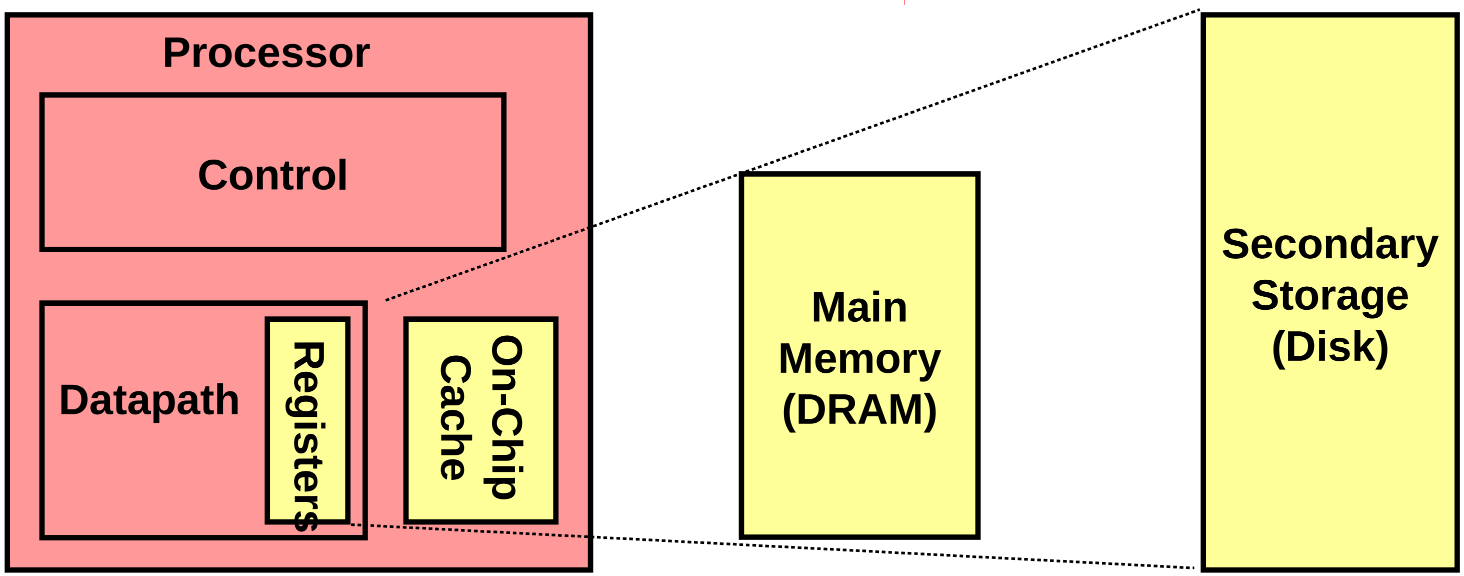
# Computer Organization
# 計算機組織

# Virtual Memory

國立成功大學資訊工程學系

105 年度第二學期

## We now go to this part




| | | | | |
|---|---|---|---|---|
| Speed (ns): | 1‘s | 10‘s | 100’s | 10,000,000’s (10’s ms) |
| Size (bytes): | 100‘s | K’s | M’s | G’s or T’s |

