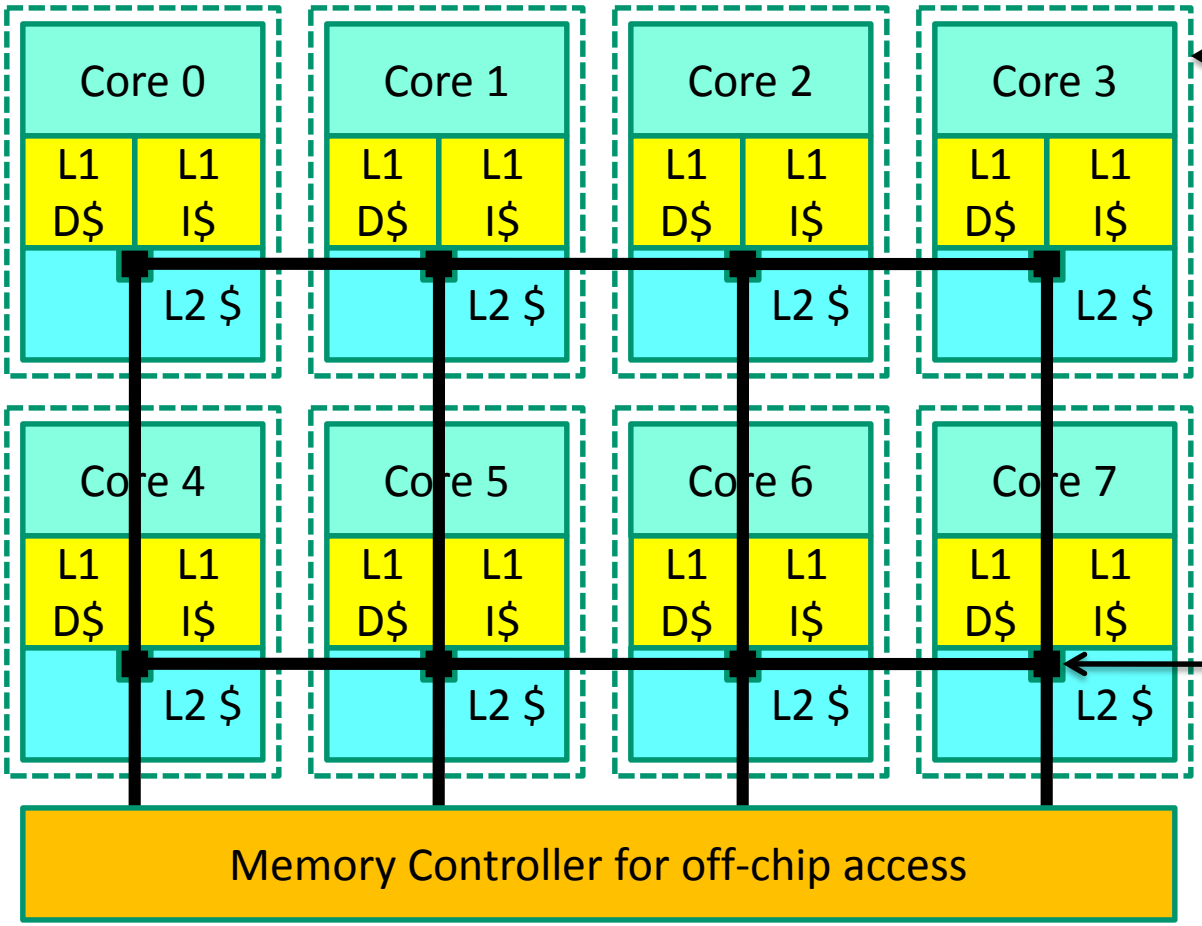
Lecture: Virtual Memory

- Topics: virtual memory, TLB/cache access (Sections 2.2)

Shared NUCA Cache



A single tile composed of a core, L1 caches, and a bank (slice) of the shared L2 cache

