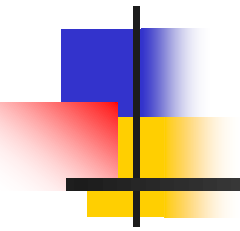
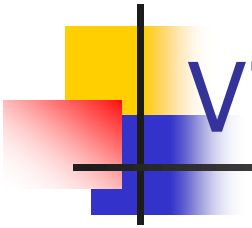


CSE331 - Computer Organization

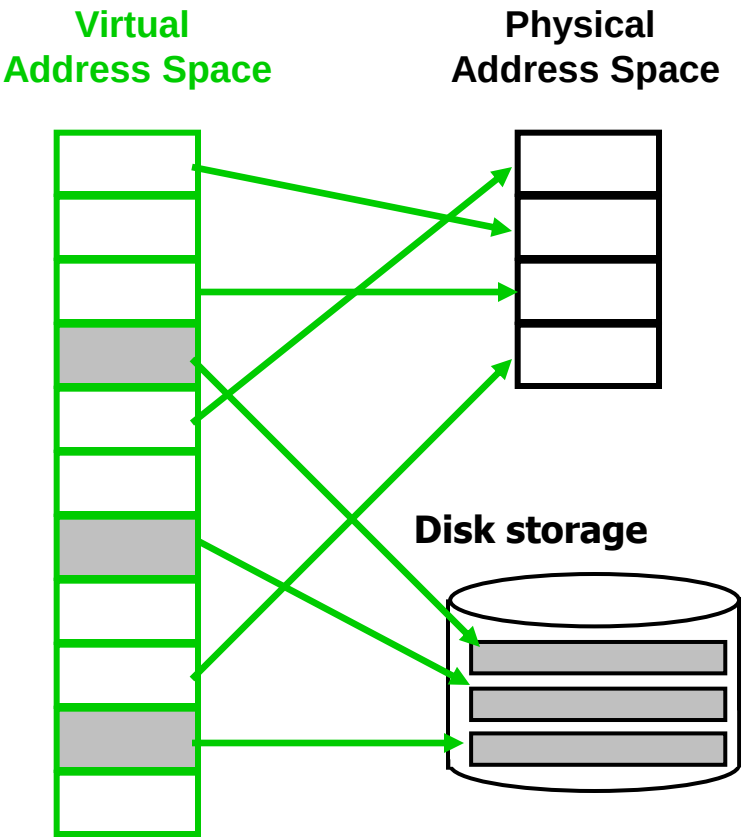
Lecture 14: Virtual Memory



Virtual Memory

- Use main memory as a “cache” for secondary (disk) storage
  - Managed jointly by CPU hardware and the operating system (OS)
- Programs share main memory
  - Each gets a private virtual address space holding its frequently used code and data
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
  - VM “block” is called a page
  - VM translation “miss” is called a page fault

What is virtual memory?



