page is written
    - A modified page is often called a dirty page

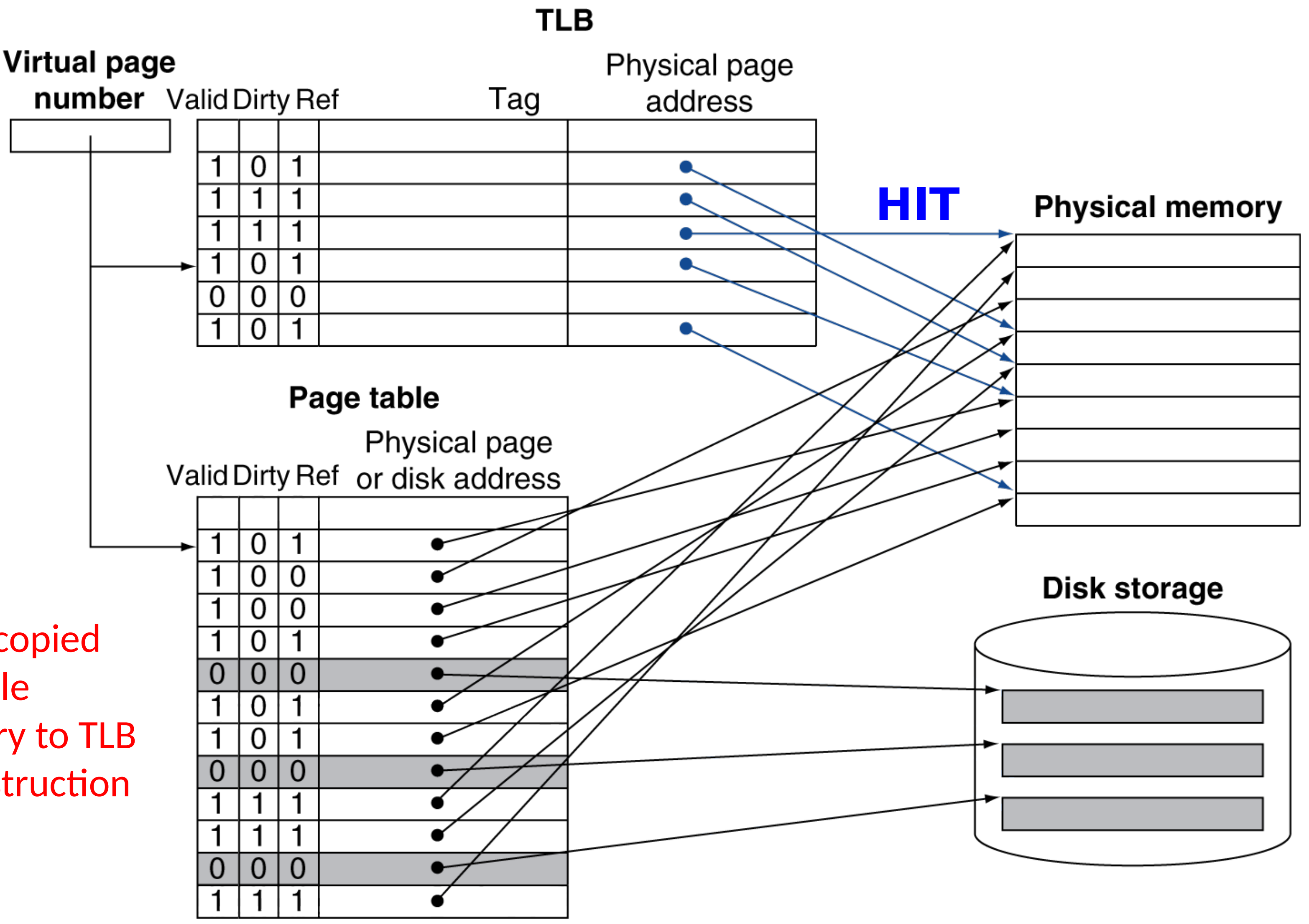
Speeding Address Translation

- How many times is memory referenced to access a variable?
  - One to access the Page Table entry
  - Then the actual memory access

- But access to page tables has good locality
  - use ***Translation Look-aside Buffer (TLB)***
    - a fast **cache of Page Table Entries (PTE)** within the CPU
    - Typical: 16–512 PTEs, 0.5–1 cycle for hit, 10–100 cycles for miss, 0.01%–1% miss rate