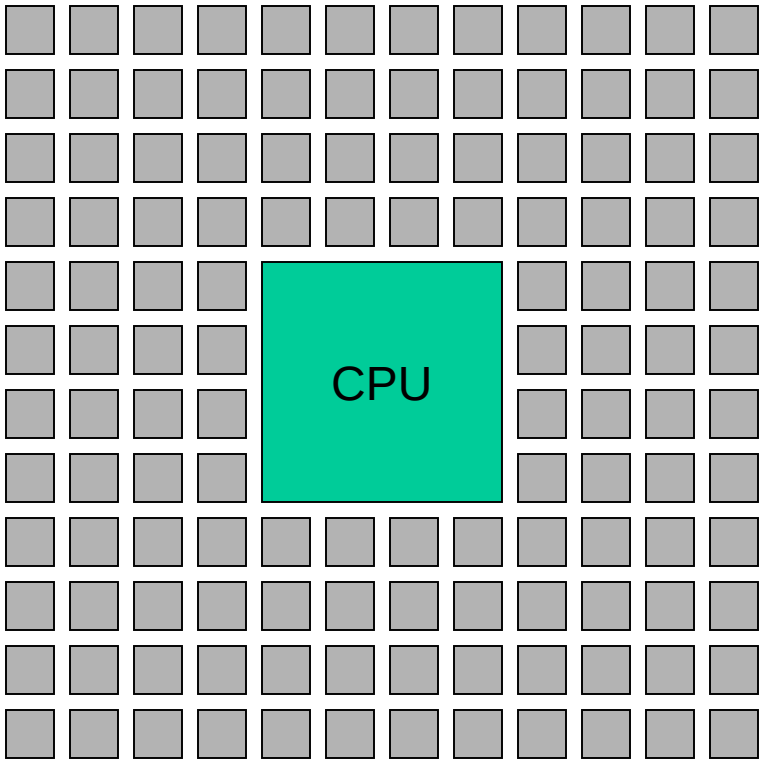
The cache controller forwards address requests to the appropriate L2 bank and handles coherence operations

UCA and NUCA

- The small-sized caches so far have all been uniform cache access: the latency for any access is a constant, no matter where data is found

- For a large multi-megabyte cache, it is expensive to limit access time by the worst case delay: hence, non-uniform cache architecture

Large NUCA



Issues to be addressed for Non-Uniform Cache Access:

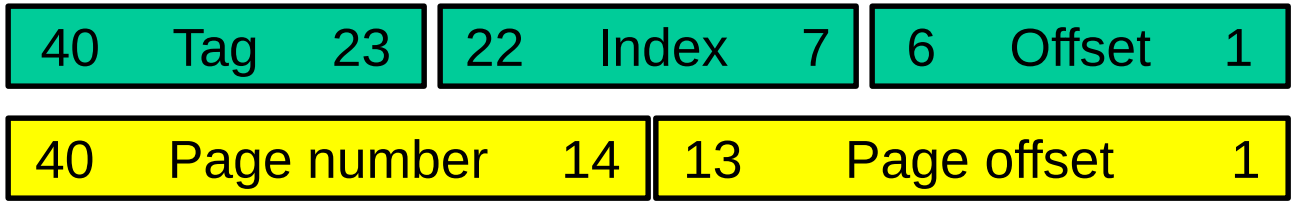
- Mapping
- Migration
- Search
- Replication

Problem 1

- Assume a large shared LLC that is tiled and distributed on the chip. Assume 16 tiles. Assume an OS page size of 8KB. The entire LLC has a size of 32 MB, uses 64-byte blocks, and is 8-way set-associative. Which of the 40 physical address bits are used to specify the tile number? Provide an example page number that is assigned to tile 0.

Problem 1

- Assume a large shared LLC that is tiled and distributed on the chip. Assume 16 tiles. Assume an OS page size of 8KB. The entire LLC has a size of 32 MB, uses 64-byte blocks, and is 8-way set-associative. Which of the 40 physical address bits are used to specify the tile number? Provide an example page number that is assigned to tile 0.



