COMPUTER ARCHITECTURE

Code: CT173

Part XI: Virtual Memory

MSc. NGUYEN Huu Van LONG

Department of Computer Networking and Communication,

College of Information & Communication Technology,

CanTho University

Agenda

- Context before having Virtual Memory
  - Issues
  - Challenges
- Virtual Memory appearance
  - Objects
  - Benefits
- Virtual Memory operation
  - Page Table
    - Basic concepts
    - Address translating
    - Types
  - Memory Management Unit with Translation Lookaside Buffer
- Big picture co-working between Virtual Memory and Cache Memory

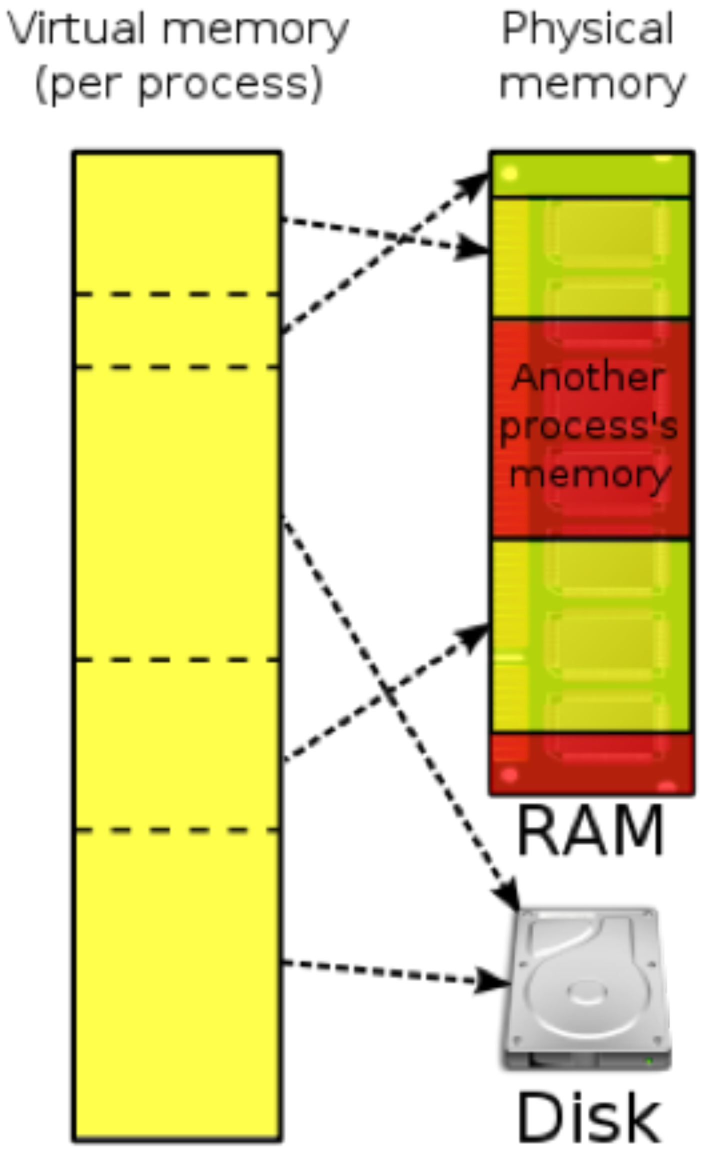
Before Virtual Memory

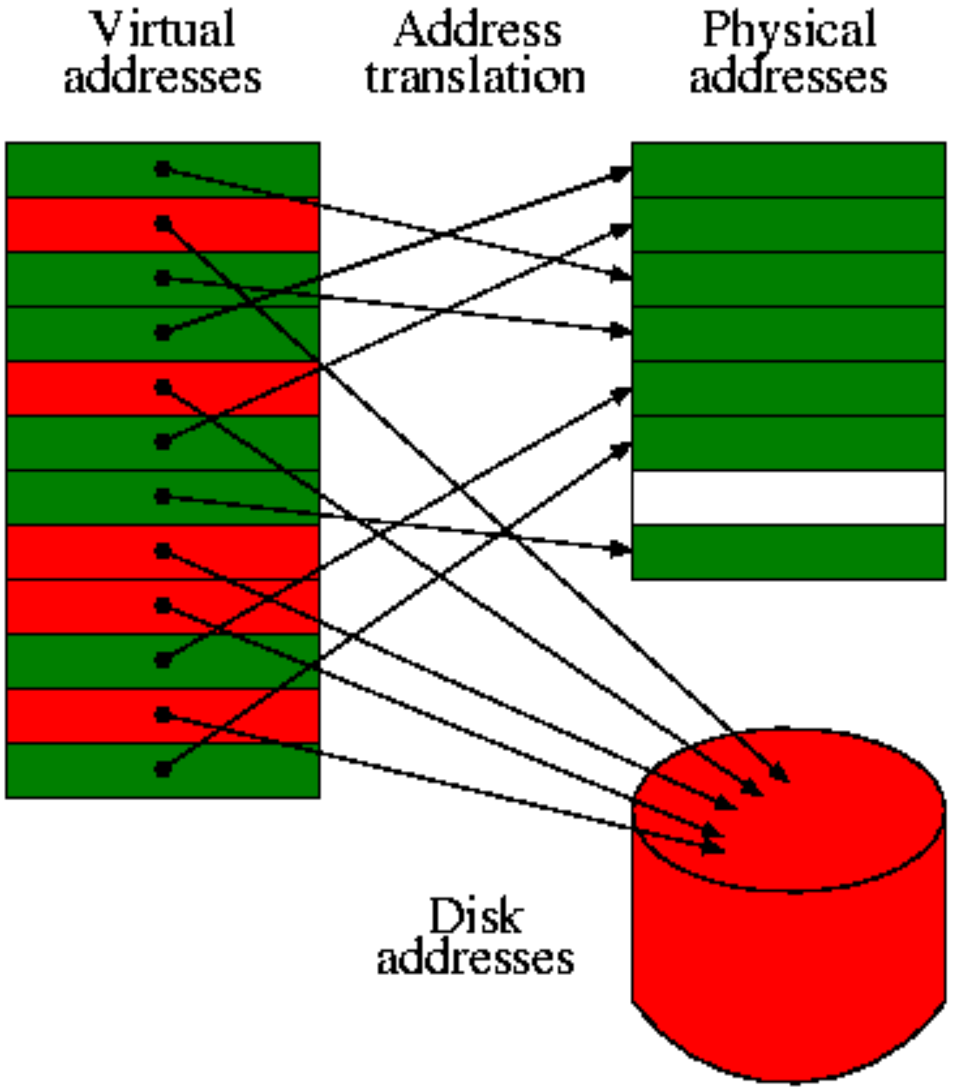
- Before the development of the virtual memory technique, *programmers in the 1940s and 1950s had to manage directly two-level storage* such as Main Memory (RAM) and Secondary Memory (Disk)
  - The program could be too big to fit into the available RAM and RAM is expensive at these days → Solution: splitting the program in to pieces called **Overlays**
  - Overlay 0 would start first followed by the next overlay (1,2,3…).
  - The *overlay were kept on Disk* and *swapped in and out of the RAM* by the operating system *DINAMICALLY*
  - The work of swapping Overlays in and out was done by the system, but the LABORIOUS work of splitting the program in to Overlays had to be done by the programmer
- Issues summary:
  - Primary memory is limited in size and expensive
  - Hard working in controlling the program
  - Mechanism to manage both data on RAM and on Disk
- Potential or Challenge:
  - How about multiprogramming and multitasking?
  - How about memory isolation and protection?
  - How about system security?

