Memory Organization (part 2)

Chapter 8

Memory Organization (part 2)

- ★ Virtual Memory
- ★ Paging Organization
- ★ Address Translation
- ★ Page Tables
  - ○ Translation Look-Aside Buffers
- ★ Virtual Memory and Cache
  - ○ Overlapped Cache & TLB Access

Virtual Memory

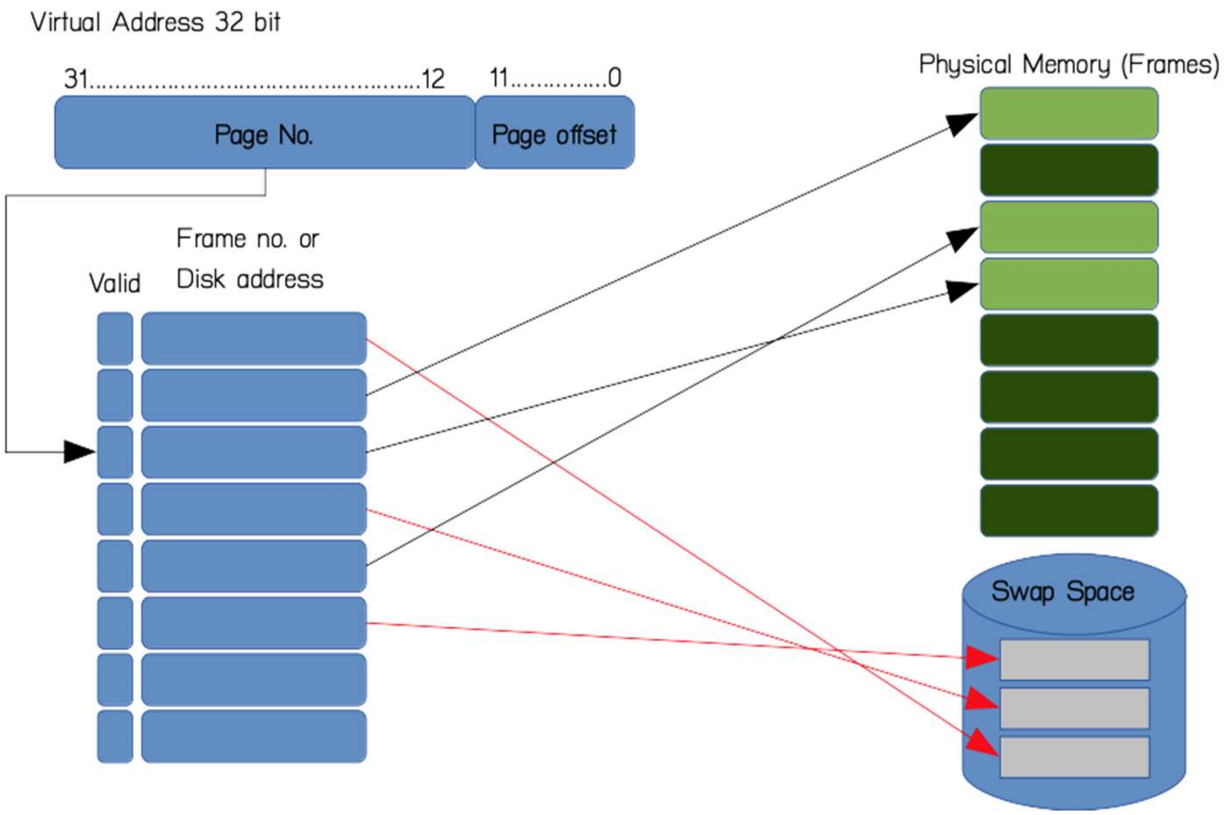
- ★ Cost (Run software that requires 4GB on 2GB hardware.)
  - ○ Illusion of more memory (borrowing from secondary storage)
  - ○ Demand loading and swap
- ★ Program Relocatable
  - ○ Different process has its own address space.
  - ○ Address 0x100 of Prog. A is not the same as address 0x100 of Prog. B.
- ★ Protection (Security)
  - ○ Prog. A may not access memory space of Prog. B directly.

Paging Organization

- ★ A **virtual address** is the address that visible to software.
- ★ A virtual address is translated into a **physical address** by looking up from **page table**.
- ★ When physical address space is smaller than the amount of virtual address space, a **swap space** is borrowed from secondary storage (disk).
- ★ A virtual block is called a **page**. (Similar to block in cache)
- ★ A physical block is called a **page frame**.
- ★ A miss is called **page faults**.

Paging Organization (ctd.)

- ★ Physical Memory can act as a cache for secondary storage (file/disk).
- ★ Missing item is loaded on fault only (demand load policy)
- ★ Replacement policy is handled by operating system.

