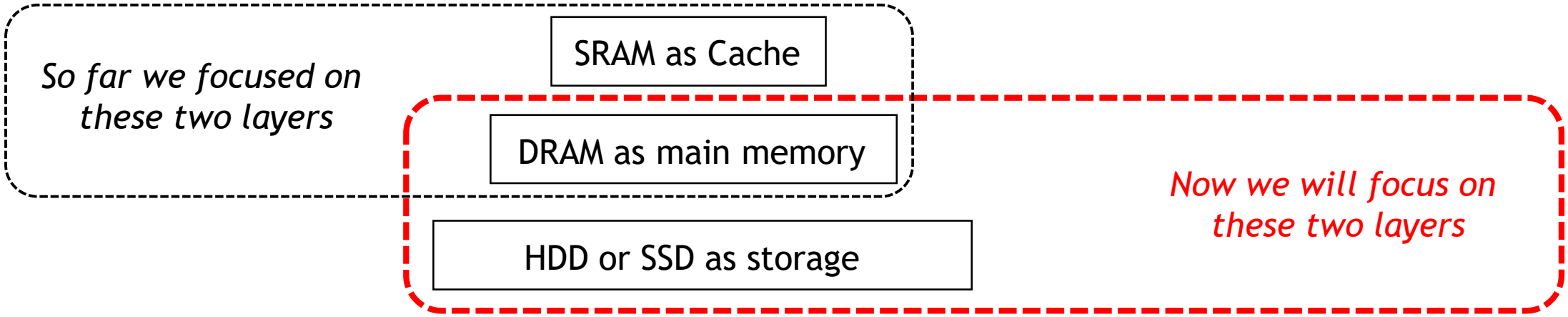
Computer Architecture
(ENE1004)

Lec – 24: Large and Fast: Exploiting Memory Hierarchy (Chapter 5) – 6

Schedule

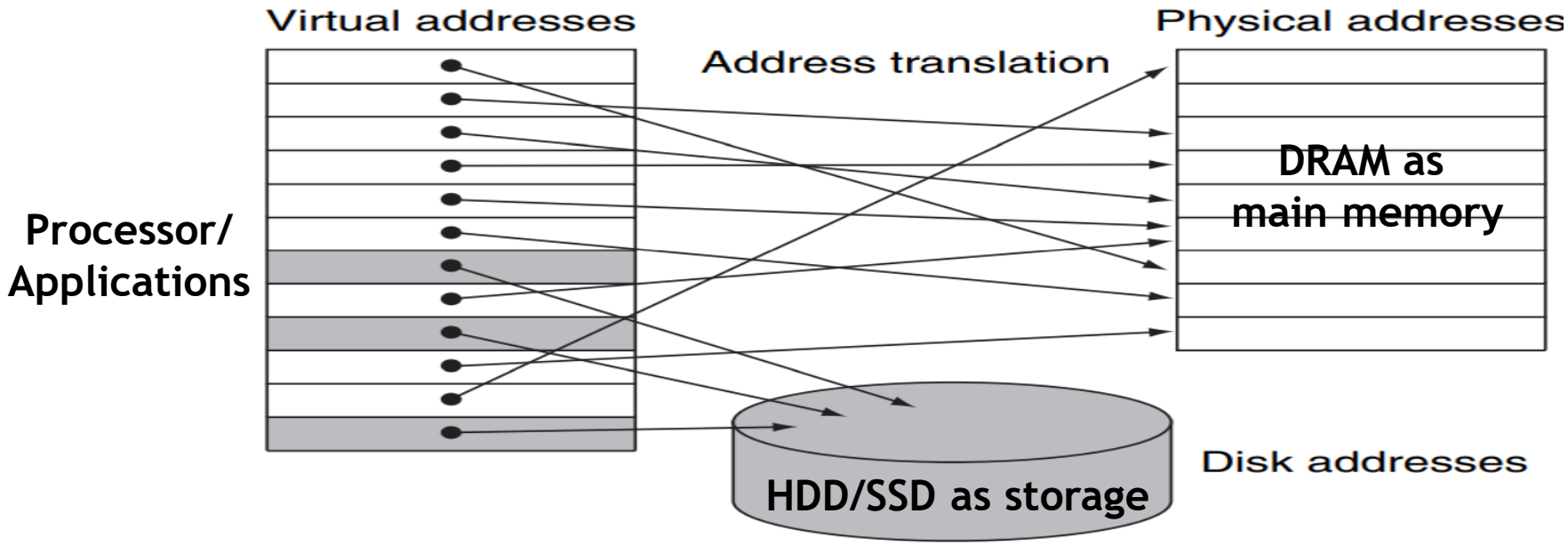
- Final exam: Jun. 19, Monday
  - (24334) 1:25~2:25pm
  - (23978) 2:30~3:30pm
  - Sample questions will be provided by Jun. 17 (Saturday)
- Assignment #2: Jun. 20, Tue. by midnight