The cache has 64K sets, i.e., 6 block offset bits, 16 index bits, and 18 tag bits. The address also has a 13-bit page offset, and 27 page number bits. Nine bits (bits 14-22) are used for the page number and the index bits. Any four of those bits can be used to designate the tile number, say, bits 19-22. An example page number assigned to tile 0 is xxx…xxx0000xxx…xxx

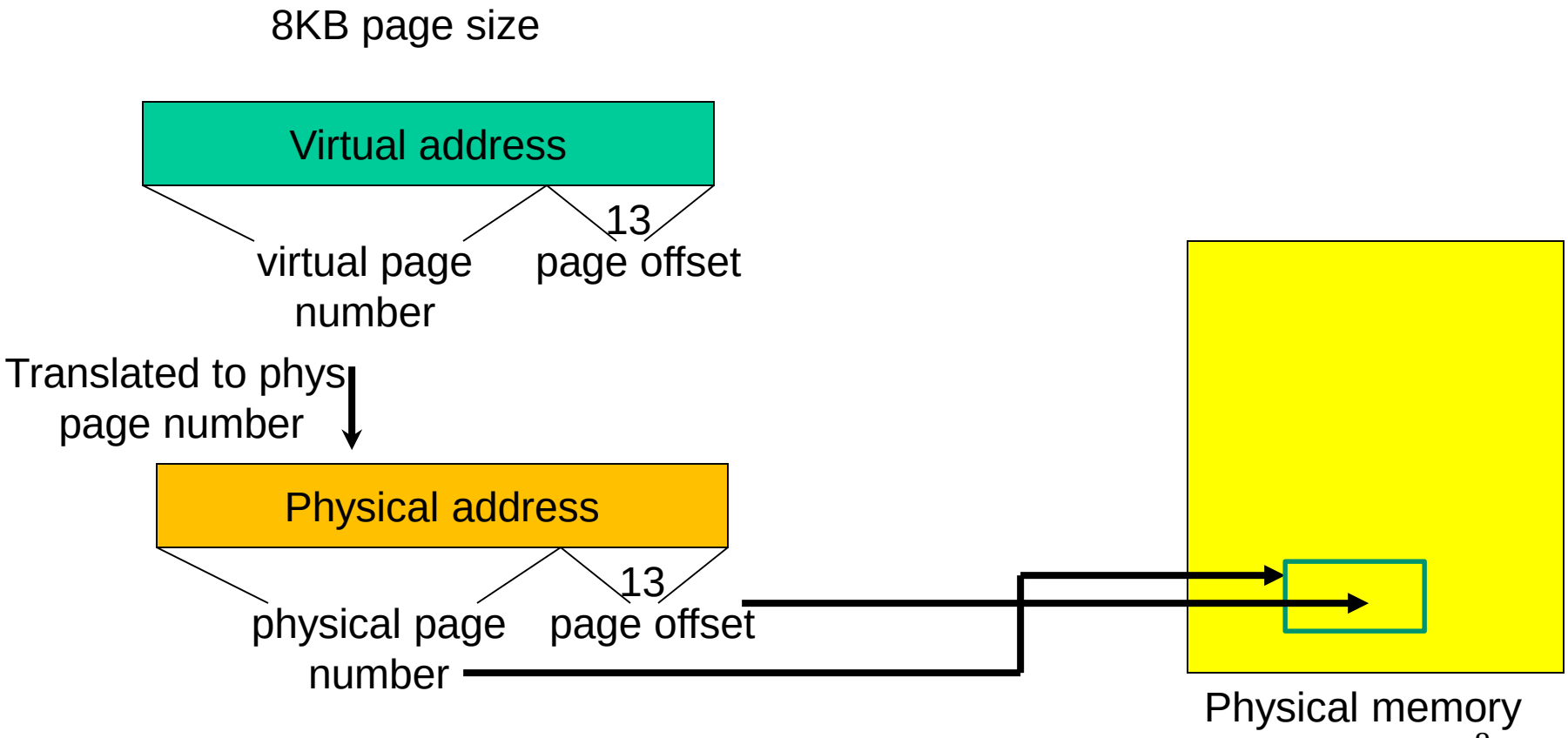
bit 22 19

Virtual Memory

- Processes deal with virtual memory – they have the illusion that a very large address space is available to them
- There is only a limited amount of physical memory that is shared by all processes – a process places part of its virtual memory in this physical memory and the rest is stored on disk
- Thanks to locality, disk access is likely to be uncommon
- The hardware ensures that one process cannot access the memory of a different process