## So far the memory system we learn…

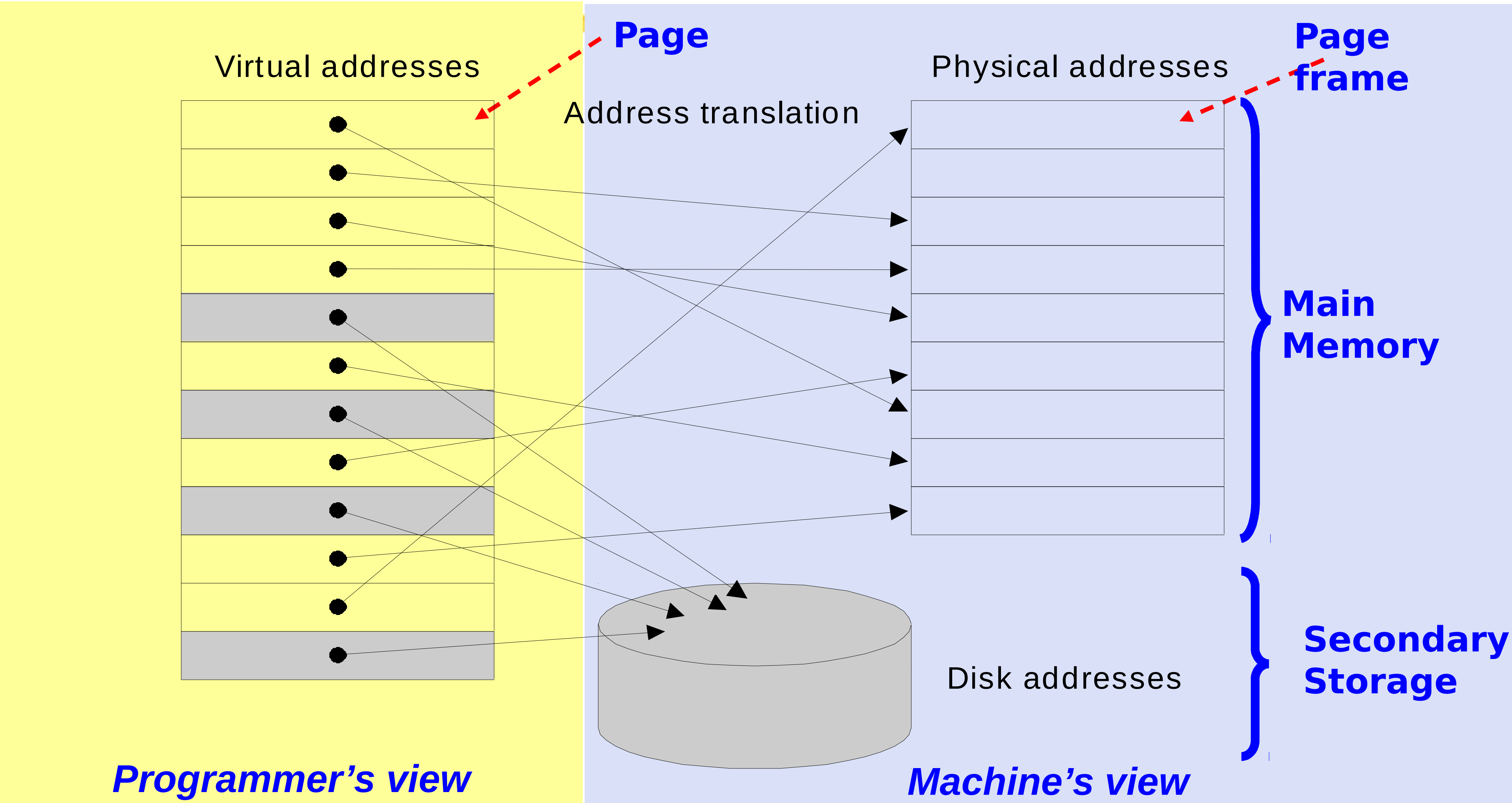
# Virtual Memory (1/2)

- Allow many programs to run at once

- Virtual memory
  - Every program has its own address space (can be greater than physical memory), starting at address 0
  - only accessible to itself
  - can run anywhere in physical memory

- Virtual memory is a technique implementing the translation from virtual space to physical space
  - OS runs all the time and allocates physical resources