Virtual Memory



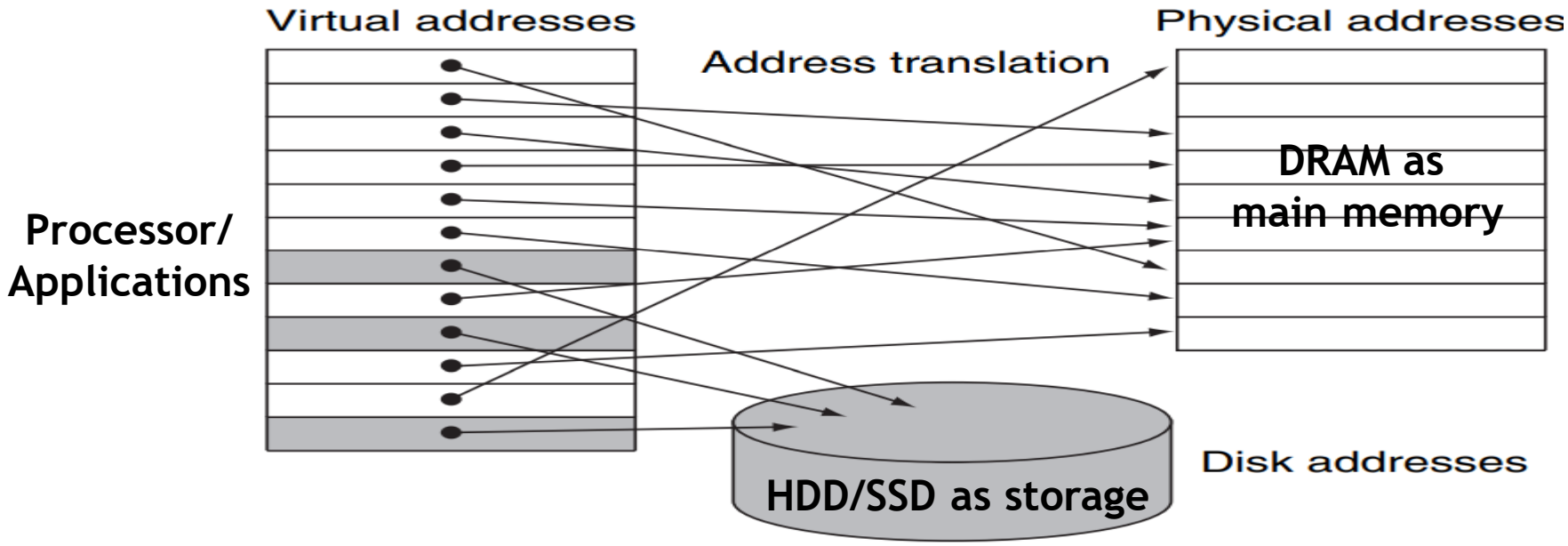
- Now, we will focus on another two-layers memory system: main memory + storage
  - All data sets of applications are stored in the storage
  - The main memory hold part (subset) of all the data sets in the storage
  - Processors feel like the main memory holds all the data sets
  - So, particularly, we call this system “virtual memory”
- Page: The base unit of the virtual memory is called “page” (vs block/line in cache)
- Page fault: A virtual memory miss is called “page fault”