Address Translation

- The virtual and physical memory are broken up into pages



Memory Hierarchy Properties

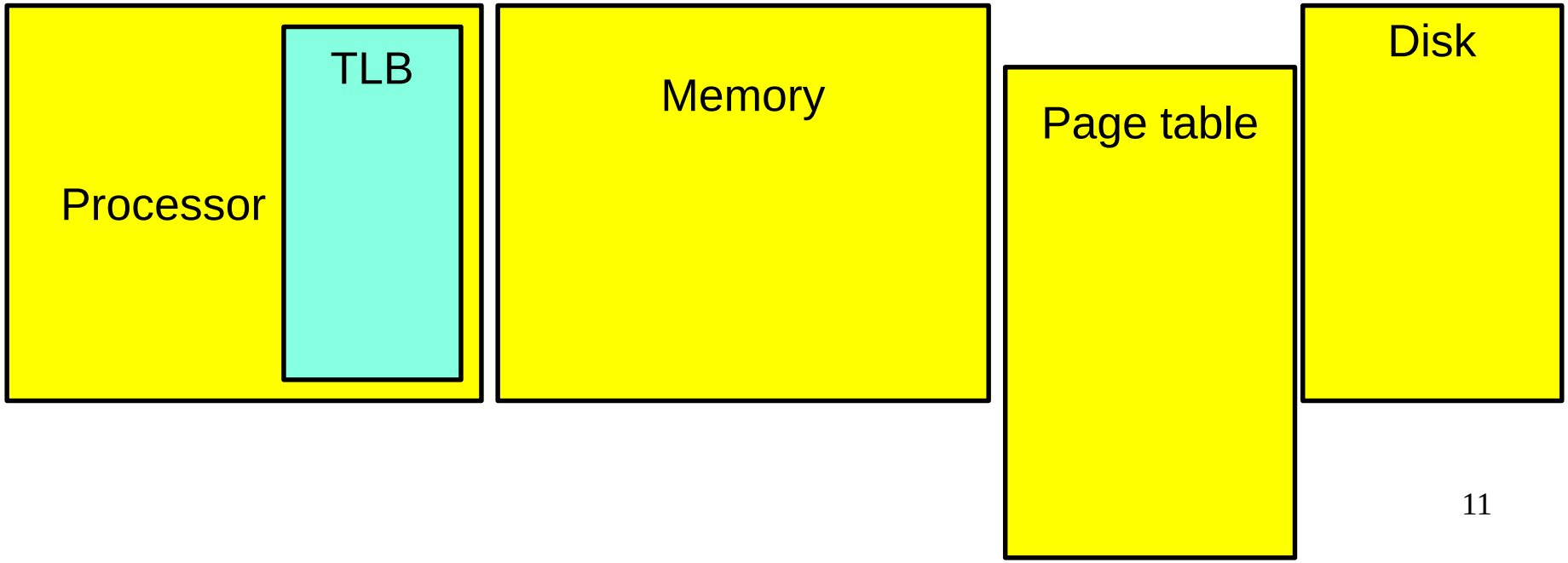
- A virtual memory page can be placed anywhere in physical memory (fully-associative)
- Replacement is usually LRU (since the miss penalty is huge, we can invest some effort to minimize misses)
- A page table (indexed by virtual page number) is used for translating virtual to physical page number
- The memory-disk hierarchy can be either inclusive or exclusive and the write policy is writeback