# Virtual Memory (2/2)




**Mapping of pages from a virtual address to a physical address or disk address**

# Issues in Virtual Memory

- **Size of a page**

  ***Page*: a basic unit of memory blocks transferring between main memory and secondary storage (e.g., disk)**

- **Placement policy**

  **where to place a page in *main memory***

- **Replacement policy**

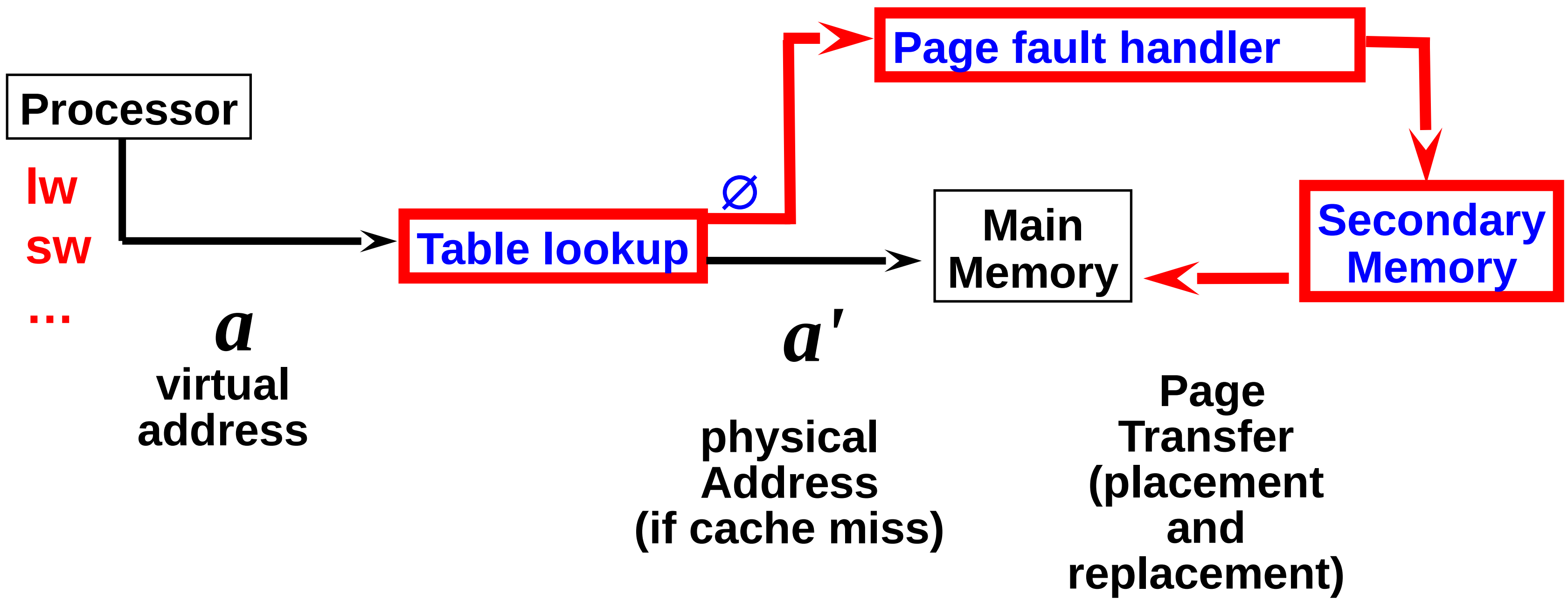
  **replace pages if *cannot* find any free space in main memory**