Fast Translation Using a TLB

TLB misses will be much more frequent than true page faults



TLB MISS

1. Replaced PTE is copied back to page table
2. PTE from memory to TLB
3. Then restarts instruction

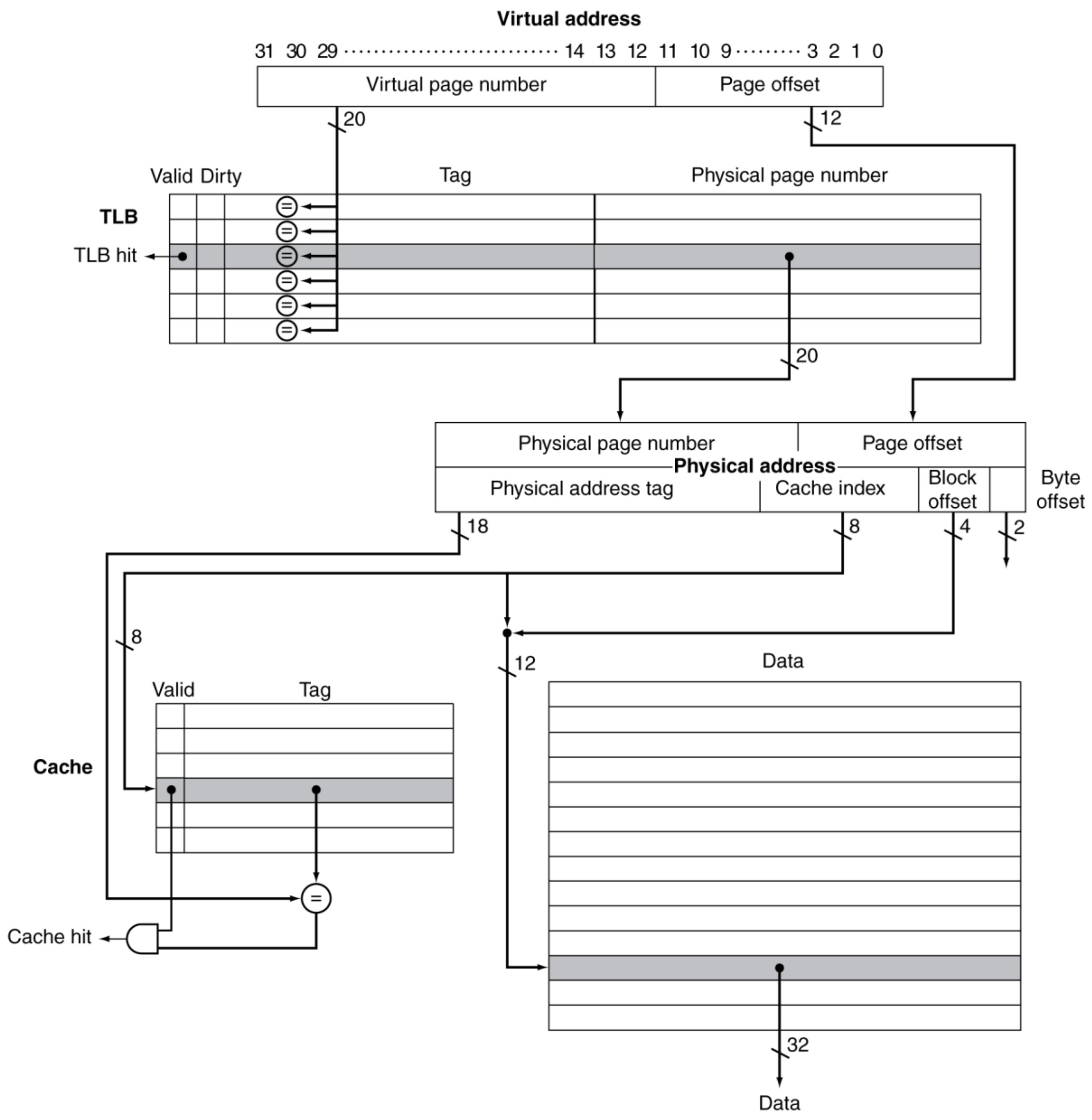
True page fault-→ the processor invokes the operating system using an exception