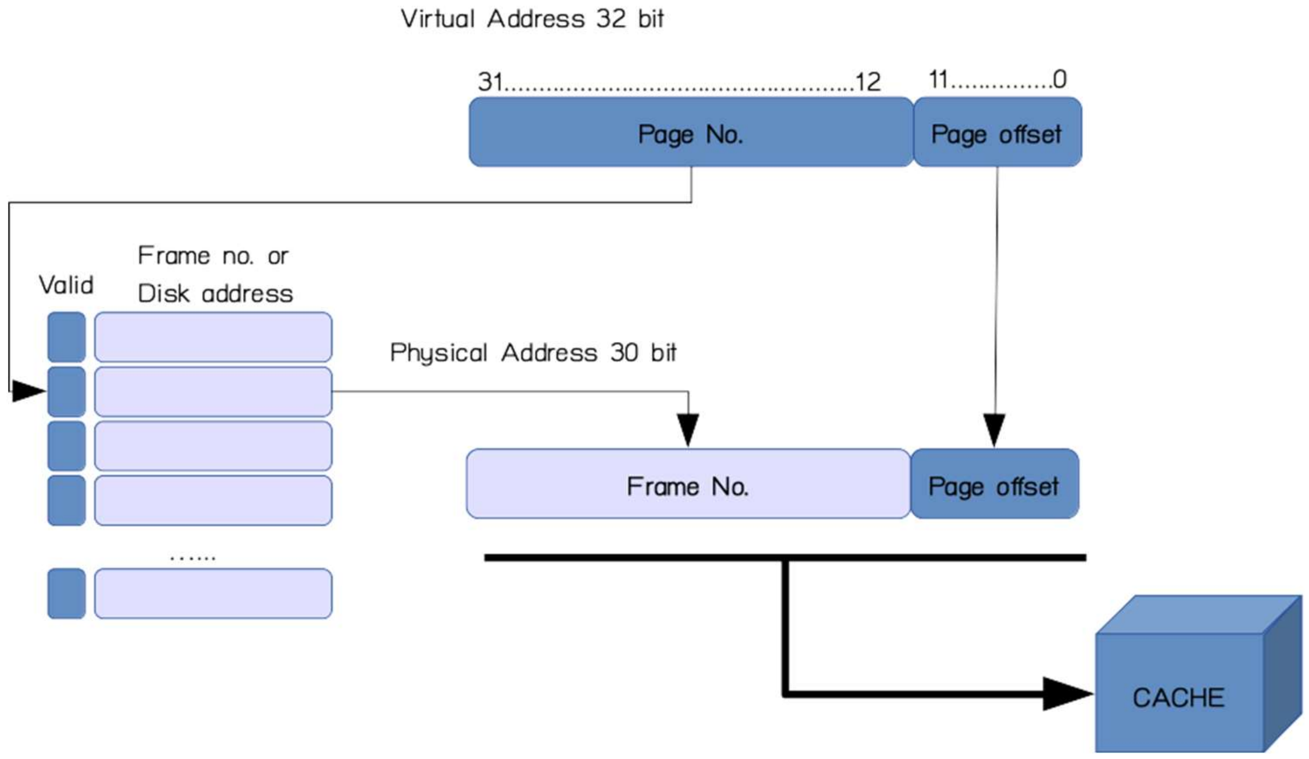


Pages: Virtual Memory Blocks

- ★ Virtual and physical address space partitioned into blocks of equal size
- ★ Virtual -- pages, Physical -- page frames
- ★ Page faults -- data is not in memory, retrieve from disk
  - ○ Hugh miss penalty. A page is usually large (eg. 4KB).
  - ○ A write through is too expensive. Only uses **write back** (page out).
  - ○ A fault is handled by software.
    (Hardware does not know file system or swap management.)