Virtual Address and Physical Address



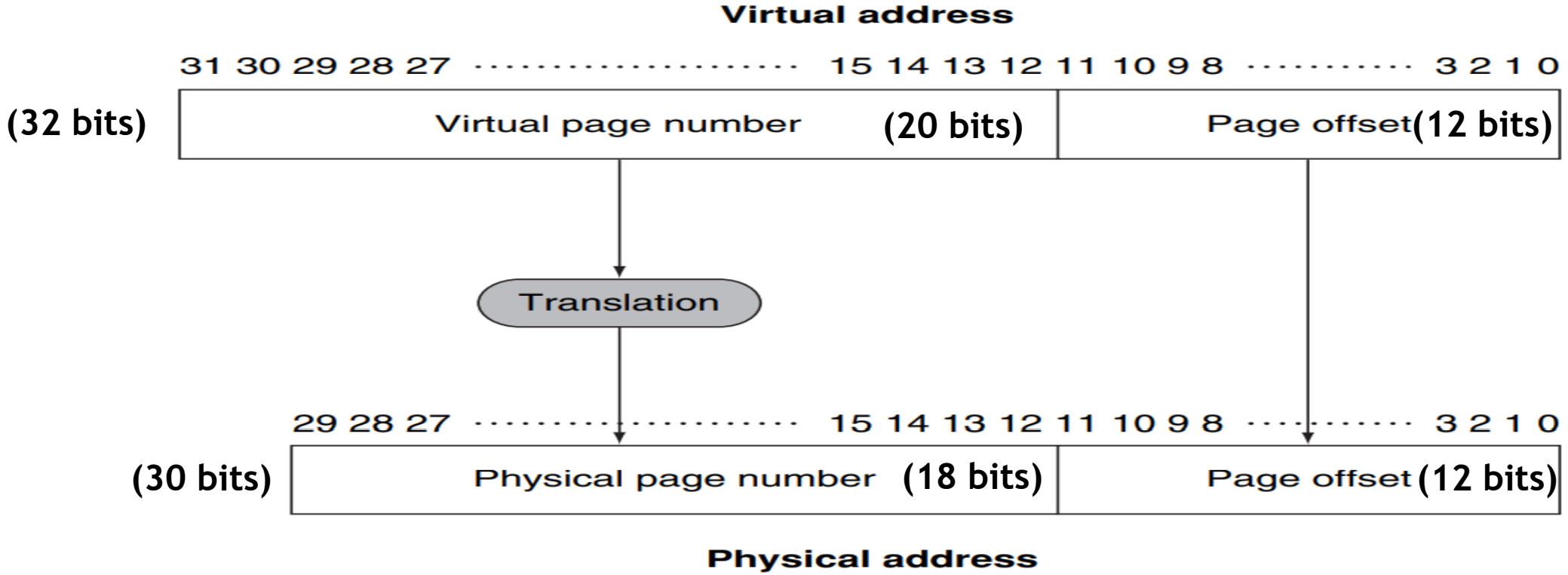
- DRAM can be accessed using its device addresses; we call them "physical addresses"
- However, processor does NOT use these physical addresses
  - Instead, the processor uses "virtual addresses"
  - Memory addresses generated in load/store, branch, and jump instructions are virtual addresses
  - This is because all the data of applications cannot be accommodated by limited capacity of DRAM
- Actually, an application running on the processor has part of its data in the main memory while the remaining data in the storage; but, the processor does NOT care about it and just uses the virtual addresses of the application

Address Translation: Virtual → Physical Address (1)