Virtual Memory

- **Objects of Virtual Memory**
  - Propose a kind of *unified address for both data in RAM and in Disk* which is *managed by operating system* → Virtual Address
  - Propose a kind of data structure to store the Virtual Addresses of (a) Program(s) in RAM → Page Table
  - Propose a kind of unit for both data in RAM and in Disk → Page
  - Propose a mechanism to translate Virtual addresses to the Physical addresses → Memory mapping
- **Main benefits of Virtual Memory**
  - Make the computer system shares memory efficiently when executing many programs at the same time
  - Eliminate the restriction in size of program could be run by computer system
  - Protect program to interfere each other in memory area by distributing each suitable memory area to each program
- Virtual memory practically treats memory as a cache for the disk, where blocks in this cache are called "pages"

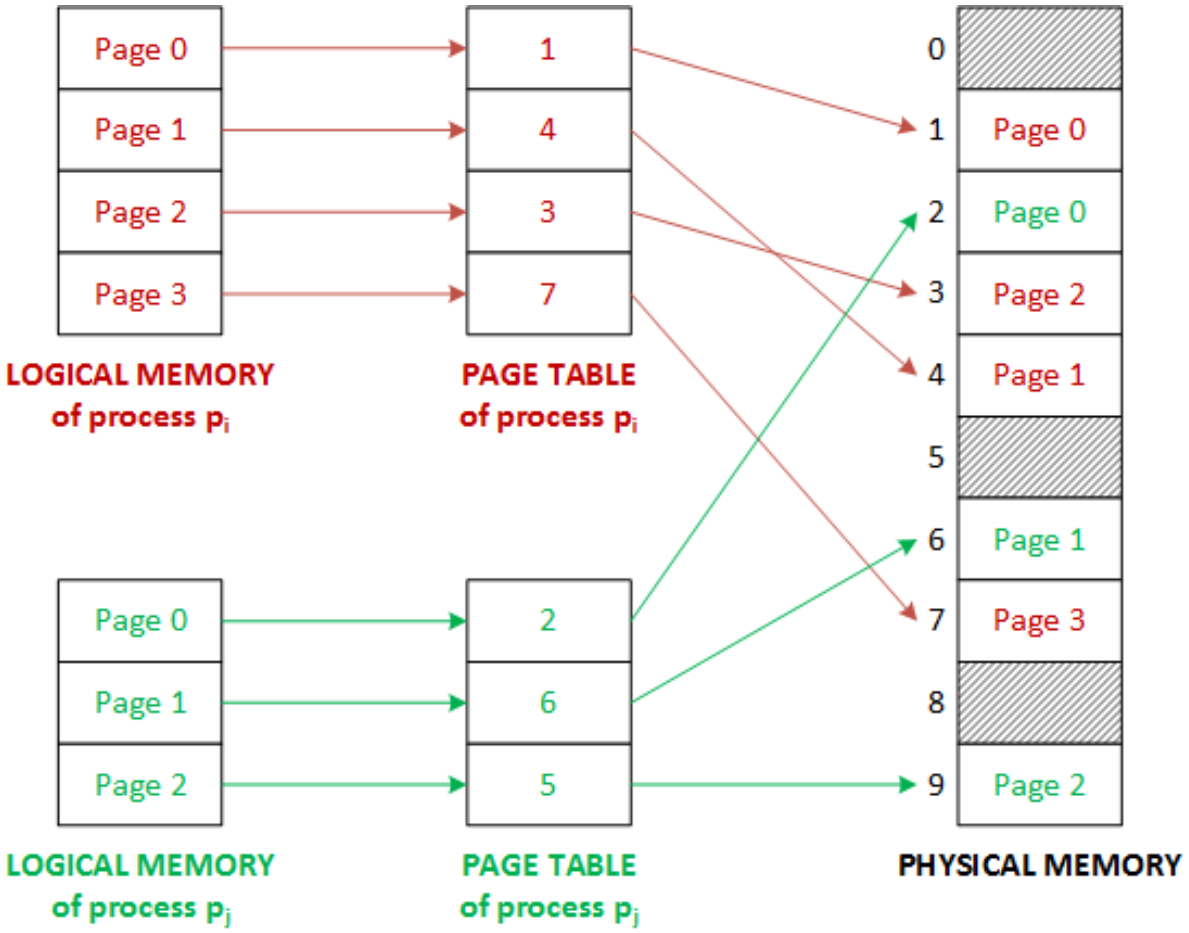
Virtual Memory





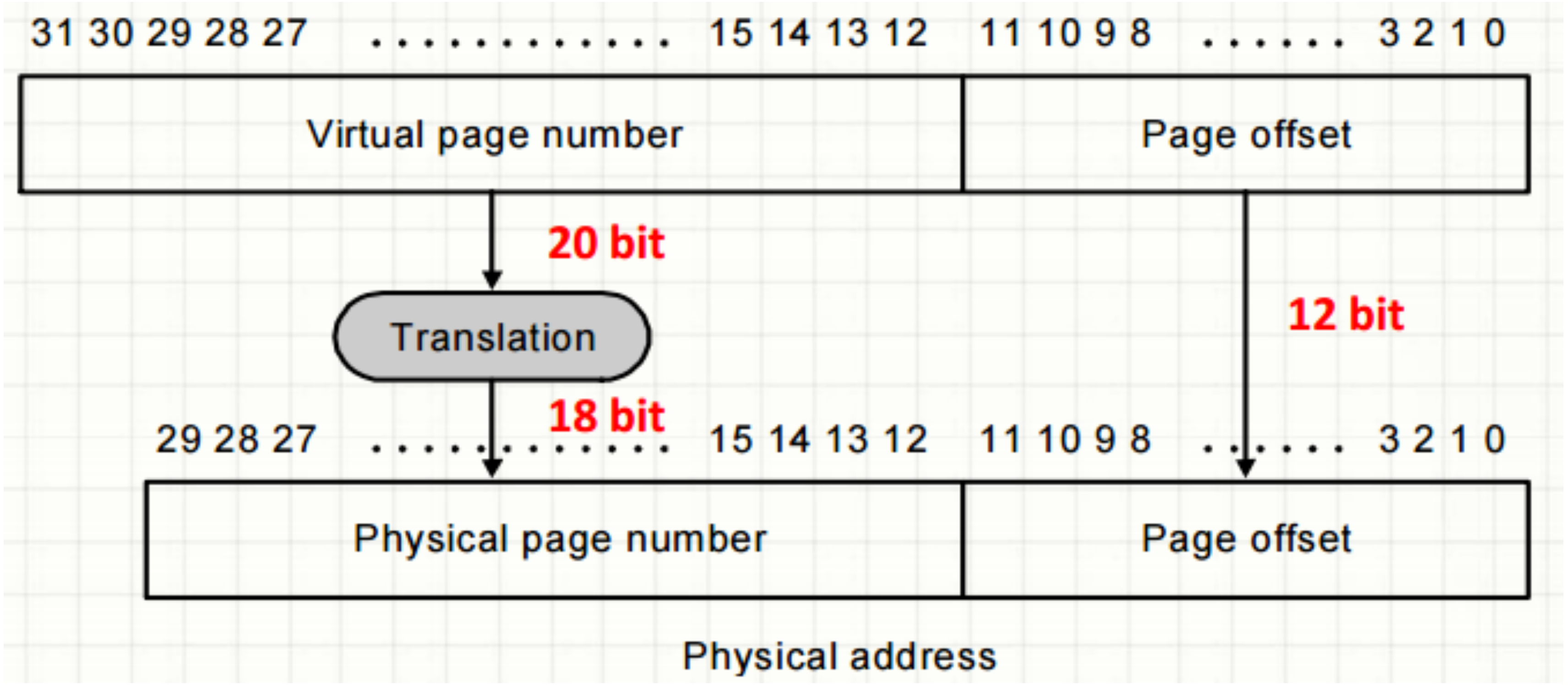
Page Table

- The translation mechanism between virtual pages to physical pages is based on a Page Table associated **to each process**:
  - Each process needs its own page table
  - Page Table for each process is located in physical memory