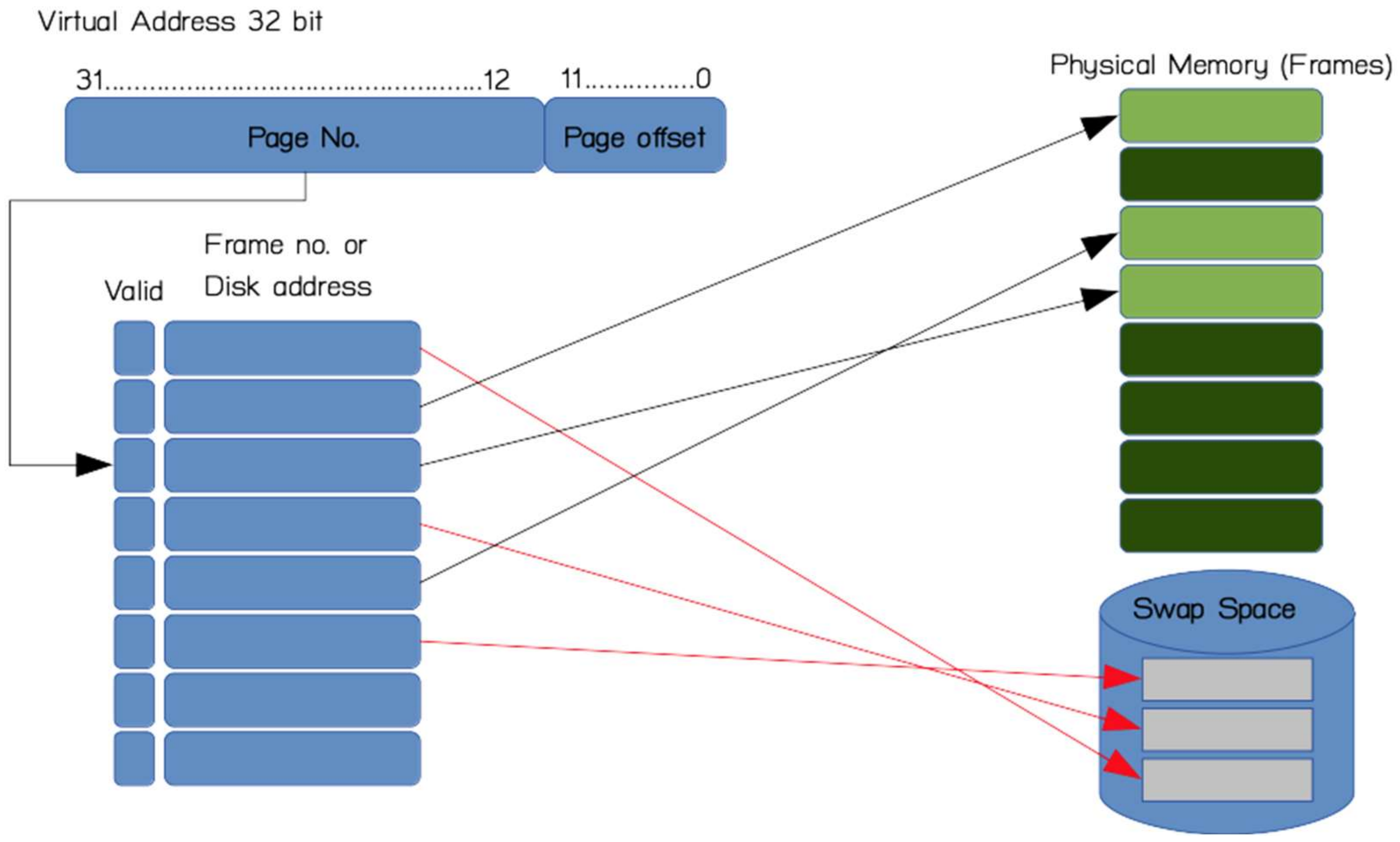
Address Translation

★ Virtual address space can be larger than Physical address space. (Run 4GB software on 2GB hardware)



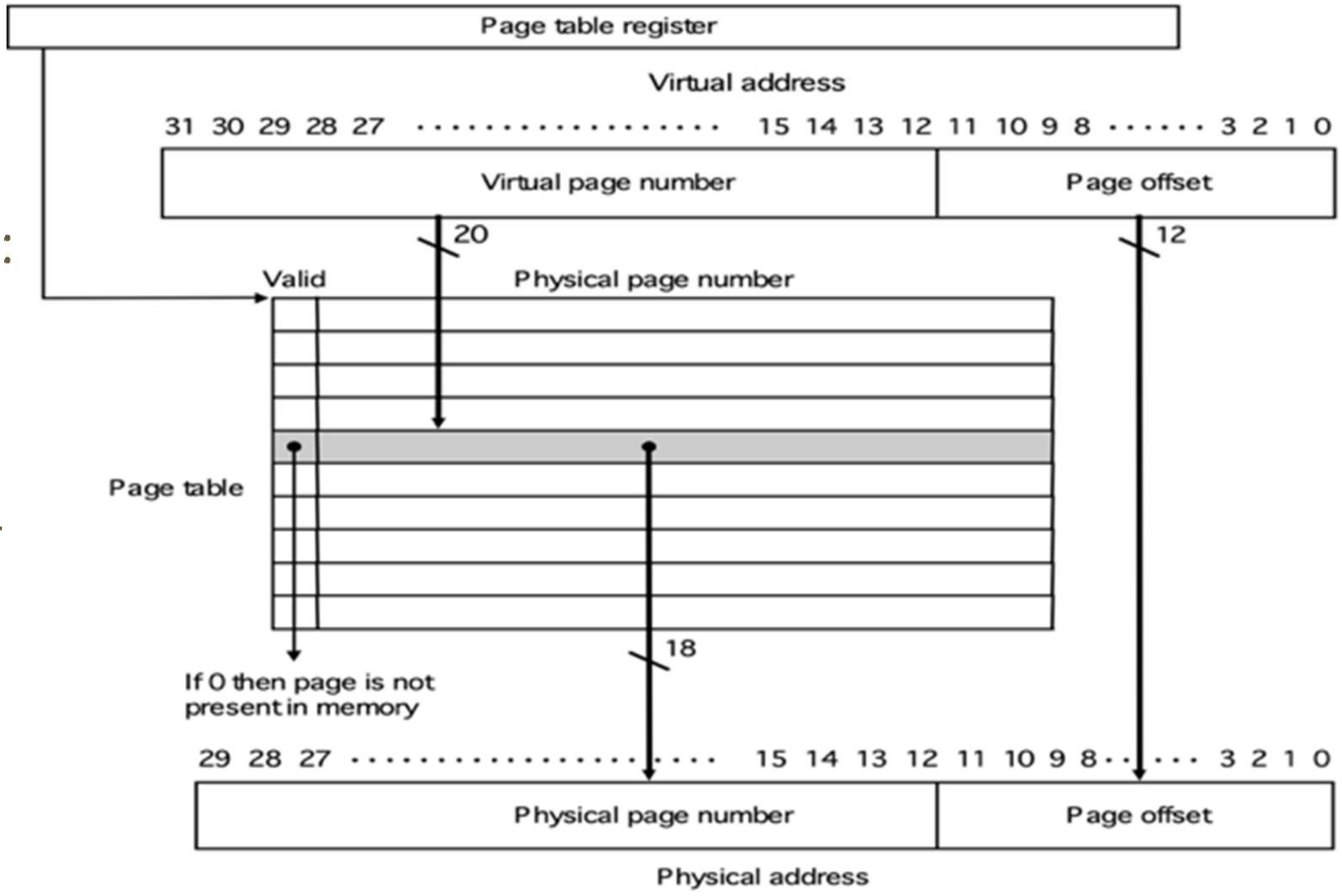
Page Tables

★ In reality, page table also include access permission (eg. readonly)