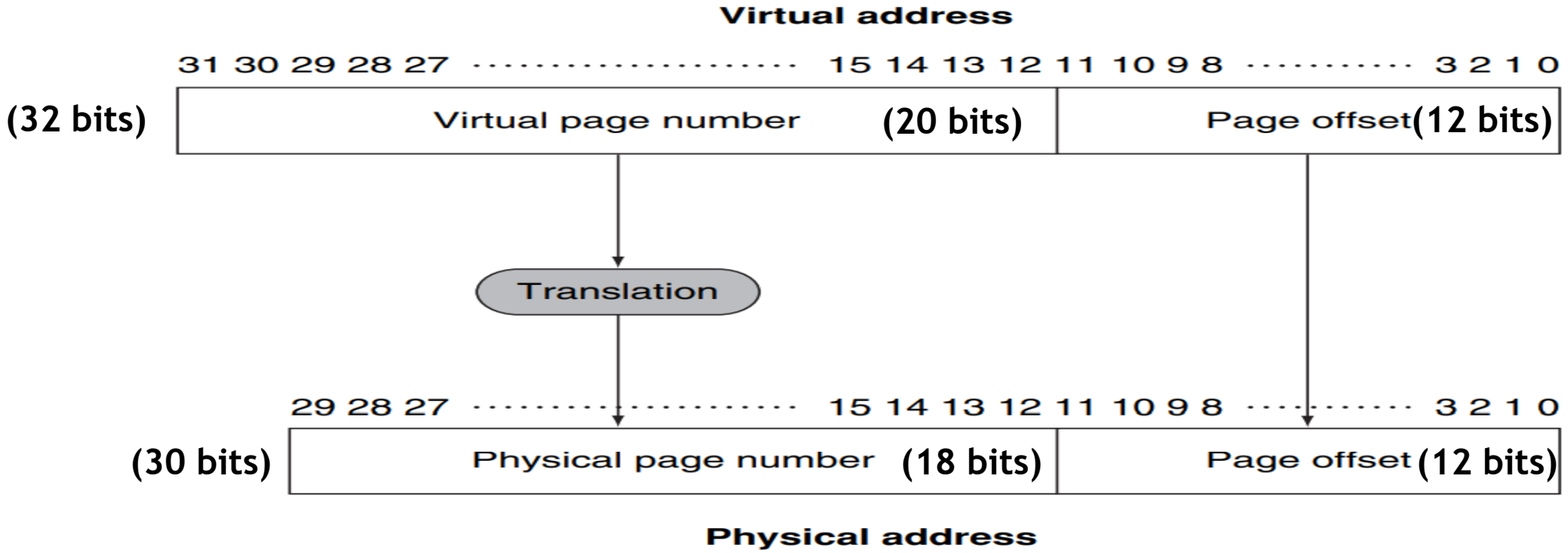
- To request a data from the main memory, an address translation is required
  - Processor requests a data item with its virtual address
  - DRAM can be accessed using its physical address
- Note that a data of an application can be in either the main memory or the storage
  - If the data is in the main memory, the virtual address → the physical address
  - If the data is in the storage, the virtual address → the disk address

Address Translation: Virtual → Physical Address (2)