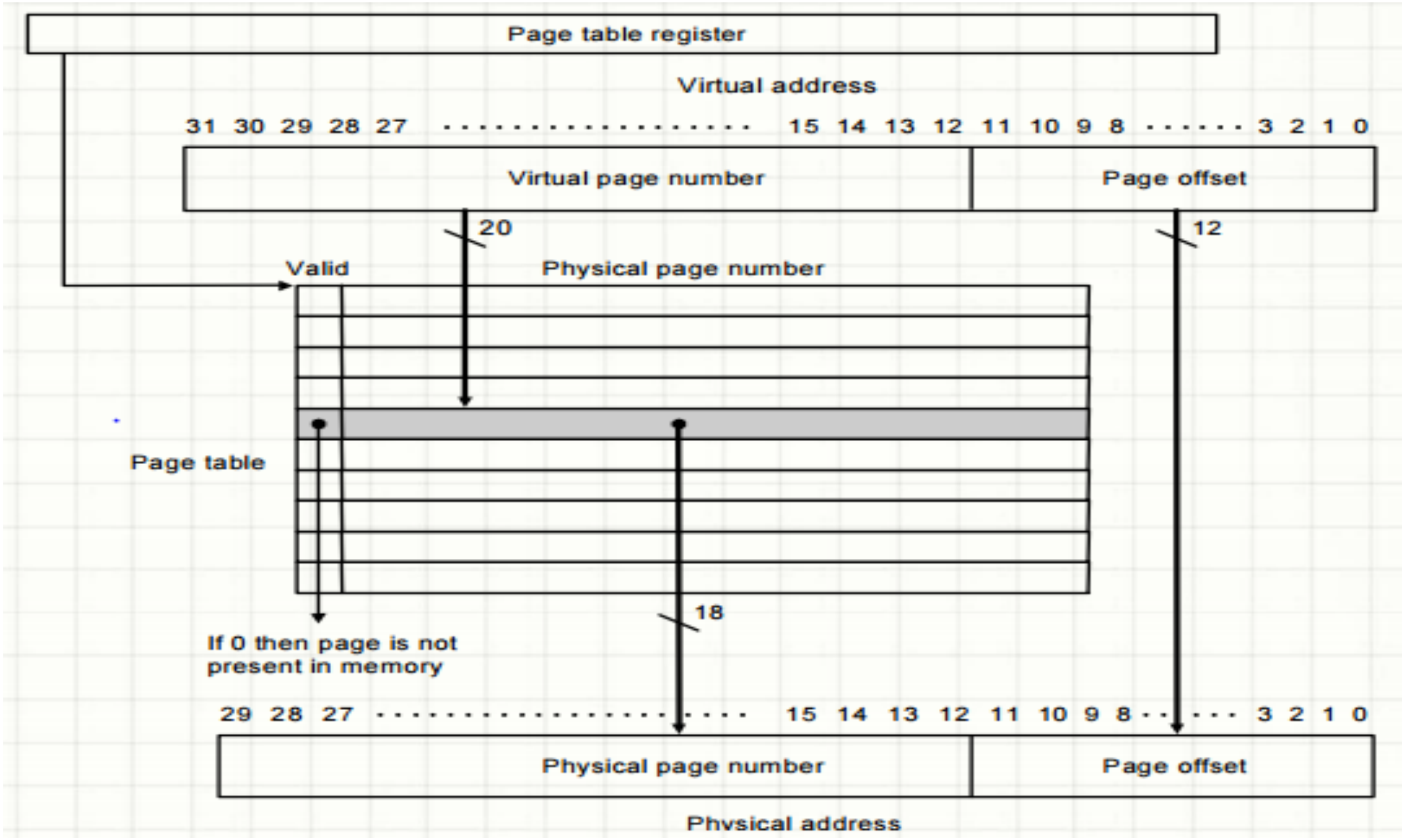
Page Table: Address Translation

- The concept of virtual memory has mainly been introduced to separate the addressing space of the processor and the actual size of the physical memory



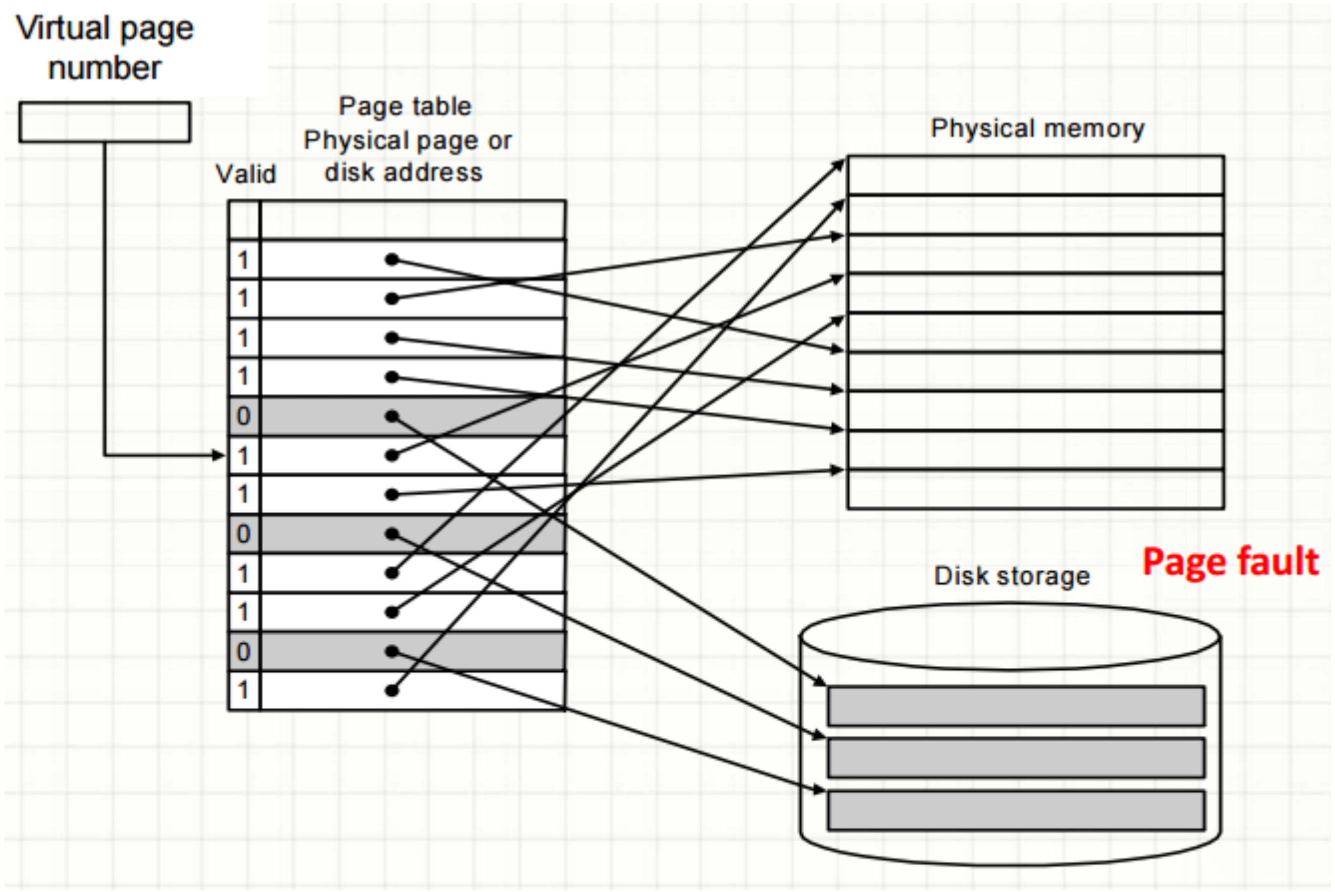
Page Table: Address Translation

- In principle, the Page Table should contain an entry for each virtual page of the process
- Page Table maps virtual page numbers (VPNs) to physical page numbers (PPNs)
- The VPN is used as index of the Page Table
- Each row in Page Table could be called Page Entry (PE)