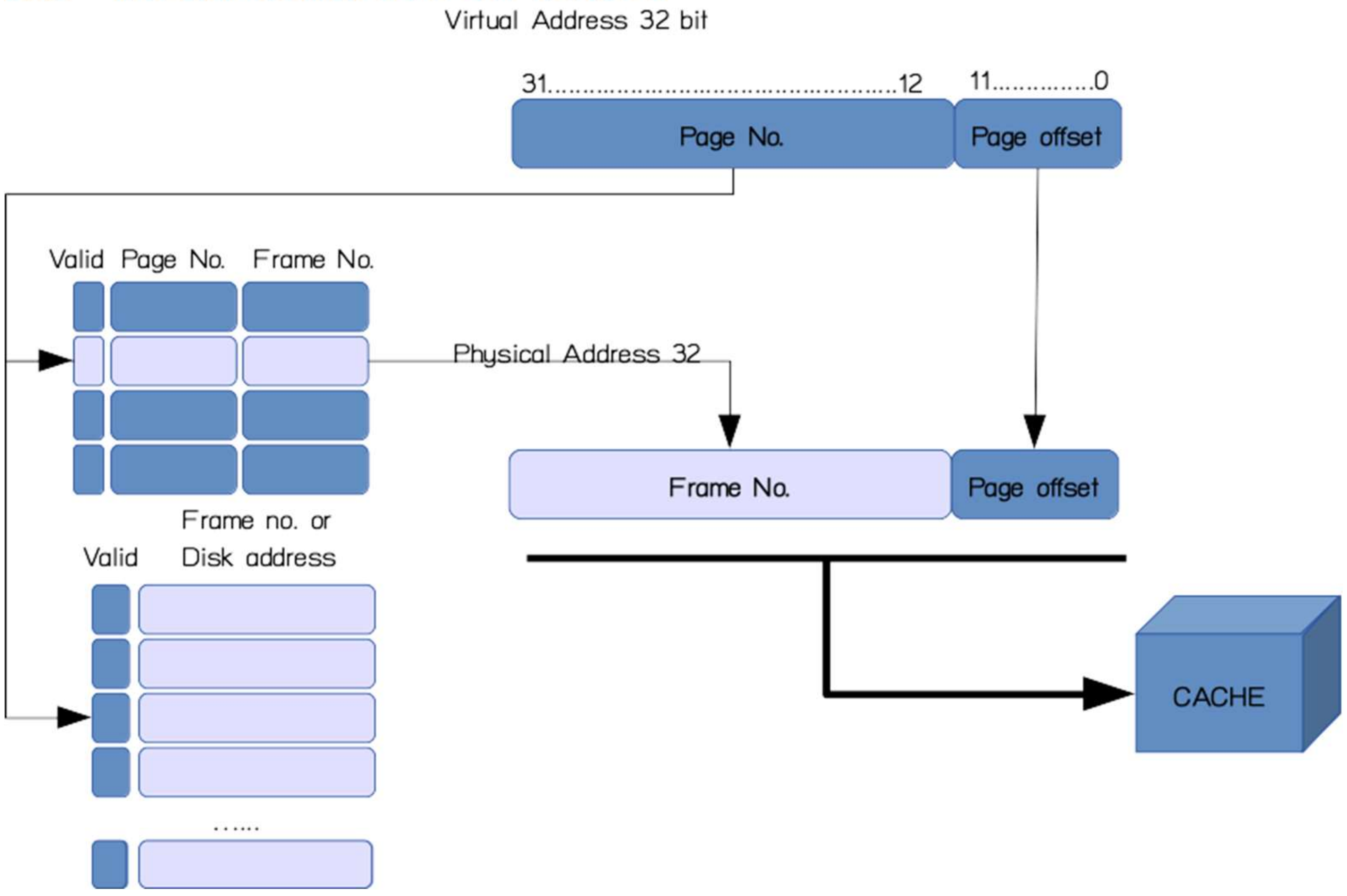
Page Table (ctd.)

- ★ From this organization, what is the size of :
  - ○ Page
  - ○ Physical memory
  - ○ A page table (4 bytes x 2^{20})
- ★ A page table is per process



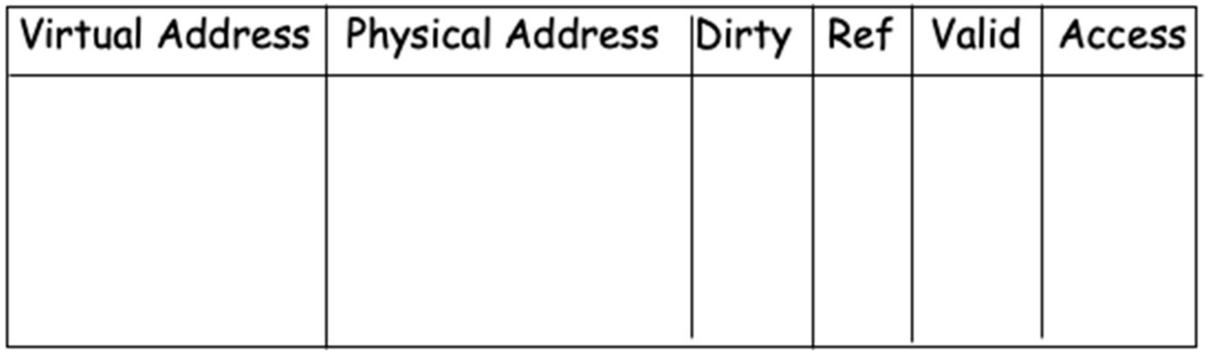
Make Address Translation fast

- ★ A cache for address translations: translation lookaside buffer



TLB (ctd.)