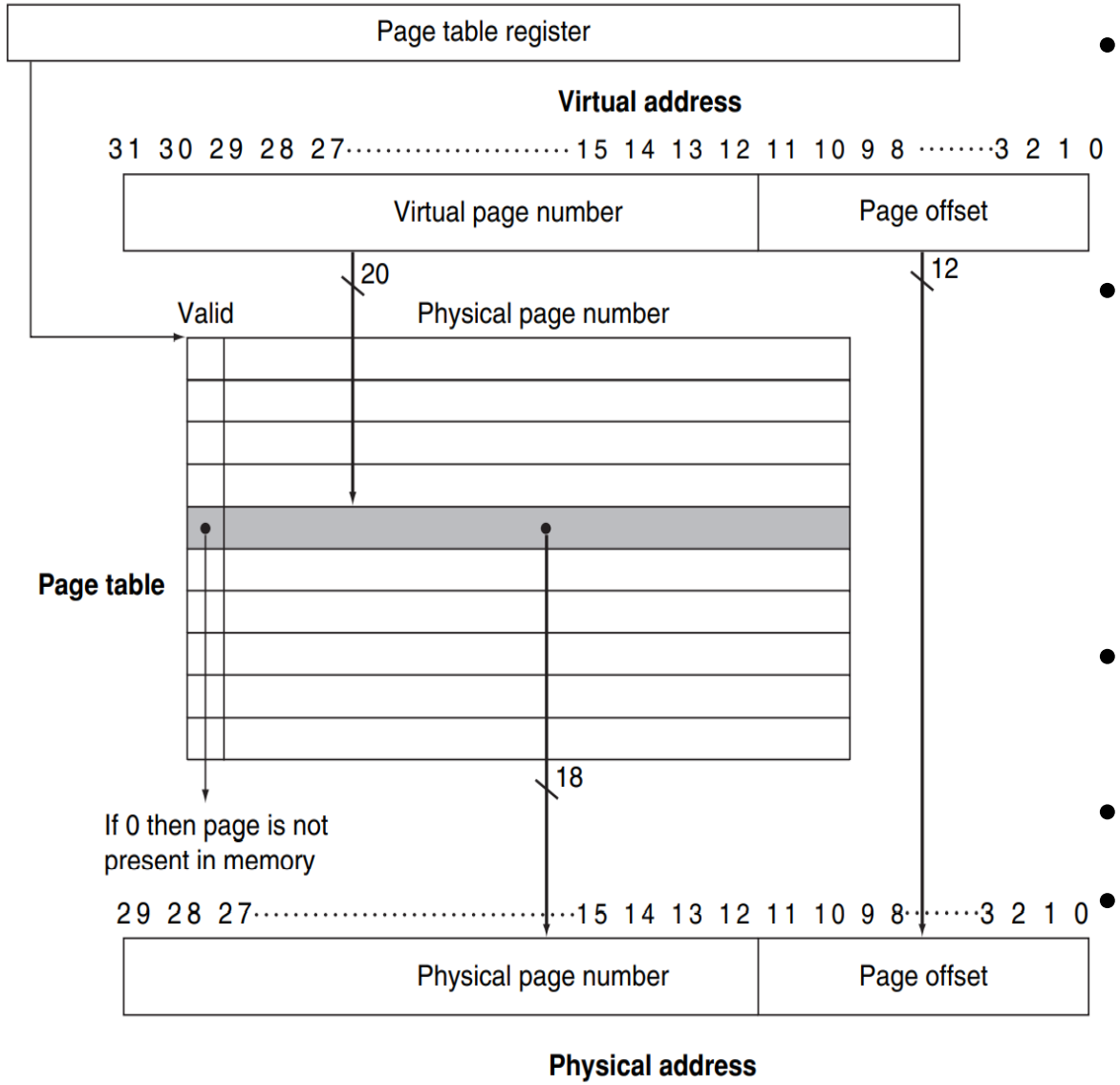
- Recall that the base unit in the virtual memory is "page"
  - The virtual address for a page can be divided into "virtual page number" + "page offset"
  - The page offset indicates a specific byte within the page (recall "byte offset" in cache)
  - So, we can infer the size of a page from the number of bits for the page offset
  - Here, the lower 12 bits of a 32-bit address is used as page offset; the page size is 2^{12} bytes (4 KiB)
- The upper portion of the virtual address indicates "virtual page number"

Address Translation: Virtual → Physical Address (3)