- **When to fetch a page?**

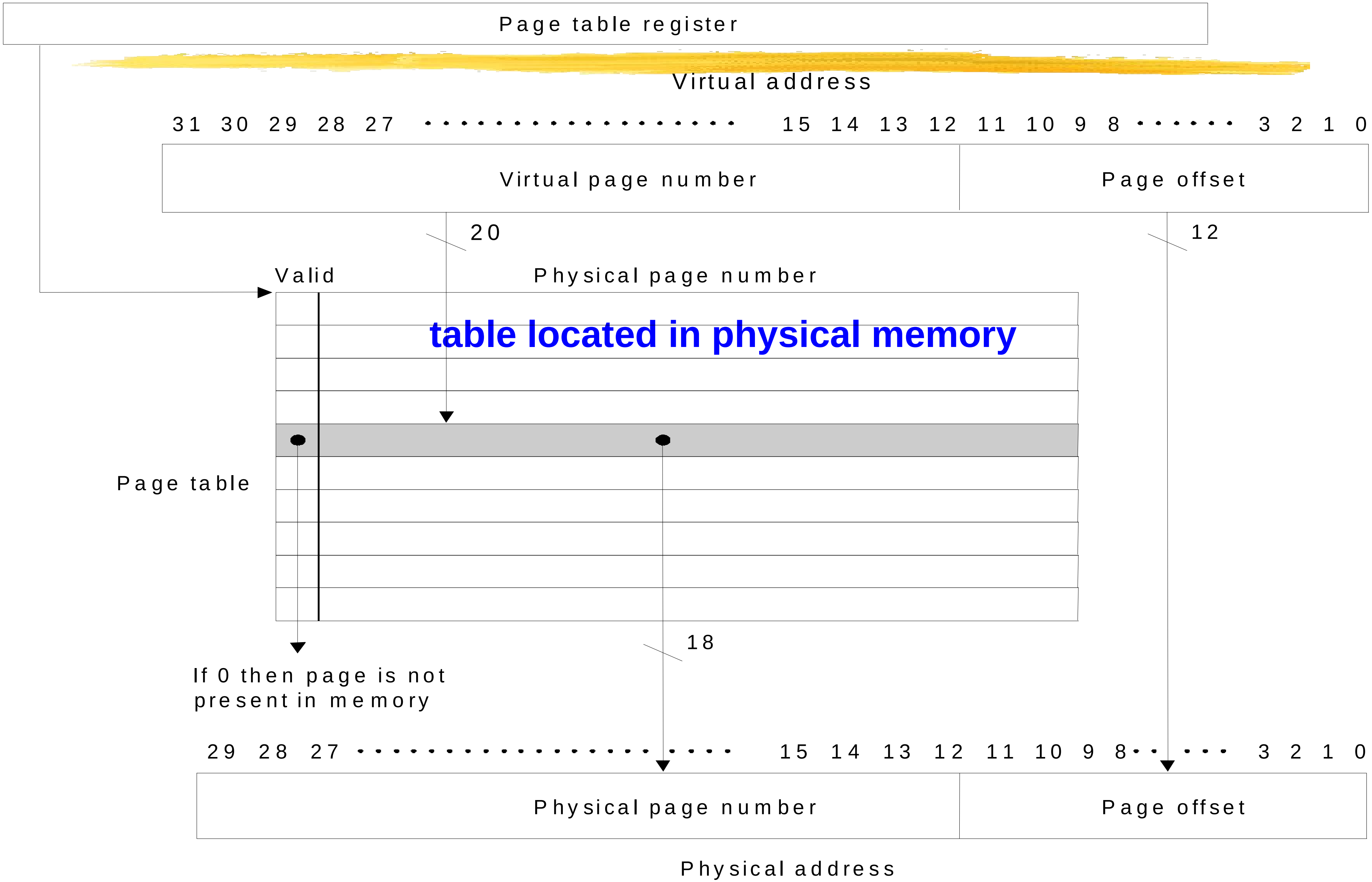
  **may fetch when we miss**

# Address Translation: Virtual to Physical

- Given a virtual address $a$, return the physical address ( $a'$ ) corresponding to $a$



# Paging with 4K-byte Page Size (assume 1GB main memory)



# Example Problem

- **Assume:**
  - **32-bit virtual address**
  - **4 KB page size**
  - **4 bytes per page table entry**
  - **The page table maps page 0 to frame 10, page 1 to frame 8, page 2 to frame 100, page 3 to frame 1, …**

- **Q1: What is the total page table size if we want to be able to access all of the virtual memory?**
- **Q2: What is the page no. of the virtual address 8196?**
- **Q3: What is the physical address corresponding to the virtual address 8196?**