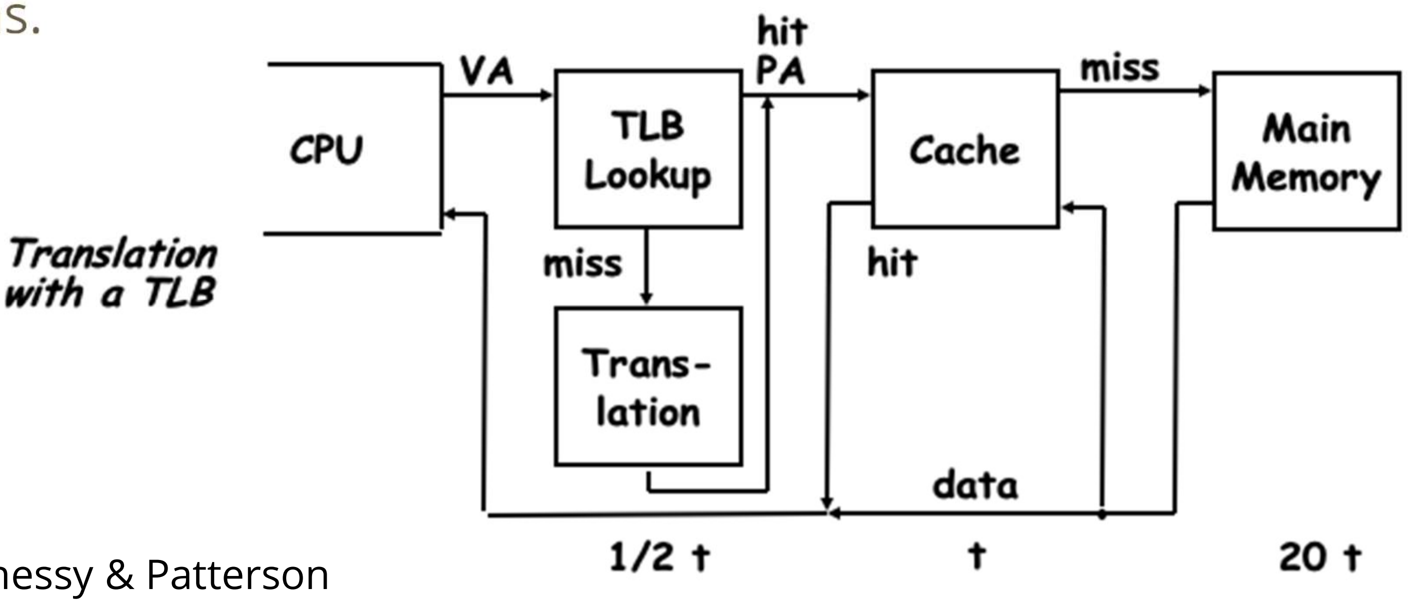
- ★ A way to speed up translation is to use a special cache of recently used page table entries -- this has many names, but the most frequently used is Translation Lookaside Buffer or TLB
- ★ TLB access time comparable to cache access time (much less than main memory access time)



Virtual Address	Physical Address	Dirty	Ref	Valid	Access

Translation Look-Aside Buffers

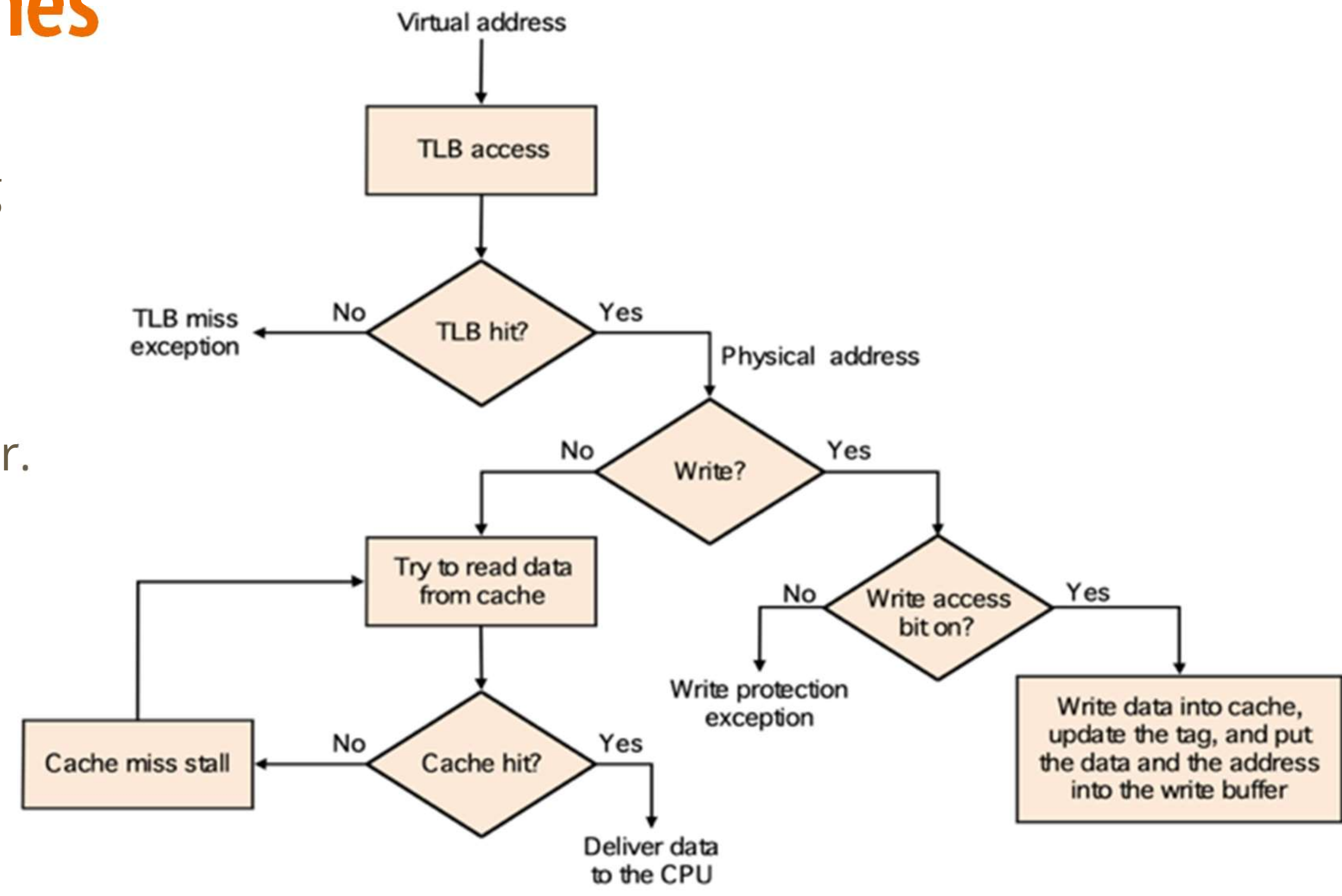
★ TLBs are usually small, typically not more than 128 - 256 entries even on high-end machines. This permits **<u>fully associative</u>** lookup on these machines. Most mid-range machines use **<u>small n-way set associative</u>** organizations.