TLB

- Since the number of pages is very high, the page table capacity is too large to fit on chip
- A translation lookaside buffer (TLB) caches the virtual to physical page number translation for recent accesses
- A TLB miss requires us to access the page table, which may not even be found in the cache – two expensive memory look-ups to access one word of data!
- A large page size can increase the coverage of the TLB and reduce the capacity of the page table, but also increases memory waste

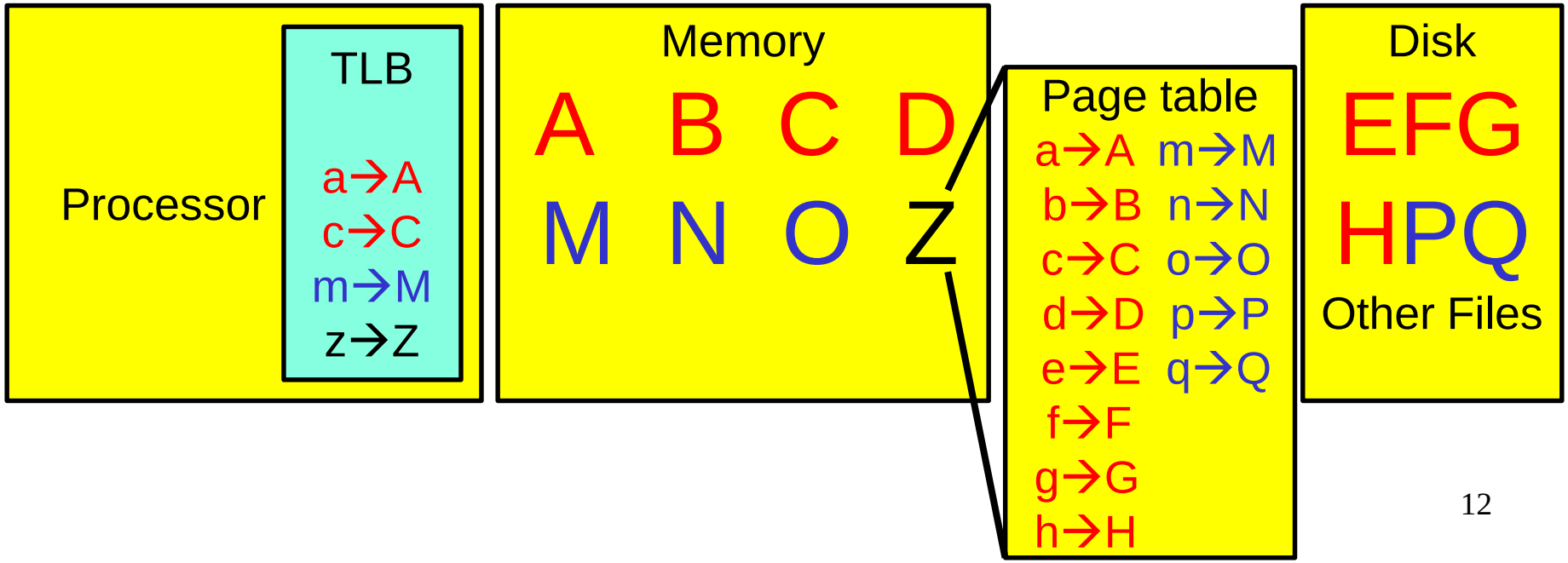
Problem 2

- Build an example toy virtual memory system. Each program has 8 virtual pages. Two programs are running together. The physical memory can store 8 total pages. Show example contents of the physical memory, disk, page table, TLB. Assume that virtual pages take names a-z and physical pages take names A-Z.



Problem 2

- Build an example toy virtual memory system. Each program has 8 virtual pages. Two programs are running together. The physical memory can store 8 total pages. Show example contents of the physical memory, disk, page table, TLB. Assume that virtual pages take names a-z and physical pages take names A-Z.