Page Fault Handler

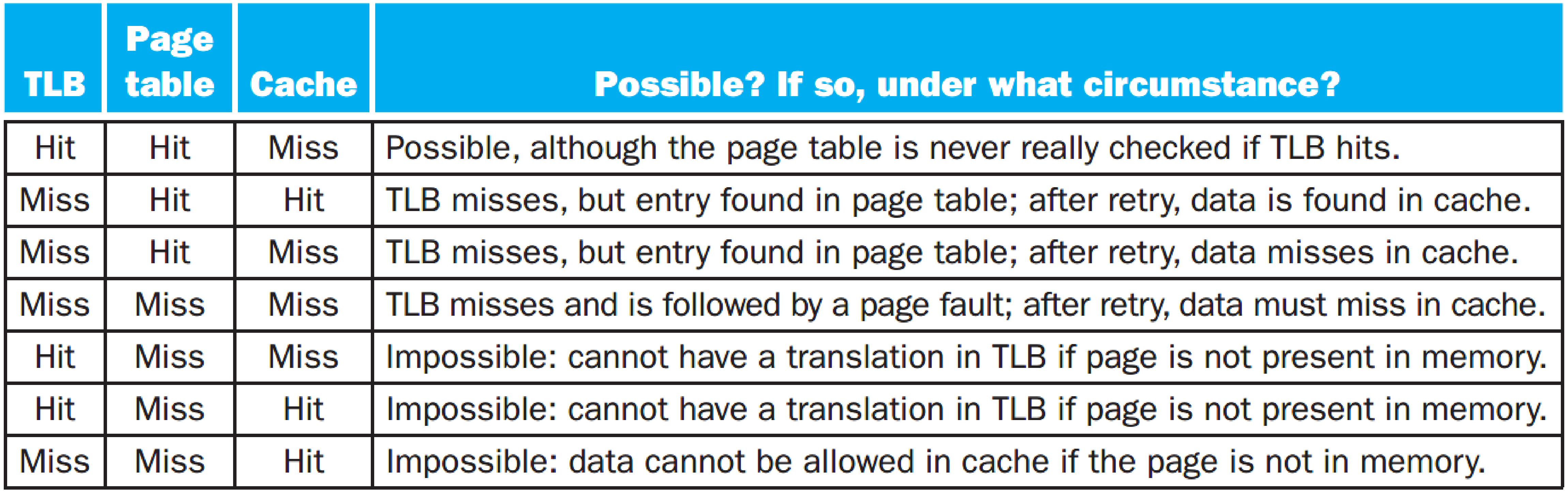
- Handled by the OS

1. Use faulting virtual address to find PTE
2. Locate page on disk
3. Choose page to replace
   - If dirty, write to disk first
4. Read page into memory and update page table
5. Make process runnable again
   - Restart from faulting instruction