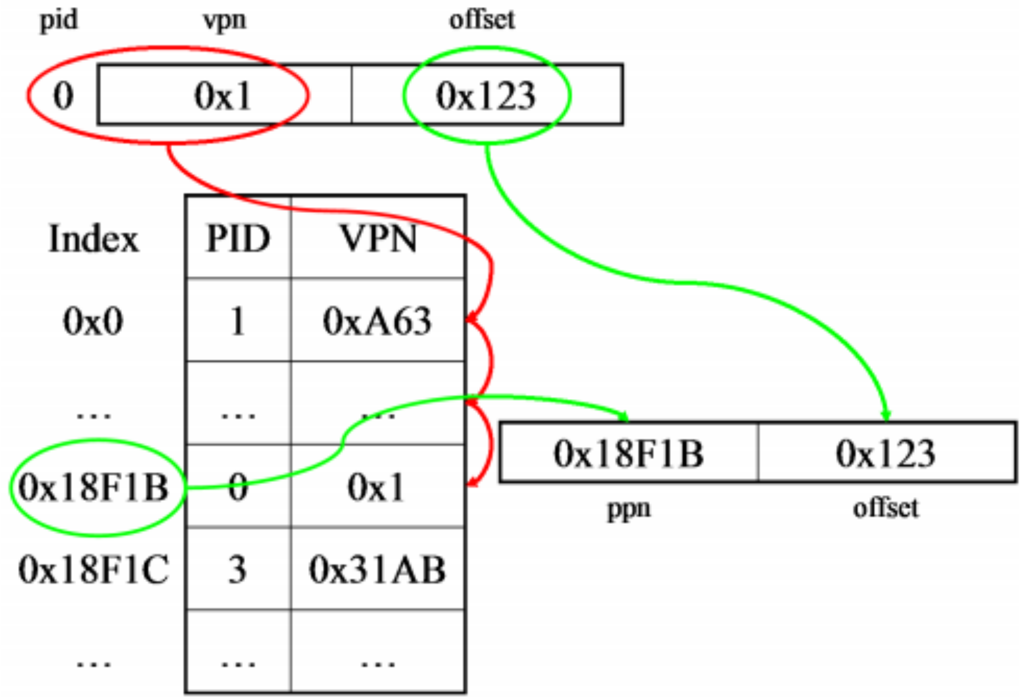
Page Table: Valid bit

- The Page Table requires a validity bit to distinguish between pages in memory and pages in disk storage
  - **Page Fault** happens when Valid bit in dedicated PE is 0 → need to update mapping address in PE by moving page from Disk to Memory