# Solution

- No. of page table entries = address space size / page size = $2^{32} / 2^{12} = 2^{20}$
- (Maximum) Size of page table = No. of entries $\times$ entry size = $2^{20} \times$ 4 bytes = 4 MB
  - However, a program typically takes a few of pages
- 8196/4096=2.x (i.e. address 8196 is at the 2nd page)
- Physical address is 100*4096+4

# Page Fault

- Page fault means that page is not resident in memory
- Then, hardware must trap to the operating system so that it can remedy the situation
  - "May" need to pick a page to discard (may write the replaced page to disk)
    - i.e., page replacement policy
    - Note: if a page in "main memory" is updated, then it is "dirty"
  - Load the requested page in from disk
    - i.e., placement policy
  - Update the page table
  - Resume to program so HW will retry and succeed!

# So Far

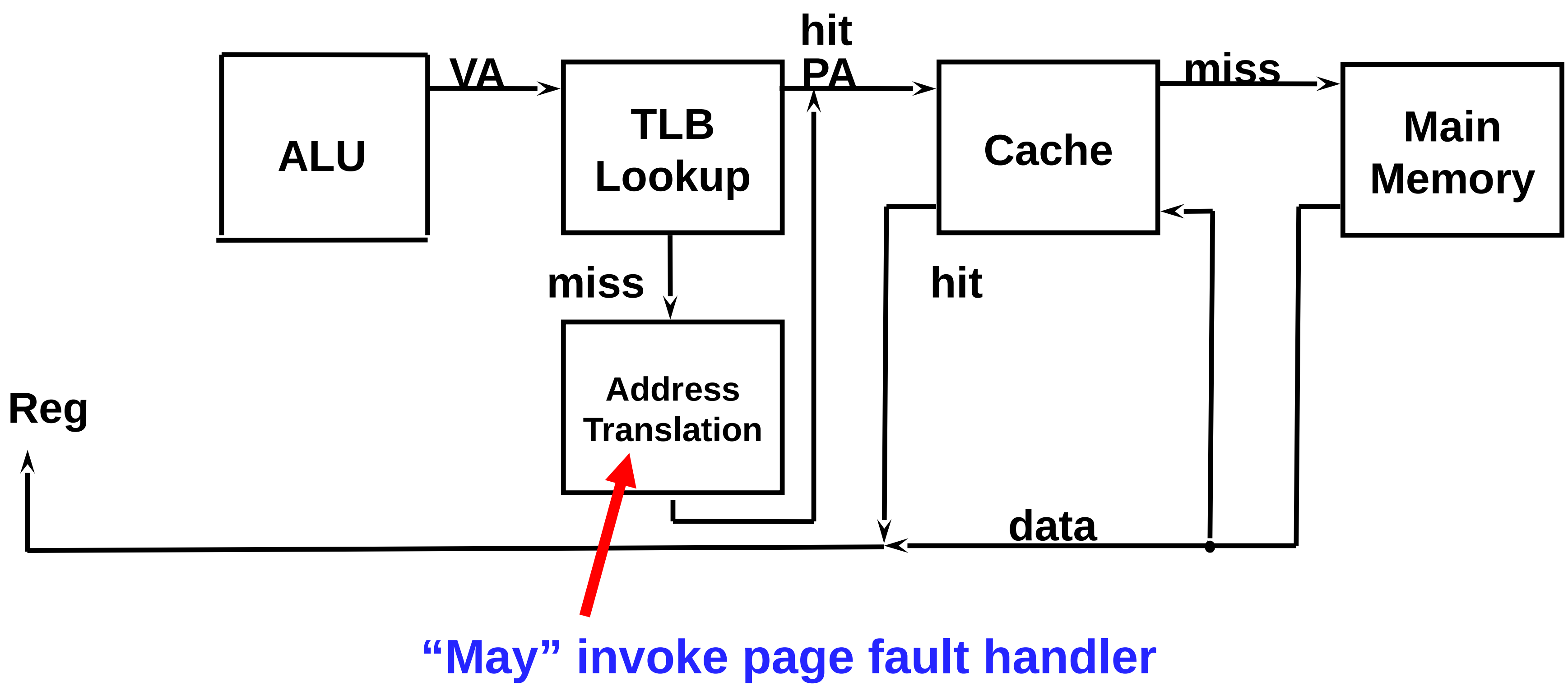
- We assume that we are given a virtual memory address to access the virtual memory system