Picture from Hennessy & Patterson

TLB & Caches

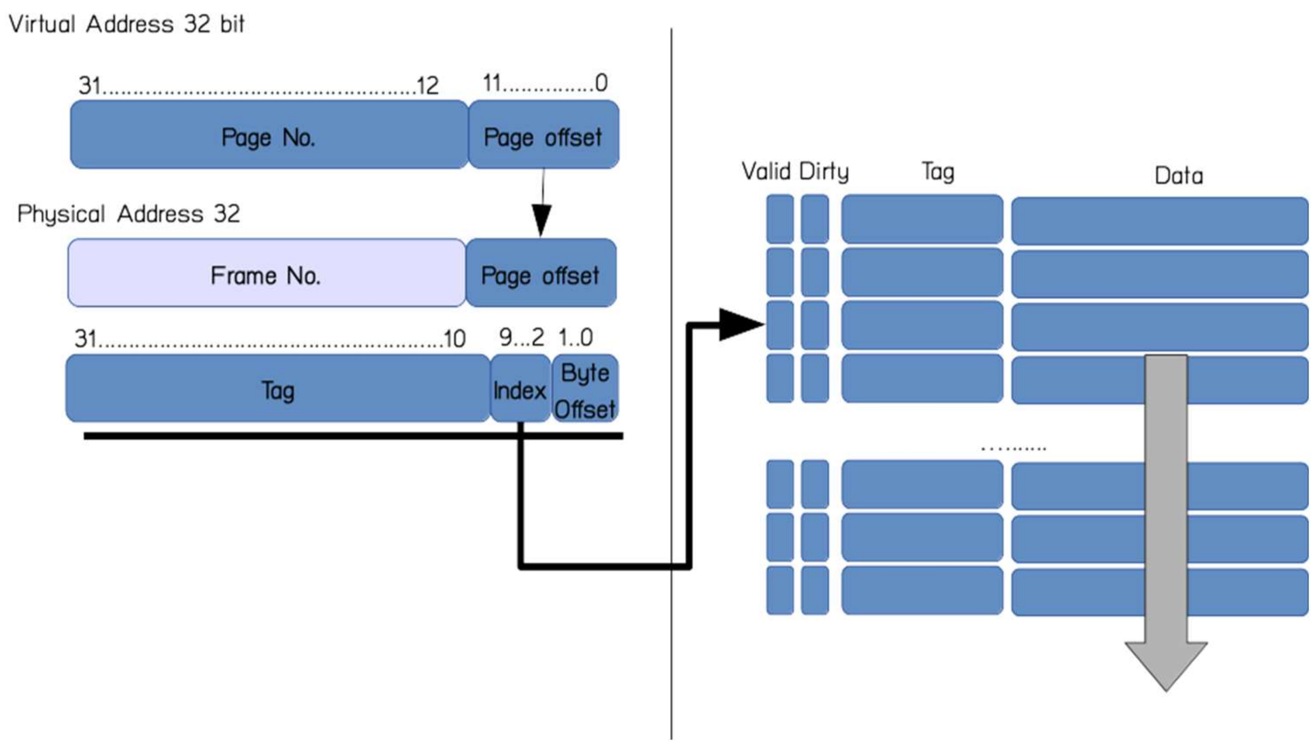
- ★ Steps in accessing data (taken from Intel x86)
- ★ This is slow.
- ★ Let’s make it faster.



Reduce Translation Time

- ★ Machines with TLBs go one step further to reduce # cycles/cache access
- ★ They overlap the cache access with the TLB access
- ★ Works because high order bits of the VA are used to look in the TLB while low order bits are used as index into cache

Overlapped Cache & TLB Access



- ★ If we can parallel (overlap) the cache look up (index) and translation from virtual address to physical address at the same time.
- ★ Possible when **index** is **not changed** in address translation

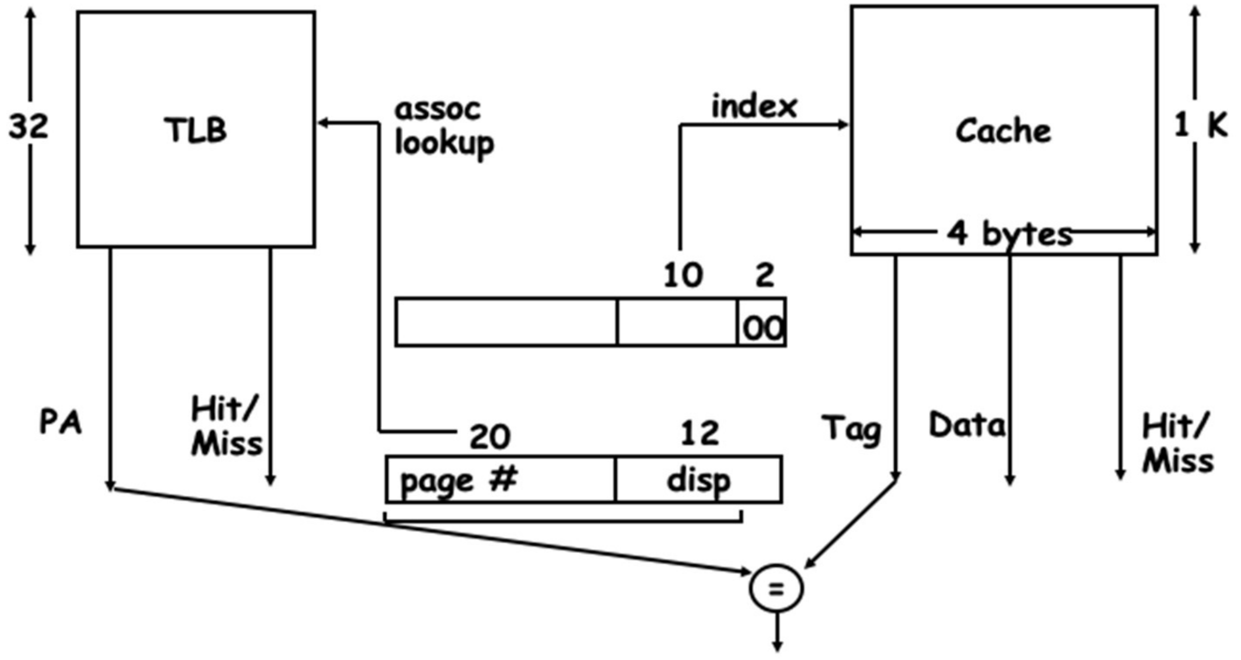
Overlapped Cache & TLB Access (ctd.)