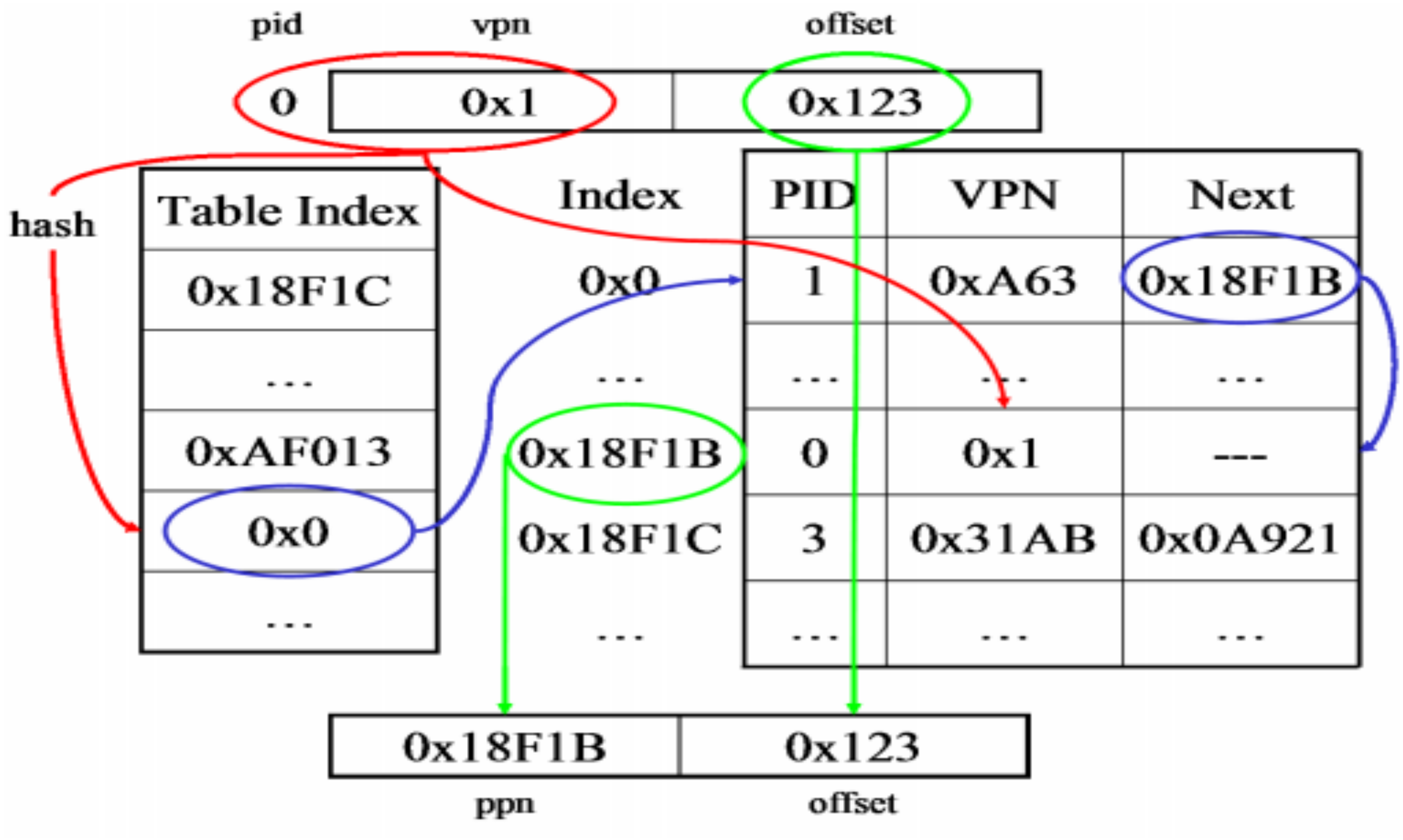
Page Table: Various Types

- A bare bone Page Table is a data structure which contains VPN, PPN and possibly some address space information
- Each Page Table for each Process could contains million of entries → RAM would be overloaded
- **Inverted Page Table**
  - Operating system uses a global pages table for all the processes
  - Each entry of Inverted Page Table has <PID, VPN> → Inverted Page Table consumes less space (small size) but the lookup time (translation time) could be very large



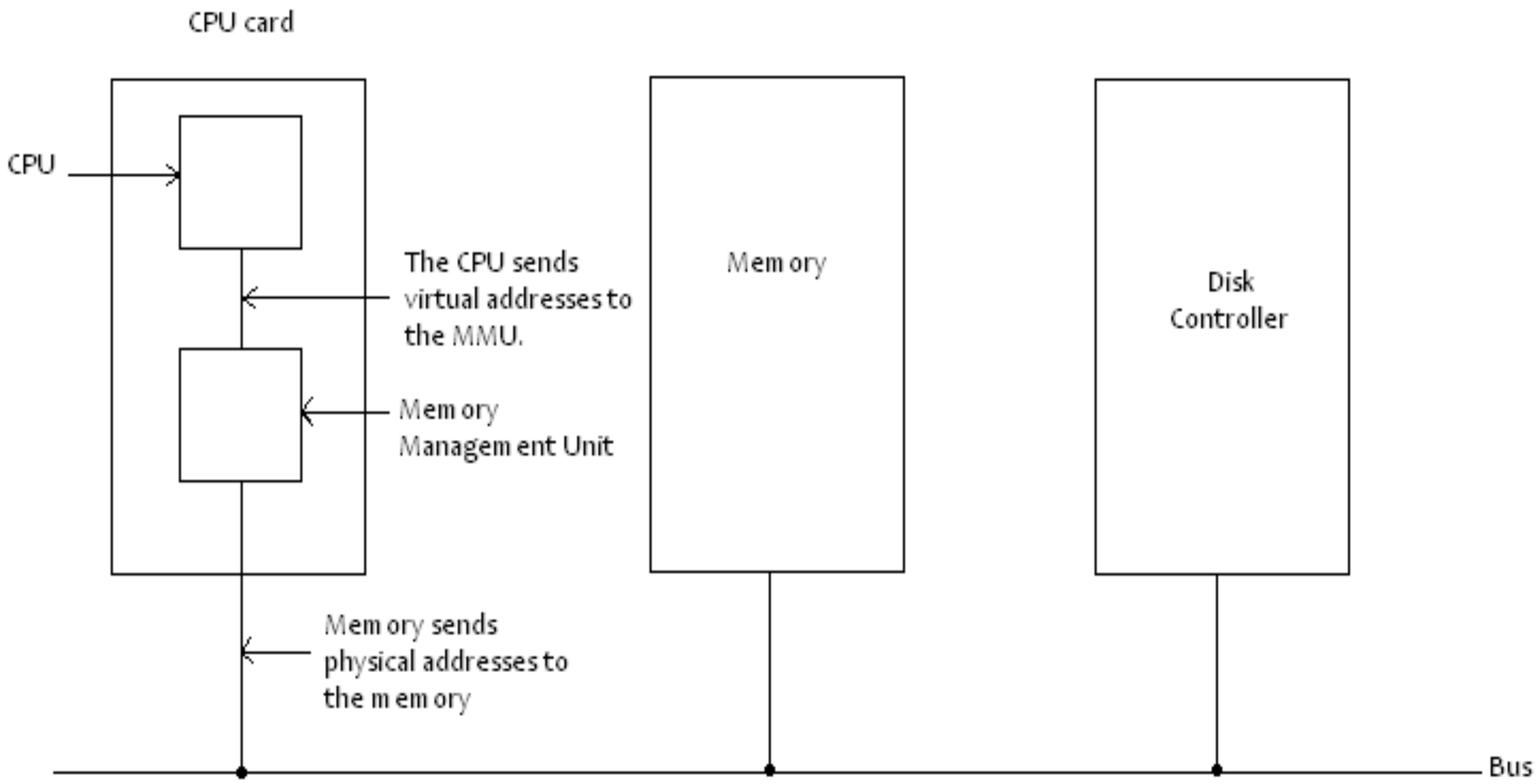
Page Table: Various Types

- **Hashed Inverted Page Table**
  - Add an extra level before the actual page table, called a hash anchor table
  - Collisions in mapping is possible, the page table must be chaining. The chain can be represented as a sequence of page table entries, with each entry pointing to the next entry in the chain