- We have not integrate virtual memory with processor, i.e., how does a processor send out a virtual memory address due to `lw` or `sw`?

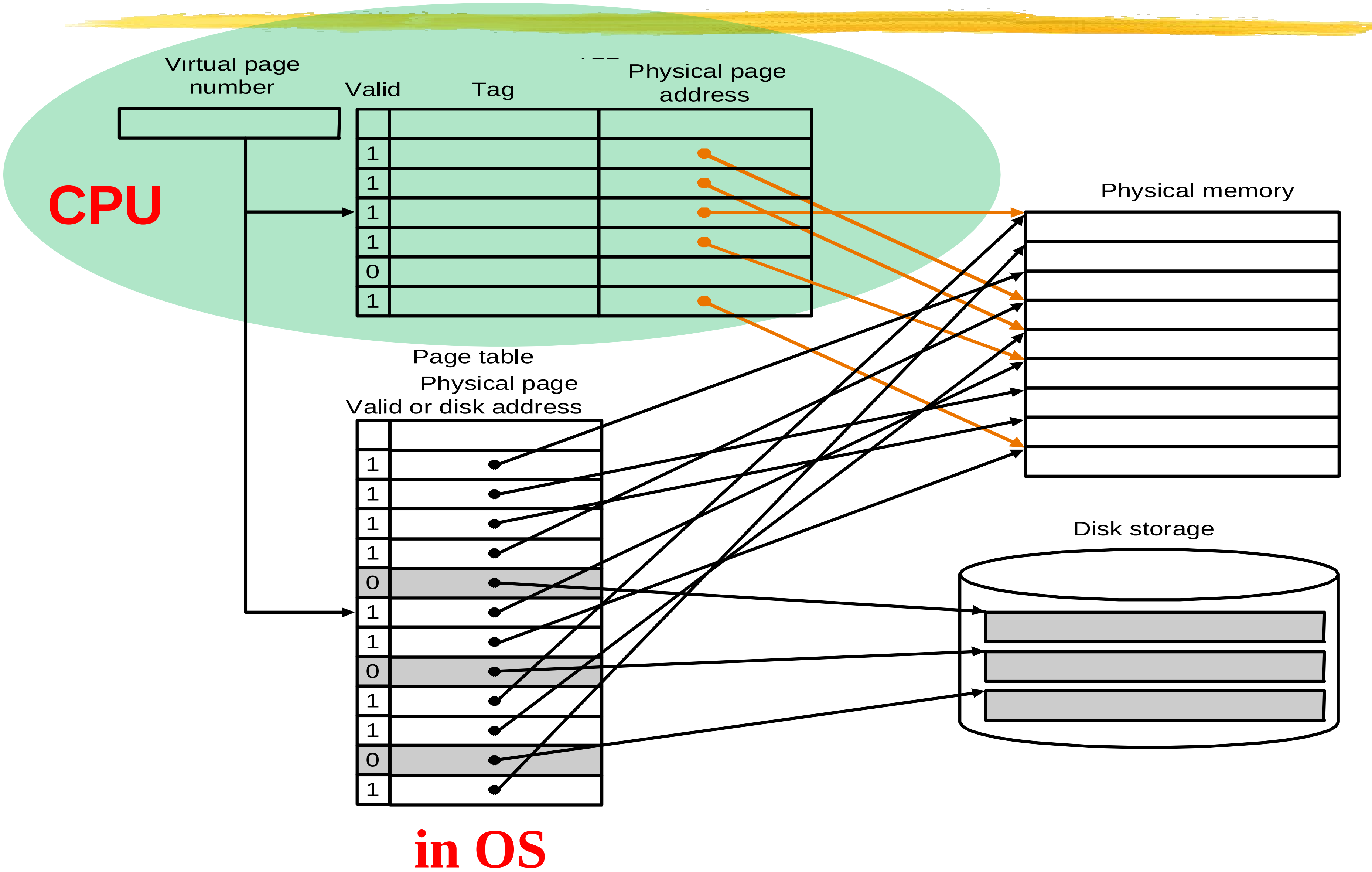
- Put everything together in the following…