TLB and Cache

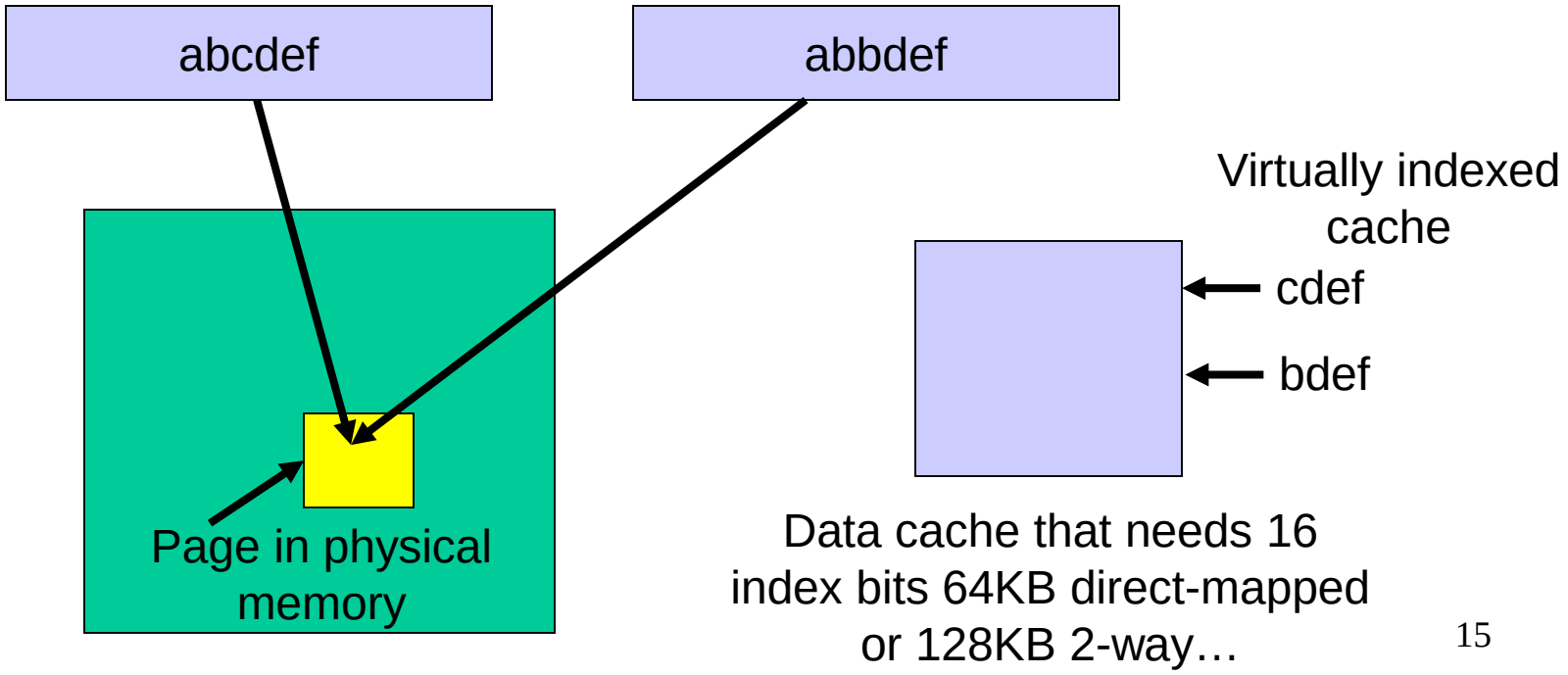
- Is the cache indexed with virtual or physical address?
  - To index with a physical address, we will have to first look up the TLB, then the cache → longer access time
  - Multiple virtual addresses can map to the same physical address – can we ensure that these different virtual addresses will map to the same location in cache? Else, there will be two different copies of the same physical memory word

- Does the tag array store virtual or physical addresses?
  - Since multiple virtual addresses can map to the same physical address, a virtual tag comparison can flag a miss even if the correct physical memory word is present