IF cache hit AND (cache tag = PA) then deliver data to CPU

ELSE IF [cache miss OR (cache tag != PA)] and TLB hit

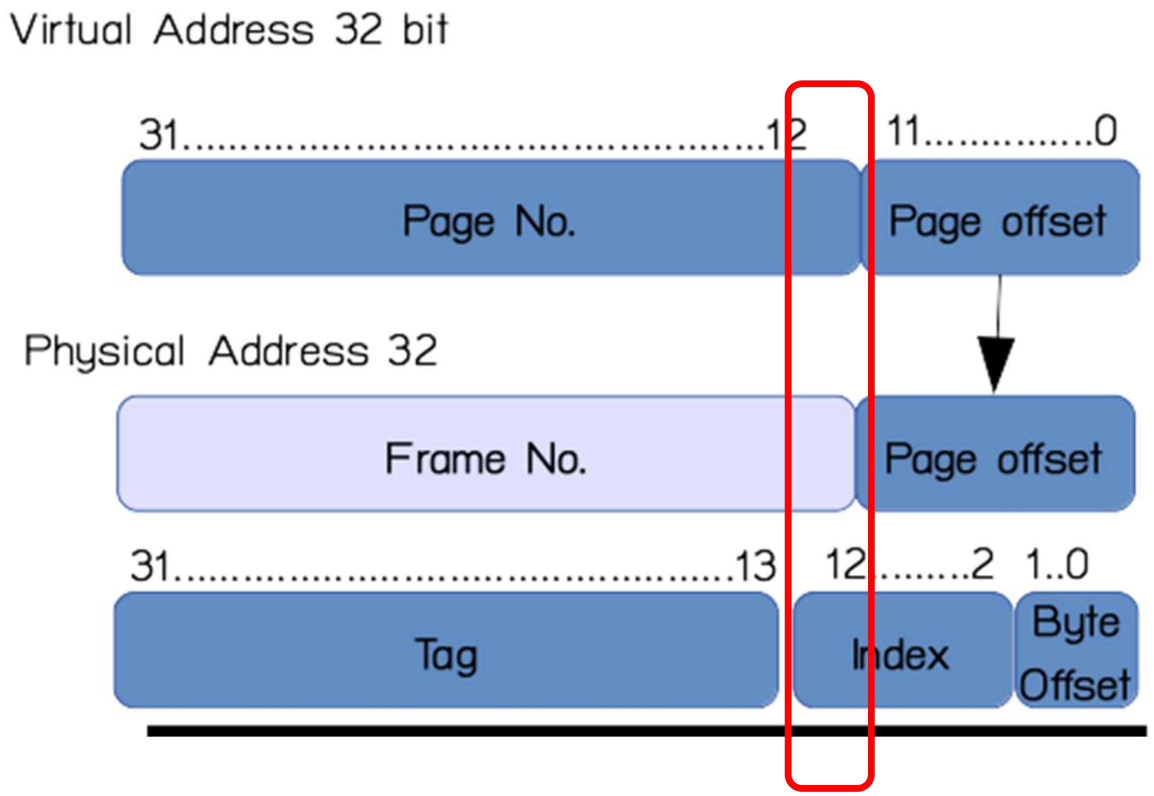
THEN access memory with the PA from the TLB

ELSE do standard VA translation



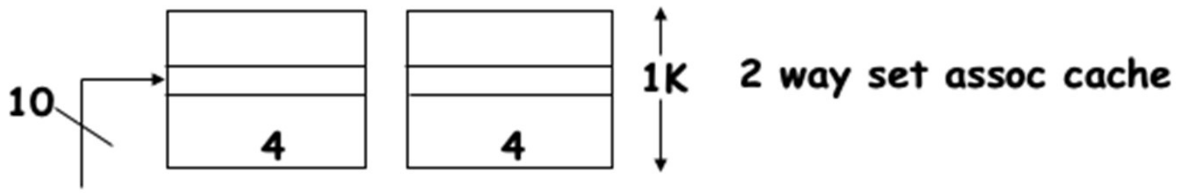
When overlapped access is not possible

- ★ Larger cache (e.g. 8KB) in 4KB page.
- ★ 13 bits cache vs. 12 bits page offset