# Practical Address Translation

- To speedup the address translation, typical processor implements a hardware page table cache, namely **Translation Lookaside Buffer (TLB)**



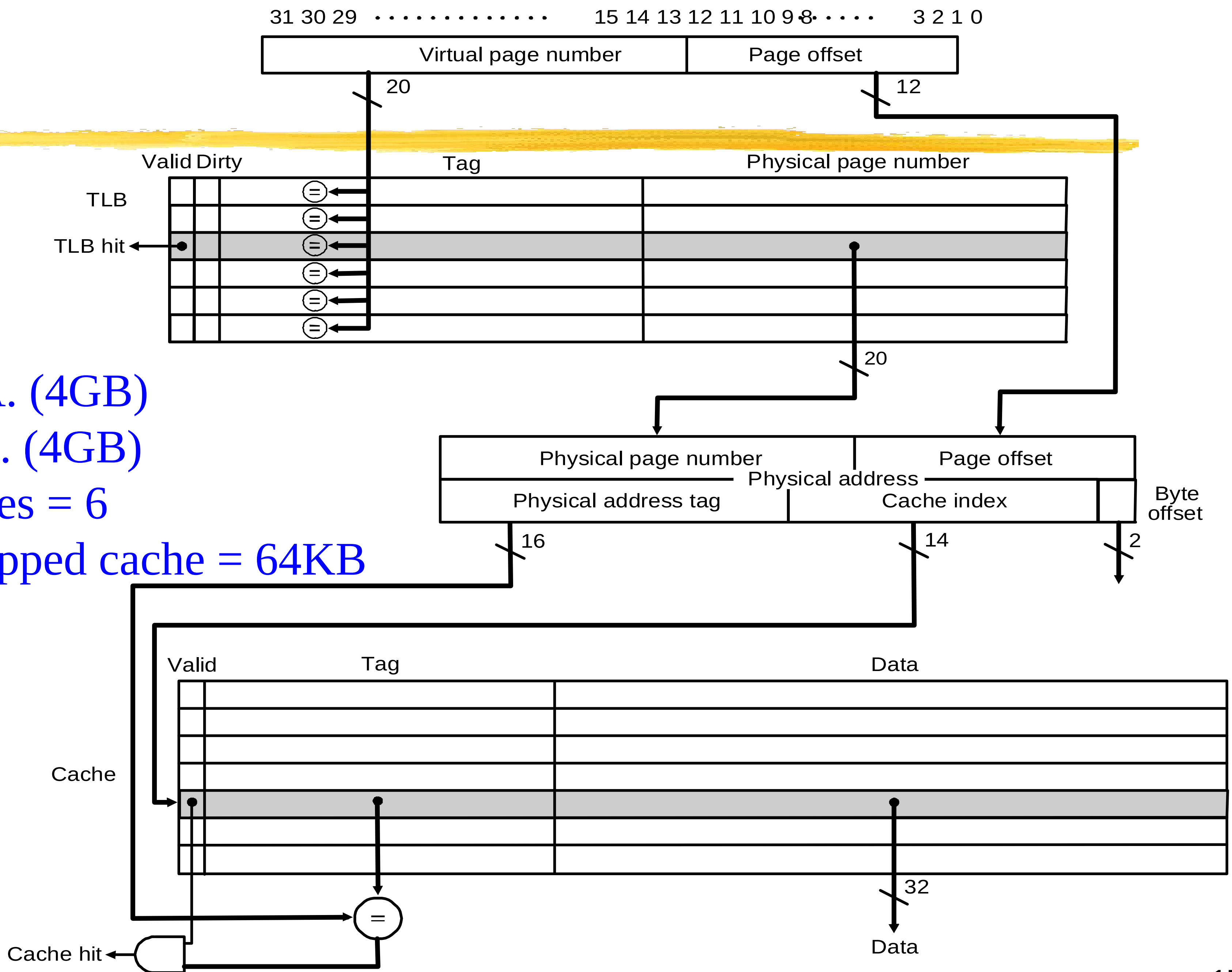
# TLB



CPU
Virtual page number
Valid
Tag
Physical page address
1
1
1
1
0
1
Physical memory
Page table
Physical page
Valid or disk address
1
1
1
1
0
1
1
0
1
1
0
1
Disk storage
in OS

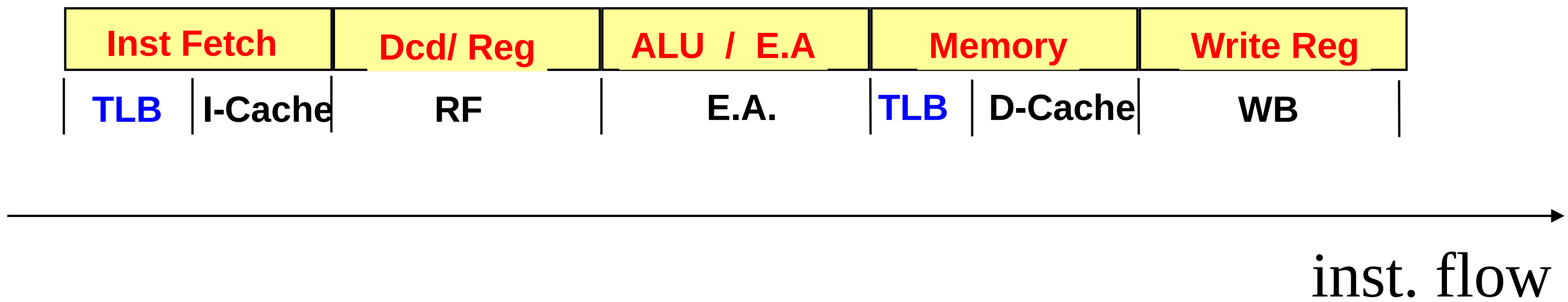
# More on TLB

- TLB can be organized as fully associative, set associative, or direct mapped
  - TLBs are typically small in size, typically < 128~256 entries
- TLB hit on write:
  - Toggle dirty bit (write back to page table on replacement)
- TLB miss:
  - If only TLB miss => load "page table entry" into TLB
  - If page fault also => OS exception
- Inclusion property
  - If L1 is a upper level "data" memory of L2, then "data" in L1 must appear in L2




- 32-bit V.A. (4GB)
- 32-bit P.A. (4GB)
- TLB entries = 6
- Direct mapped cache = 64KB

# TLB in Pipeline



# TLB + Cache

| Cache | TLB | Virtual Memory | Possible? Conditions? |
|---|---|---|---|
| Miss | Hit | Hit | Yes; but page table never checked if TLB hits |
| Hit | Miss | Hit | TLB miss, but entry found in page table;after retry, data in cache |
| Miss | Miss | Hit | TLB miss, but entry found in page table; after retry, data miss in cache |
| Miss | Miss | Miss | TLB miss and is followed by a page fault; after retry, data miss in cache |
| Miss | Hit | Miss | impossible;<br>not in TLB if page not in memory |
| Hit | Hit | Miss | impossible;<br>not in TLB if page not in memory |
| Hit | Miss | Miss | impossible;<br>not in cache if page not in memory |