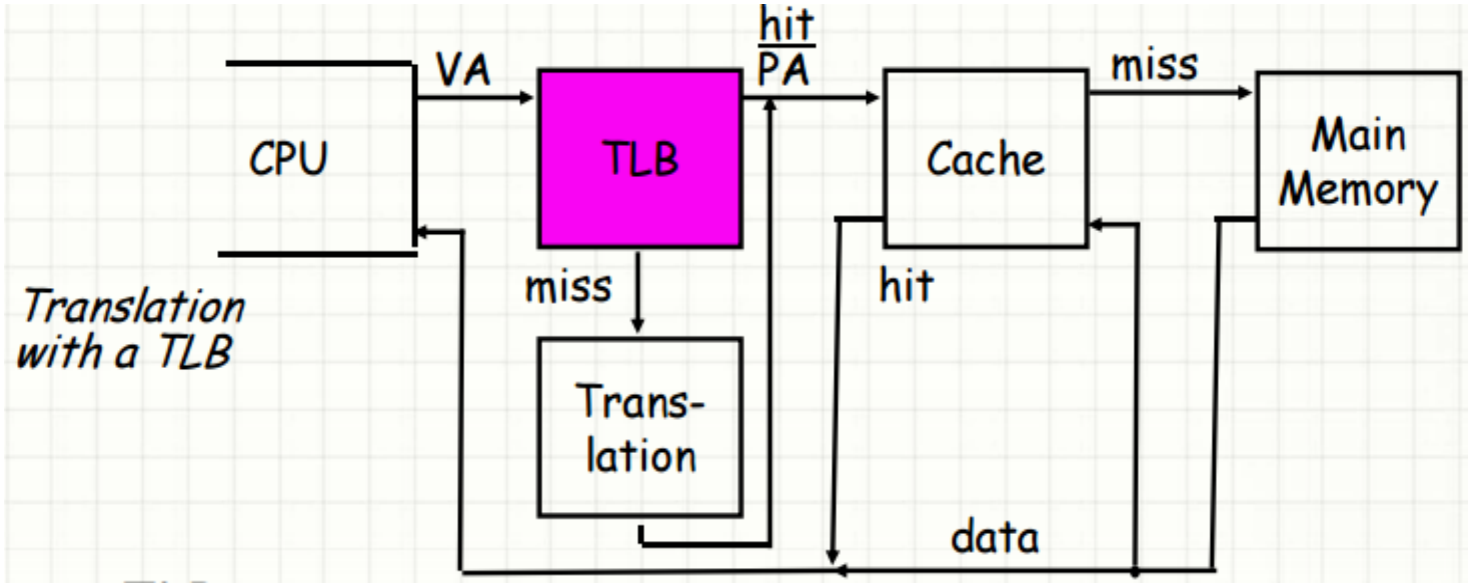
Memory Management Unit - MMU

- Basic idea: a hardware called **Memory Management Unit - MMU** makes the translation from VPNs to PPNs