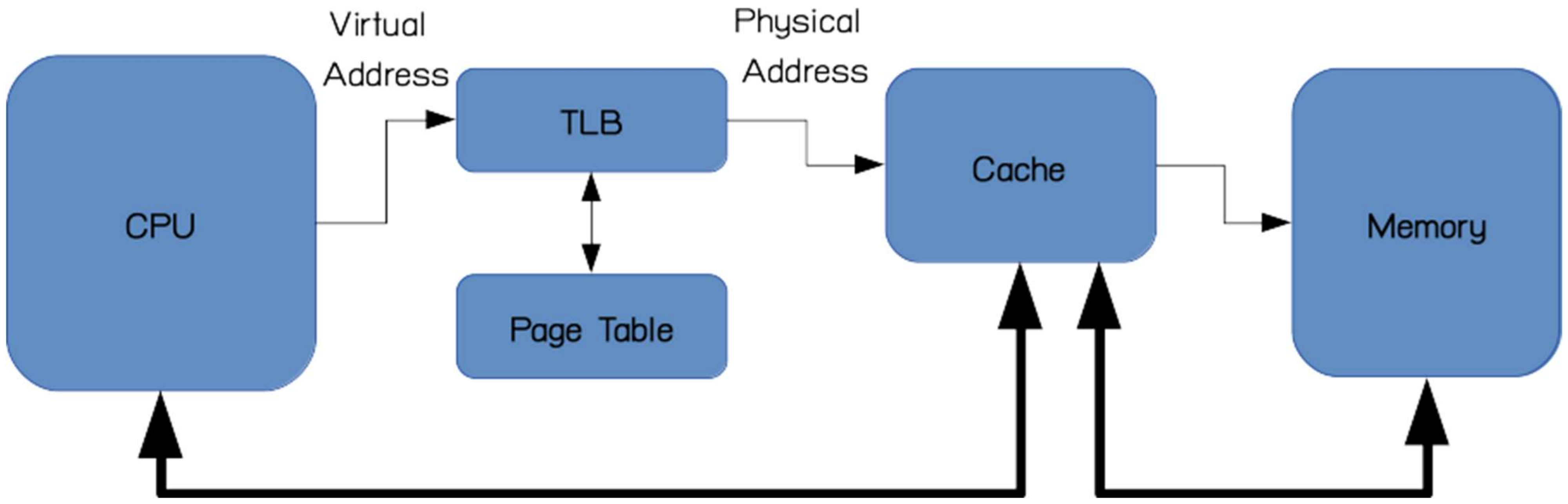


When overlapped access is not possible

- ★ Solutions:
  - go to 8K byte page sizes;
  - go to 2 way set associative cache; or
  - SW guarantee VA[13]=PA[13]

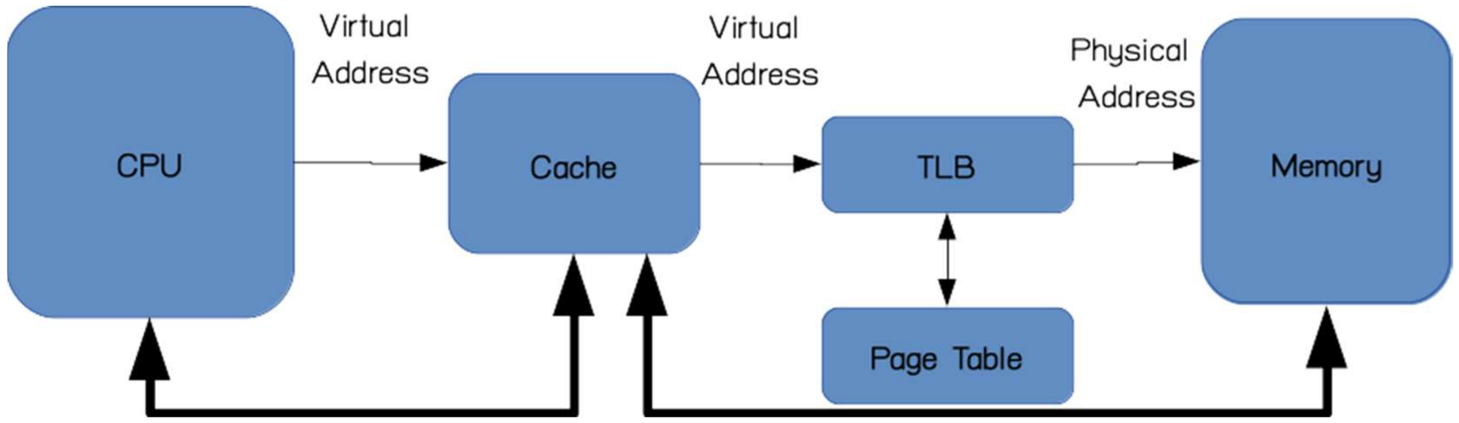


Virtual Memory and Cache



Can we access cache with virtual address?

- ★ Will there be any benefit (or issue) if we move cache before address translation?
- ★ Issues with shared memory (where different virtual addresses from different processes are mapped to same physical address.)



Beyond the scope of this class

- ★ There exists several technologies for performing memory relocation and memory management (such as overlay and segmentation).
- ★ There are also several methods for managing page table. (Inverted Page Table, Multilevel page table)

  With virtualization, several processors now also support virtualized page table.

Summary

★ Caches, TLBs, Virtual Memory all understood by examining how they deal with 4 questions:
  ○ 1) Where can block be placed?
  ○ 2) How is block found?
  ○ 3) What block is replaced on miss?
  ○ 4) How are writes handled?

★ Page tables map virtual address to physical address

★ TLBs are important for fast translation

★ TLB misses are significant in processor performance

https://gabrieletolomei.wordpress.com/miscellanea/operating-systems/virtual-memory-paging-and-swapping/

Exercises

Memory Design

★ With 4KB page, how should an architect design 32KB cache to the advantage of overlapped access? Please provide your analysis.

Process switch

- ★What should happen to the following elements when the OS performs a process switch
    - ★TLB
    - ★Page table
    - ★Cache

End of Chapter 8 (part 2)