TLB and Cache Interaction



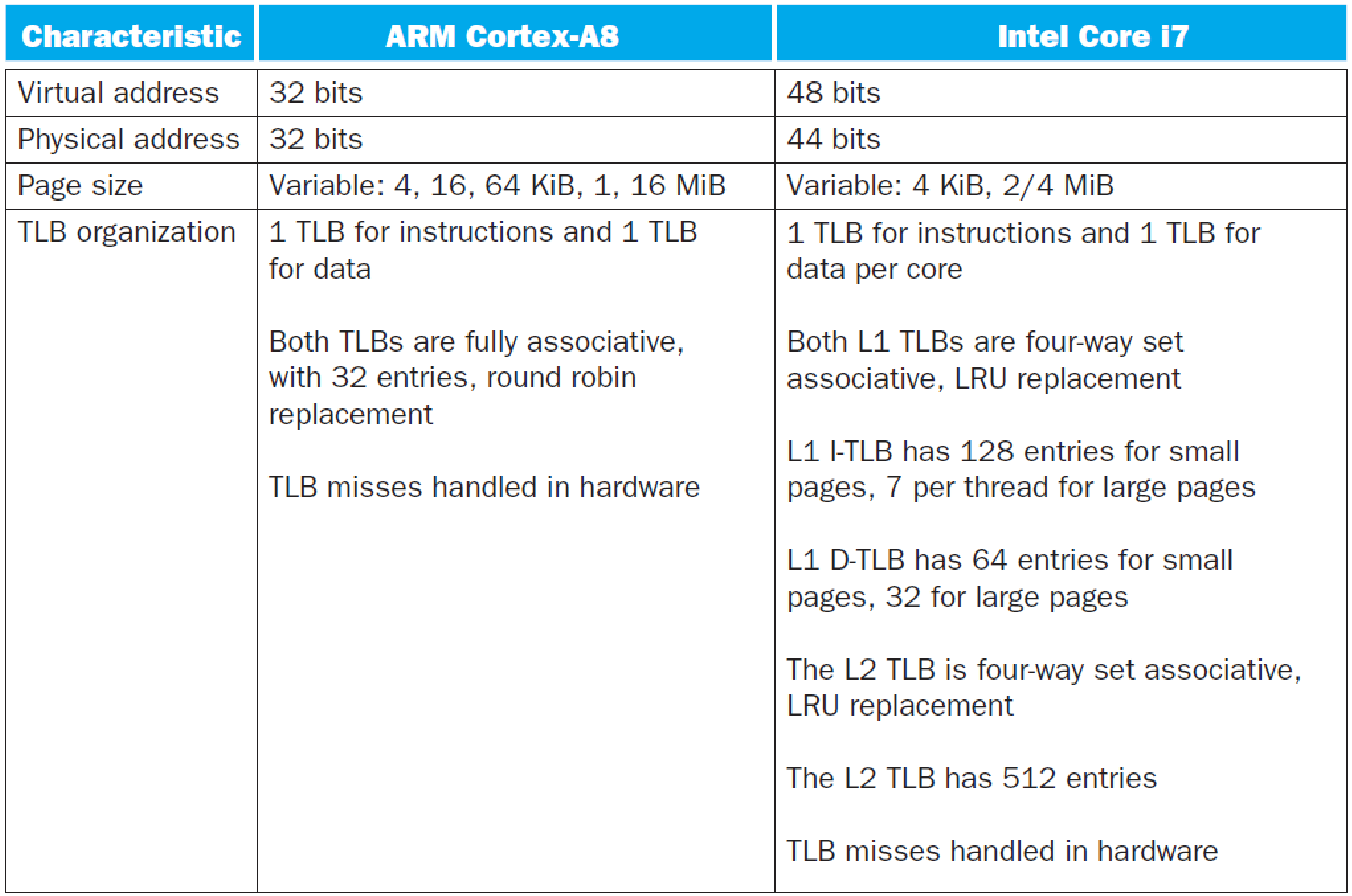
Discussion



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

Other Architectures

TLB Organization



Characteristic	ARM Cortex-A8	Intel Core i7
Virtual address	32 bits	48 bits
Physical address	32 bits	44 bits
Page size	Variable: 4, 16, 64 KiB, 1, 16 MiB	Variable: 4 KiB, 2/4 MiB
TLB organization	1 TLB for instructions and 1 TLB for data Both TLBs are fully associative, with 32 entries, round robin replacement TLB misses handled in hardware	1 TLB for instructions and 1 TLB for data per core Both L1 TLBs are four-way set associative, LRU replacement L1 I-TLB has 128 entries for small pages, 7 per thread for large pages L1 D-TLB has 64 entries for small pages, 32 for large pages The L2 TLB is four-way set associative, LRU replacement The L2 TLB has 512 entries TLB misses handled in hardware

Multilevel On-Chip Caches

Characteristic	ARM Cortex-A8	Intel Nehalem
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KiB each for instructions/data	32 KiB each for instructions/data per core
L1 cache associativity	4-way (I), 4-way (D) set associative	4-way (I), 8-way (D) set associative
L1 replacement	Random	Approximated LRU
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate(?)	Write-back, No-write-allocate
L1 hit time (load-use)	1 clock cycle	4 clock cycles, pipelined
L2 cache organization	Unified (instruction and data)	Unified (instruction and data) per core
L2 cache size	128 KiB to 1 MiB	256 KiB (0.25 MiB)
L2 cache associativity	8-way set associative	8-way set associative
L2 replacement	Random(?)	Approximated LRU
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate (?)	Write-back, Write-allocate
L2 hit time	11 clock cycles	10 clock cycles
L3 cache organization	–	Unified (instruction and data)
L3 cache size	–	8 MiB, shared
L3 cache associativity	–	16-way set associative
L3 replacement	–	Approximated LRU
L3 block size	–	64 bytes
L3 write policy	–	Write-back, Write-allocate
L3 hit time	–	35 clock cycles

Reading

- Sections 5.7, 5.9, 5.10