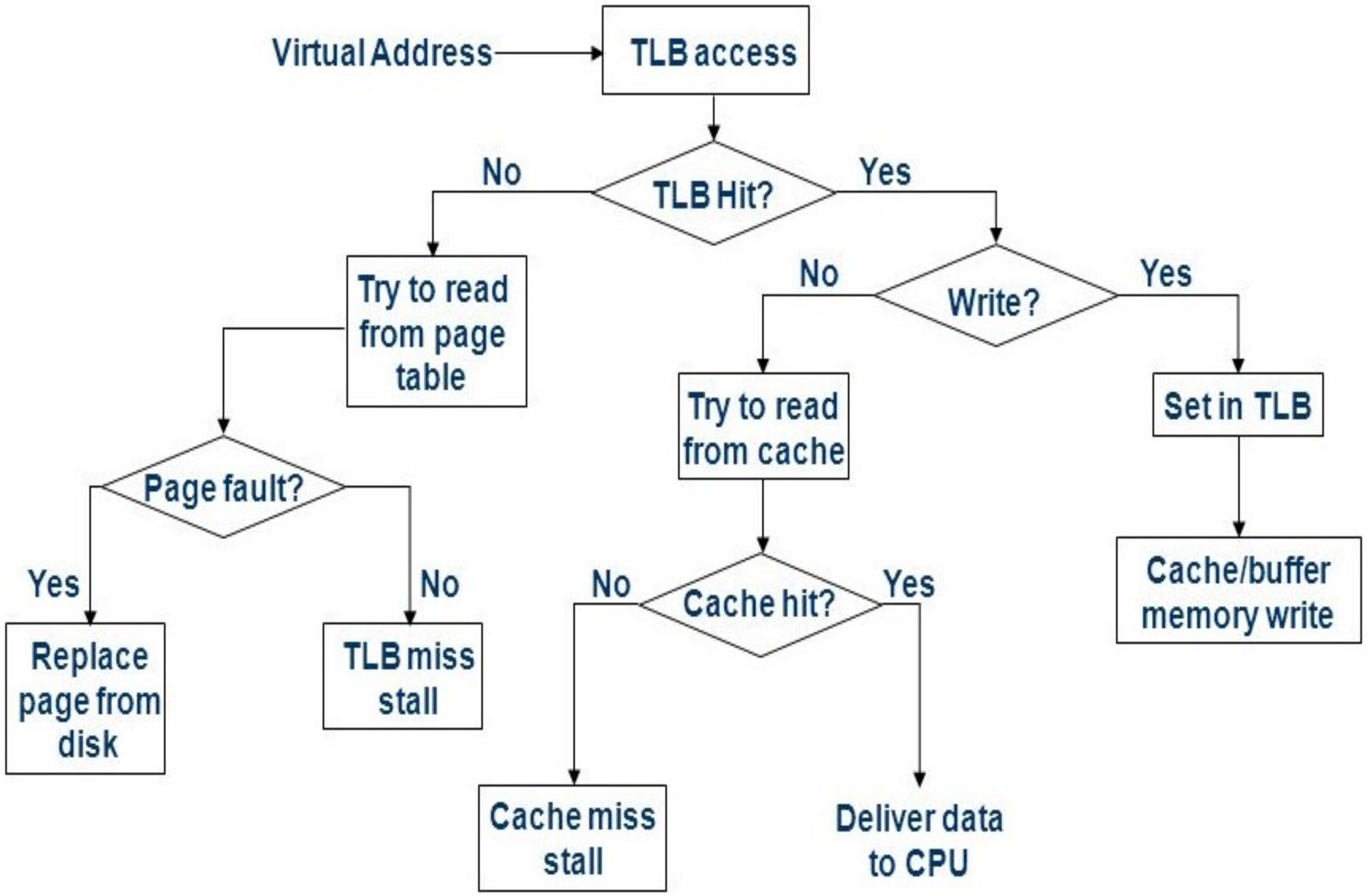

CPU card
CPU
The CPU sends virtual addresses to the MMU.
Memory Management Unit
Memory sends physical addresses to the memory
Memory
Disk Controller
Bus

Translation Lookaside Buffer - TLB

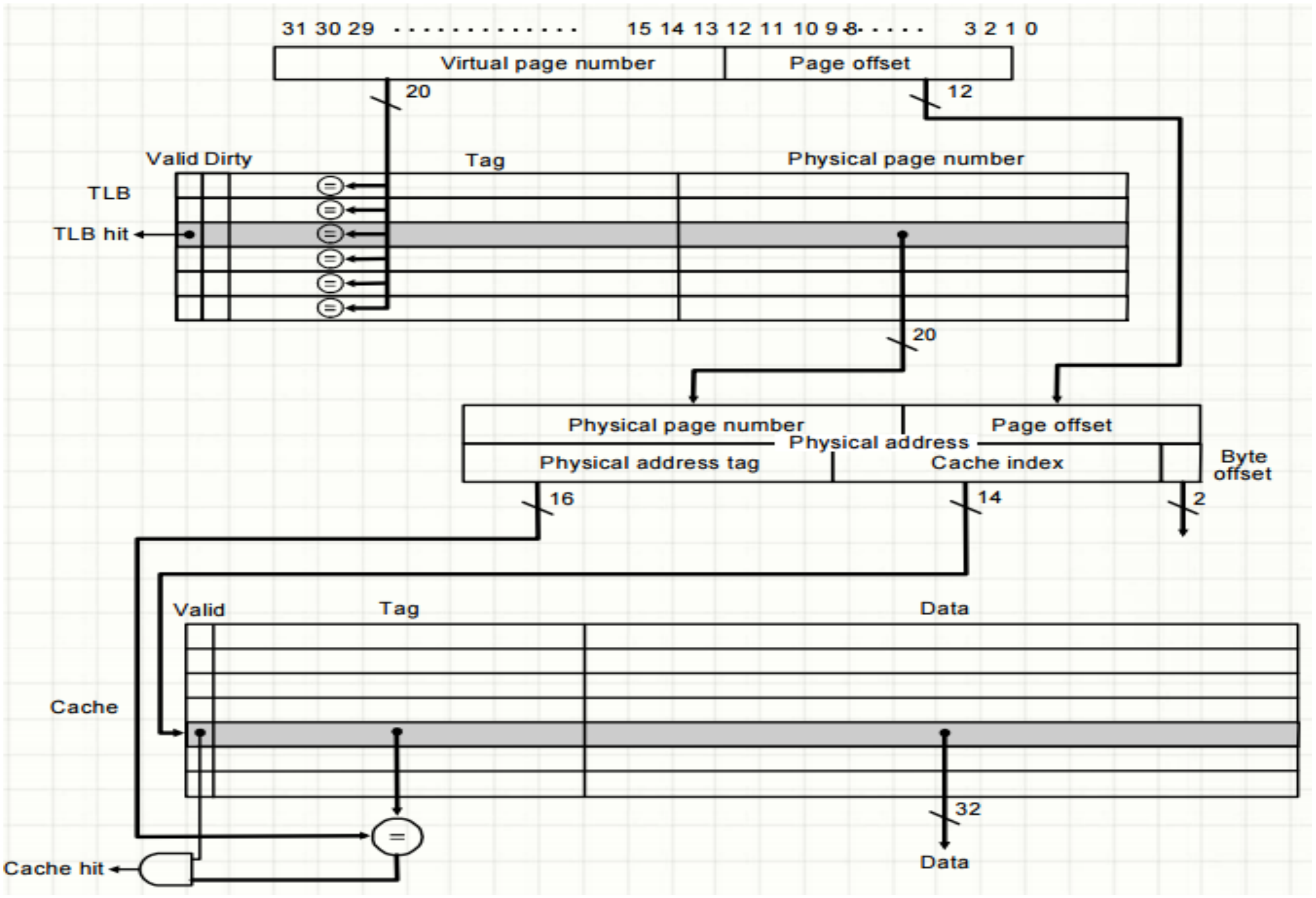
- To reduce the time needed to translate from virtual address to physical address, the MMU integrates **Translation Lookaside Buffer - TLB**
- The TLB works as a cache to *store the recent translations of virtual memory to physical memory*
- A TLB may reside between the CPU and the CPU cache, between the CPU cache and main memory or between different levels of the multilevel cache
- In a Harvard architecture, we could have Instruction TLB and Data TLB
- Similar to cache organization, *TLB may have multiple levels with different speed and size*. *Typically, TLB is fully associative*



The big picture of TLB operation



Challenges: TLB Hit vs. Cache access



Question?