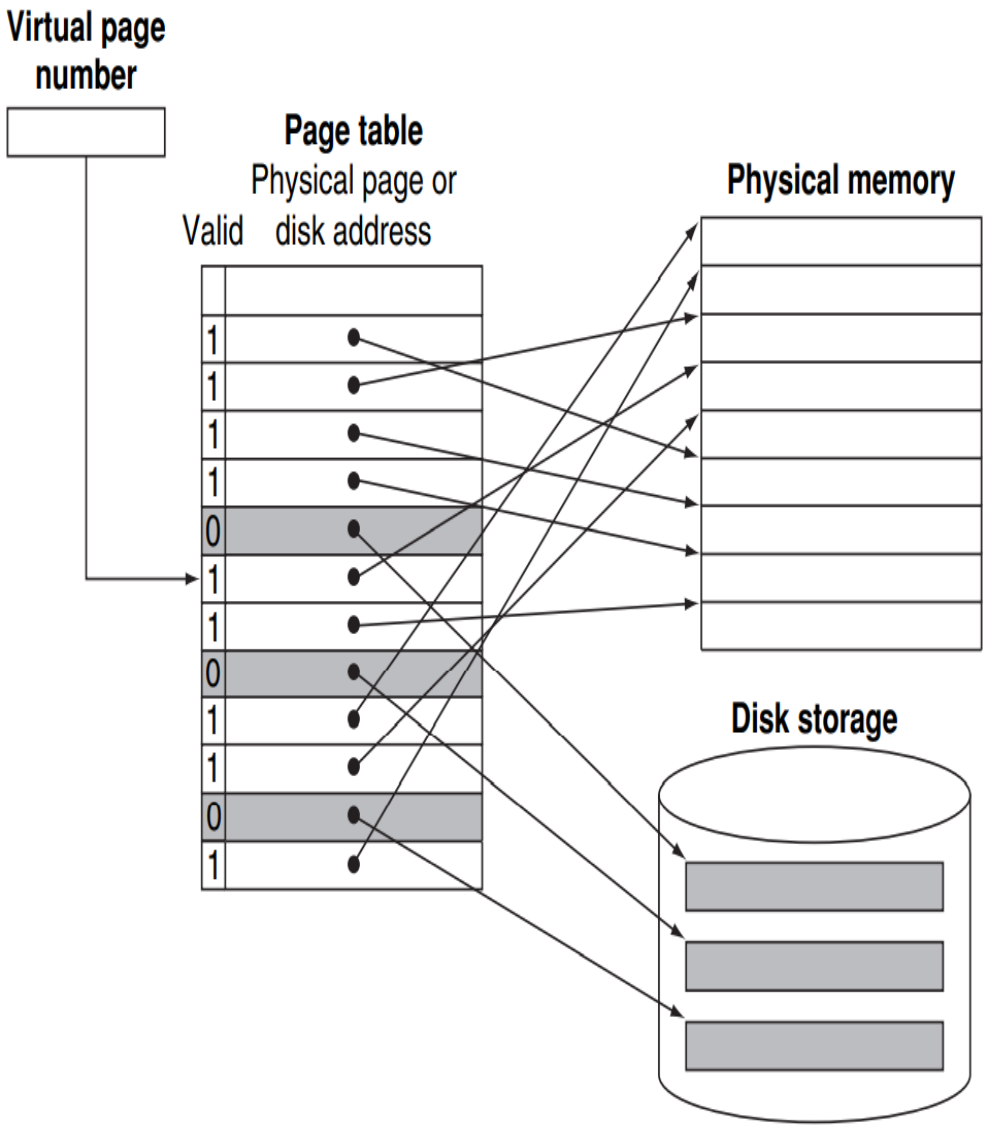
- Address translation: virtual page number → physical page number
  - The page offset does NOT change; it is for specifying bytes within a page
  - In general, # of virtual pages is much larger than # of physical pages (sizes of apps >>> DRAM)
- Here, we can infer the DRAM size from the number of bits in the physical address
  - The physical address is 30-bit (2^{30} bytes = 1 GiB)
  - There are 2^{18} pages, each of which is 2^{12} bytes, in this 1 GiB DRAM