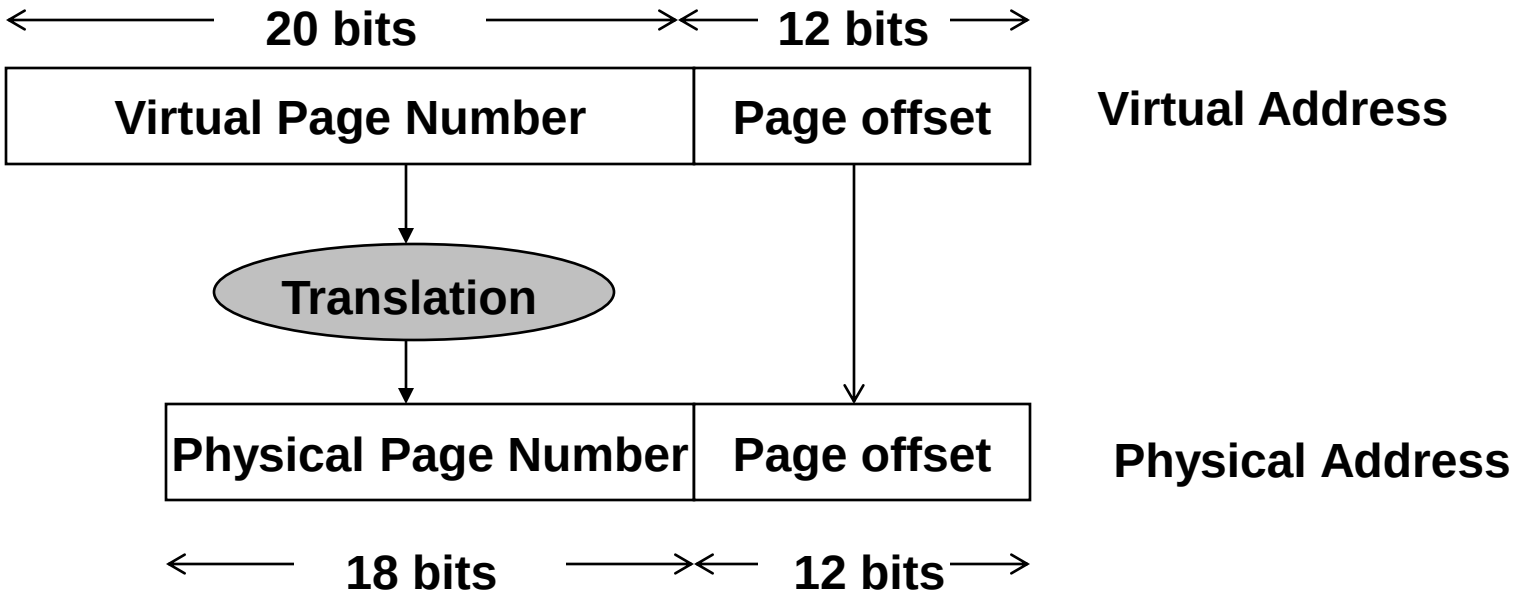
- Virtual memory => treat main memory as a cache for the disk
- Terminology: blocks in this cache are called “Pages”
  - Typical size of a page: 4K — 64K
- Page table maps virtual page numbers to physical frames

Virtual Memory

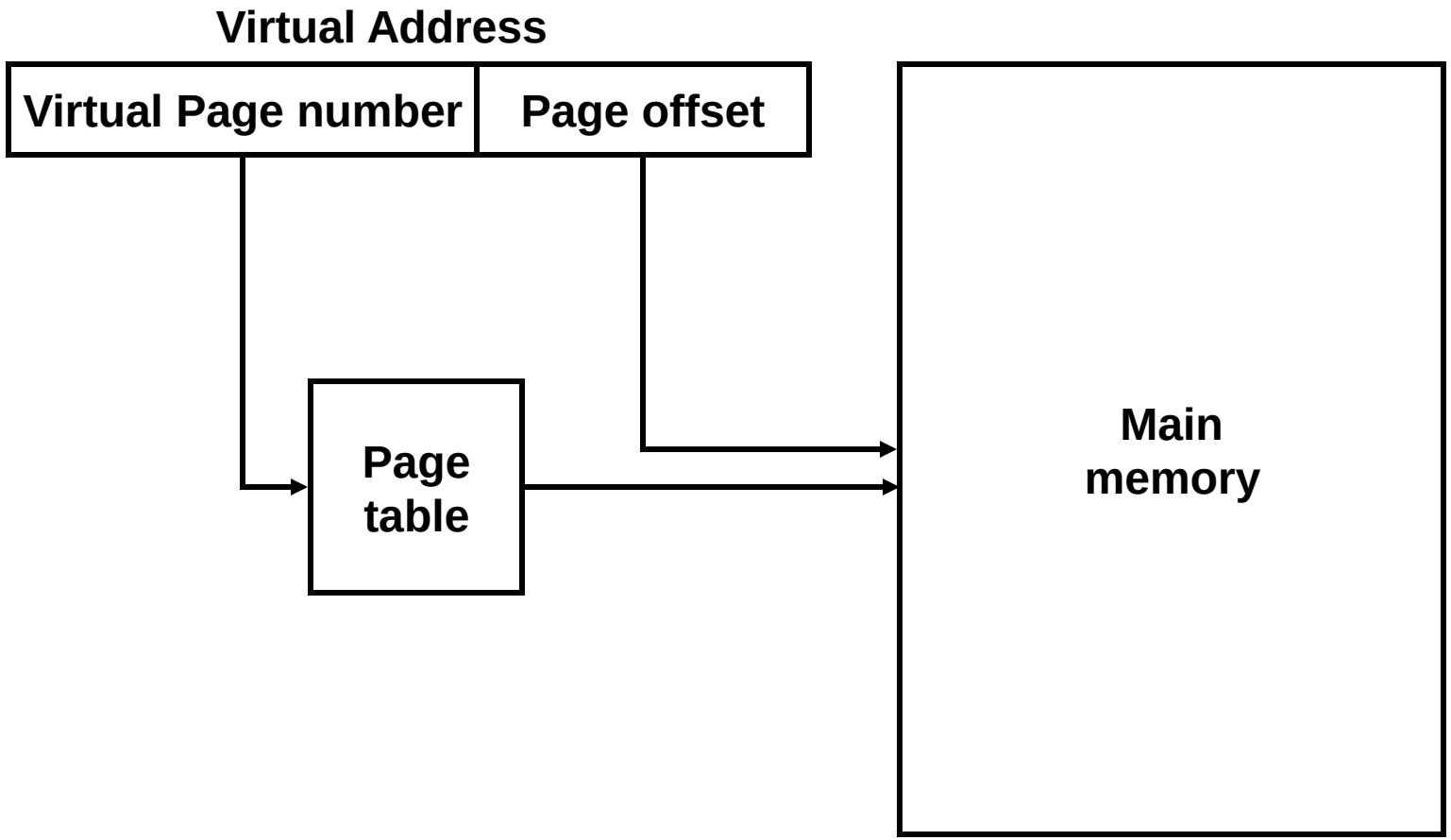
- Virtual memory (VM) allows main memory (DRAM) to act like a cache for secondary storage (magnetic disk).
- VM address translation provides a mapping from the virtual address of the processor to the physical address in main memory and secondary storage.
- VM provides the following benefits
  - Allows multiple programs to share the same physical memory
  - Allows programmers to write code (or compilers to generate code) as though they have a very large amount of main memory
  - Automatically handles bringing in data from disk
- Cache terms vs. VM terms
  - Cache block => page
  - Cache miss => page fault