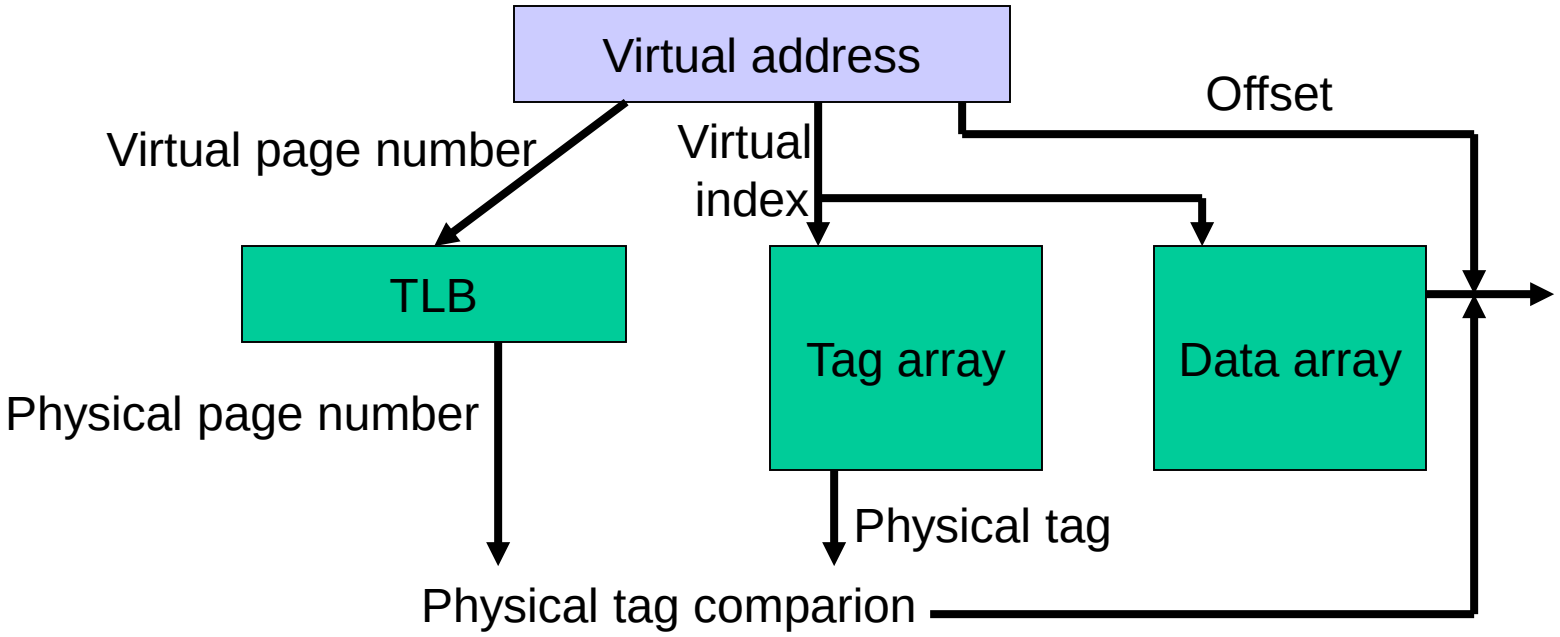
TLB and Cache

Virtually Indexed Caches

- 24-bit virtual address, 4KB page size → 12 bits offset and 12 bits virtual page number
- To handle the example below, the cache must be designed to use only 12 index bits – for example, make the 64KB cache 16-way
- Page coloring can ensure that some bits of virtual and physical address match



Cache and TLB Pipeline



Virtually Indexed; Physically Tagged Cache

Problem 3

- Assume that page size is 16KB and cache block size is 32 B. If I want to implement a virtually indexed physically tagged L1 cache, what is the largest direct-mapped L1 that I can implement? What is the largest 2-way cache that I can implement?

Problem 3

- Assume that page size is 16KB and cache block size is 32 B. If I want to implement a virtually indexed physically tagged L1 cache, what is the largest direct-mapped L1 that I can implement? What is the largest 2-way cache that I can implement?

  There are 14 page offset bits. If 5 of them are used for block offset, there are 9 more that I can use for index.

  512 sets ➔ 16KB direct-mapped or 32KB 2-way cache

Title

- Bullet