Page Table



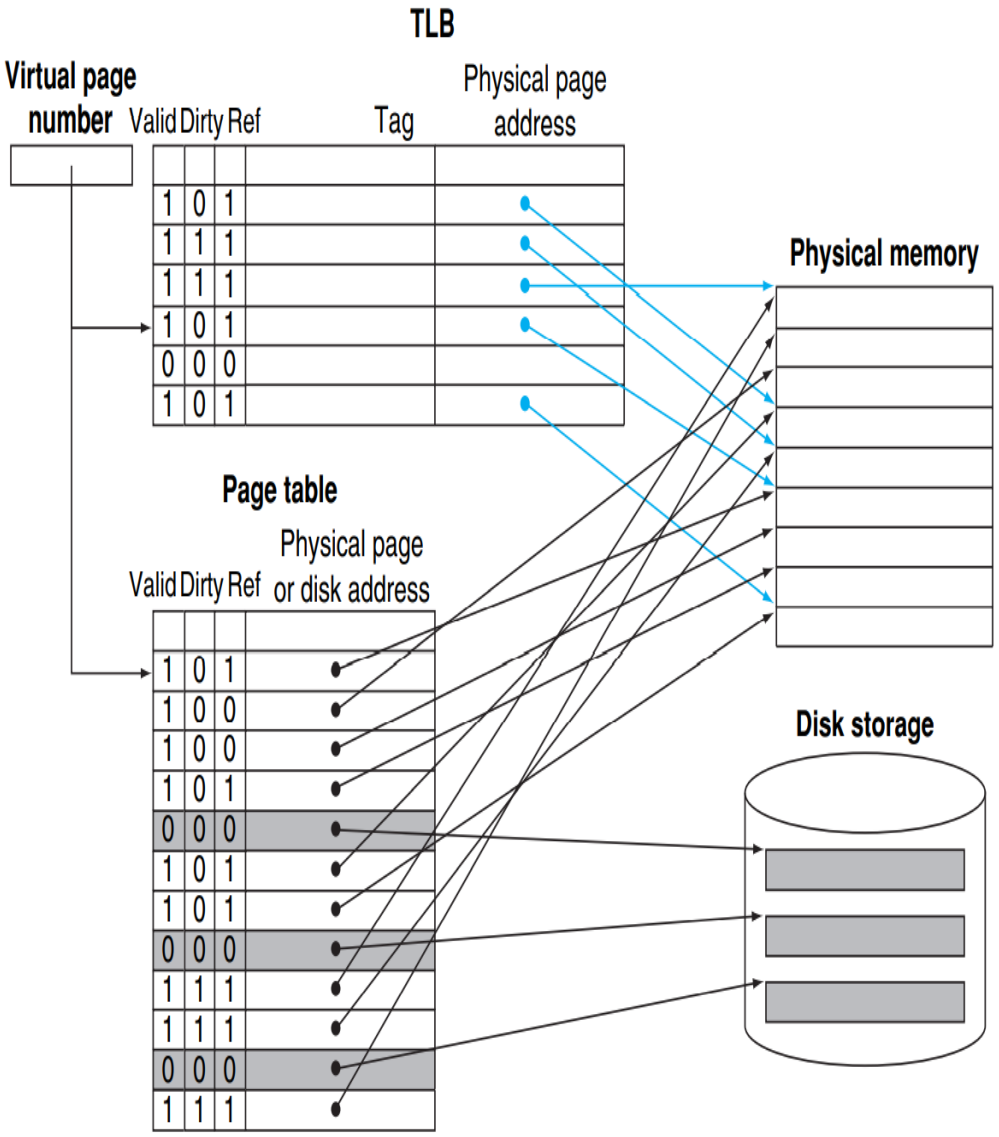
- Page table - (virtual page # : physical page #)
  - An entry is indexed with the virtual page #
  - Each holds the corresponding physical page #
- Each also has a valid bit, which tells whether the page is in the main memory or not
  - If it is set to “1”, the page in the main memory
  - If it is set to “0”, the page in the storage; so, address translation is not required
- Note that the page offset does not change in the virtual-physical address translation
- Page table is kept in the main memory
- The processor knows the location of the page table in the main memory
  - Page table register

Given a Memory Request



- A memory request with its virtual address is given
- (1) It goes to the page table and the entry indexed by the virtual page number
- (2-1) If the valid bit is set to “1”, the physical address is generated
  - The requested page is in the main memory
- (3-1) Using the physical address, the actual page is read from the main memory
- (2-2) If the valid bit is set to “0”, the address translation is not required
  - The requested page is not in the main memory
  - We call this miss “page fault”
- (3-2) To handle a page fault, the page is brought from the storage and written into the main memory
  - The related topic (I/O) is out of our interest in this course

Translation Lookaside Buffer (TLB)



- A memory request needs "two" different accesses to the main memory
  - (1) It needs to access the page table for physical address
  - (2) It needs to read the data using the physical address
- Modern computers keep frequently-accessed page table entries in a fast hardware unit
  - We call that unit translation lookaside buffer (TLB)
  - TLB holds a subset of the page table
  - TLB does NOT hold frequently-accessed data
- First, the virtual page number is given to the TLB
  - If it is "hit", it does not have to go to the page table, which in turn can avoid an access to the main memory
  - If it is "miss", it should go to the page table in the main memory
  - Here, a "hit" or "miss" does not mean whether the page is in the main memory or not