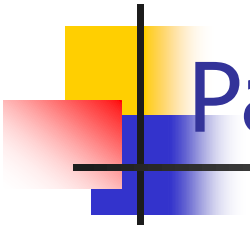
Virtual and Physical Addresses

- A virtual address consists of a virtual page number and a page offset.
- The virtual page number gets translated to a physical page number.
- The page offset is not changed



Address Translation





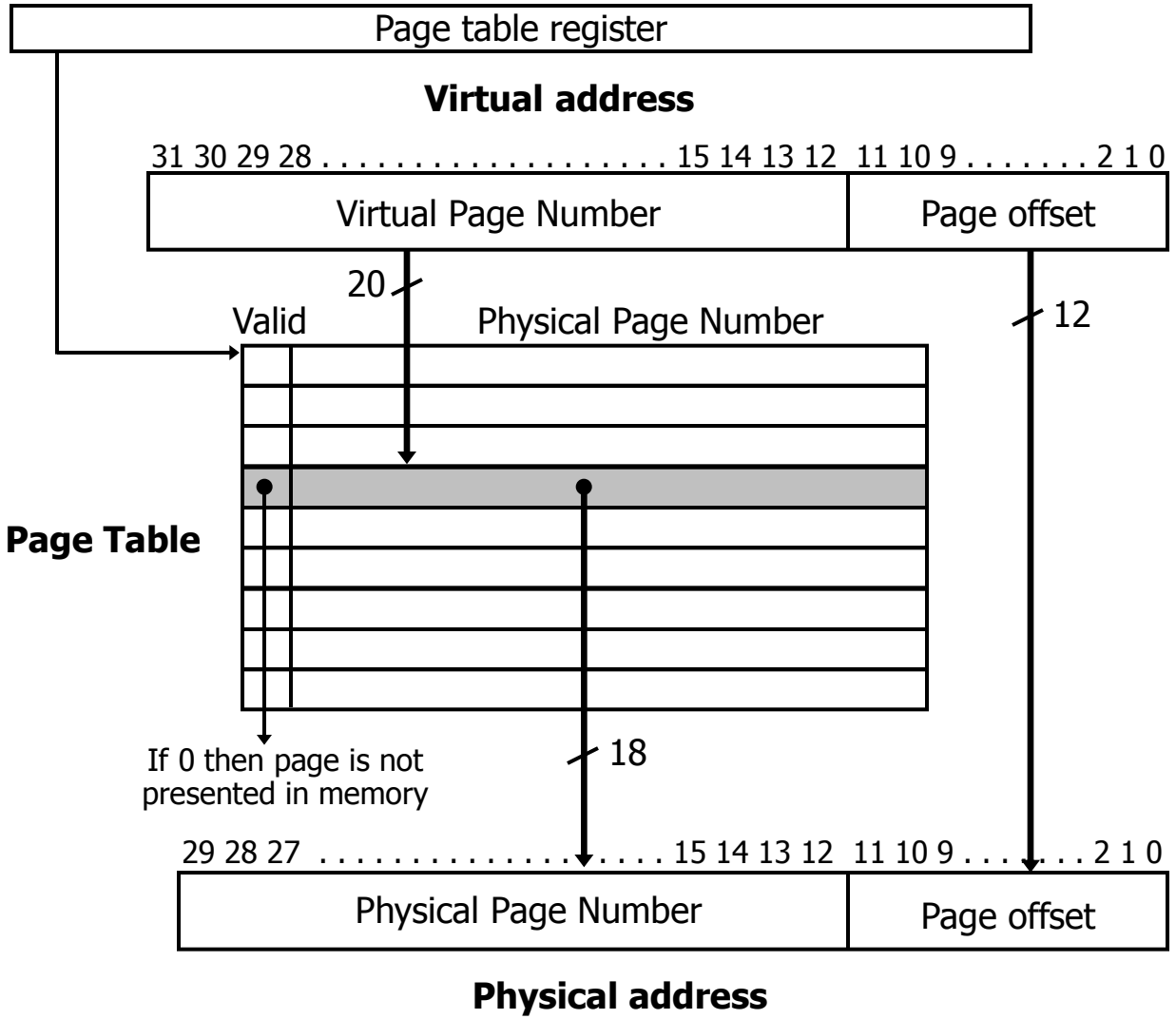
Page Tables

- Stores placement information
  - Array of page table entries, indexed by virtual page number
  - Page table register in CPU points to page table in physical memory
- If page is present in memory
  - PTE stores the physical page number
  - Plus other status bits (referenced, dirty, …)
- If page is not present
  - PTE can refer to location in swap space on disk

Address Translation with Page Tables

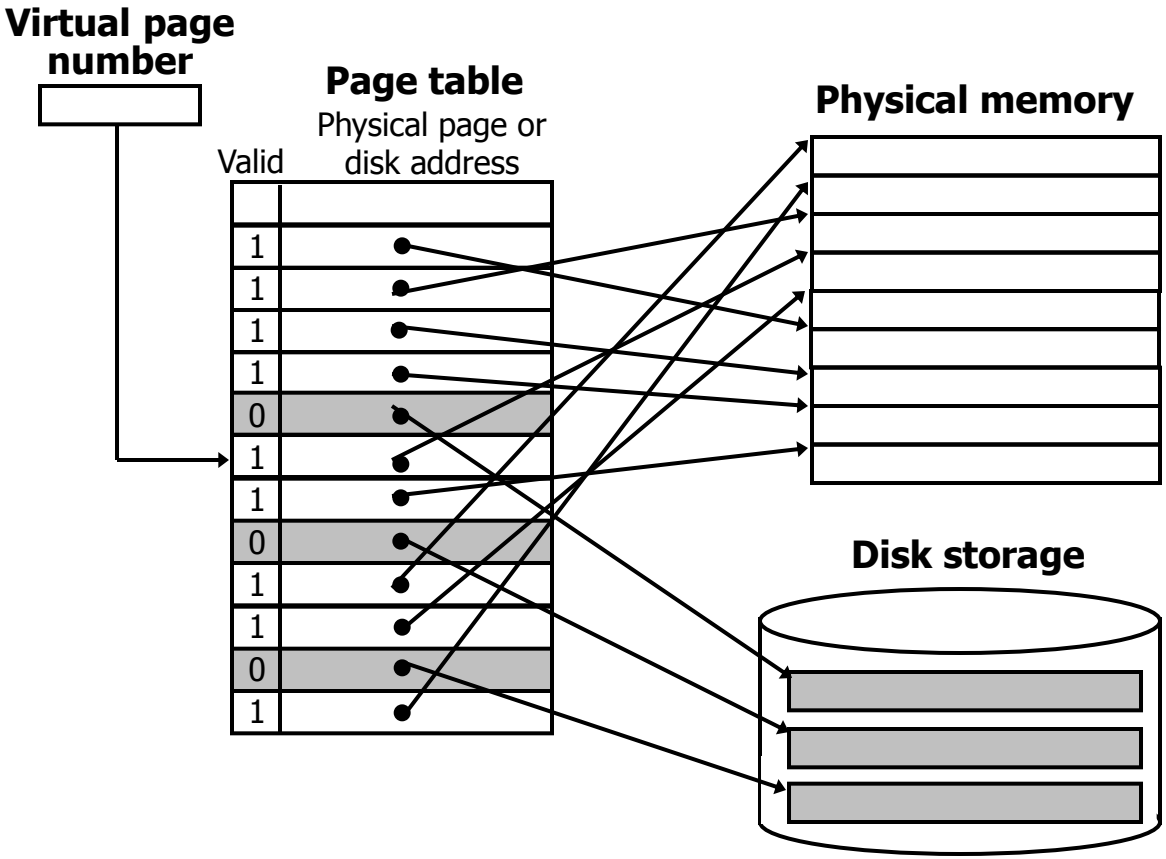
- A page table translates a virtual page number into a physical page number.
- The virtual page number is used as an index into the page table that contains
  - The physical page number
  - A valid bit that indicates if the page is present in main memory
  - A dirty bit to indicate if the page has been written
  - Protection information about the page (read only, read/write, etc.)
- Since page tables contain a mapping for every virtual page, no tags are required.

Page Table Diagram (See Figure 7.21 on page 517)



Accessing Main Memory or Disk (See Figure 7.22 on page 518)

- If the valid bit of the page table is zero, this means that the page is not in main memory.
- In this case, a page fault occurs, and the missing page is read in from disk.

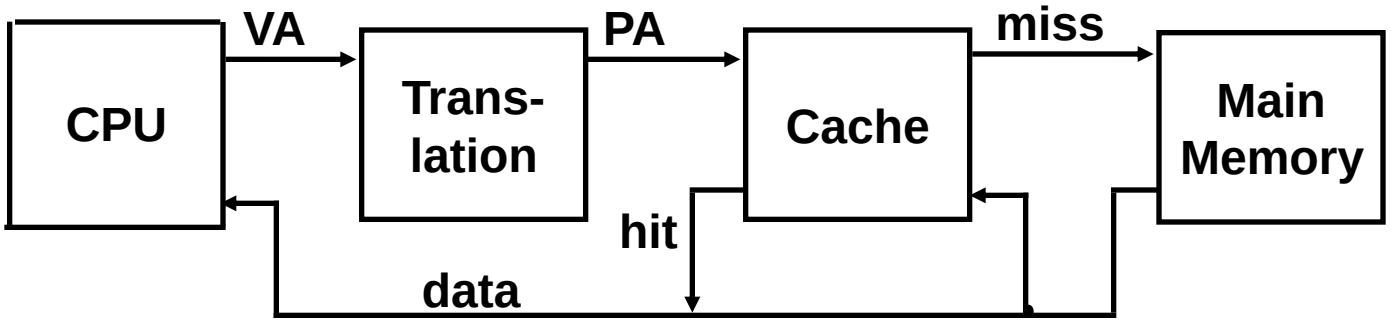


Determining Page Table Size

- Assume
  - 32-bit virtual address
  - 30-bit physical address
  - 4 KB pages => 12 bit page offset
  - Each page table entry is one word (4 bytes)
- How large is the page table?
  - Virtual page number = 32 - 12 = 20 bits
  - Number of entries = number of pages = 2^20
  - Total size = number of entries x bytes/entry

    = 2^20 x 4 = 4 Mbytes
- Each process running needs its own page table
- Since page tables are very large, they are almost always stored in main memory, which makes them slow.

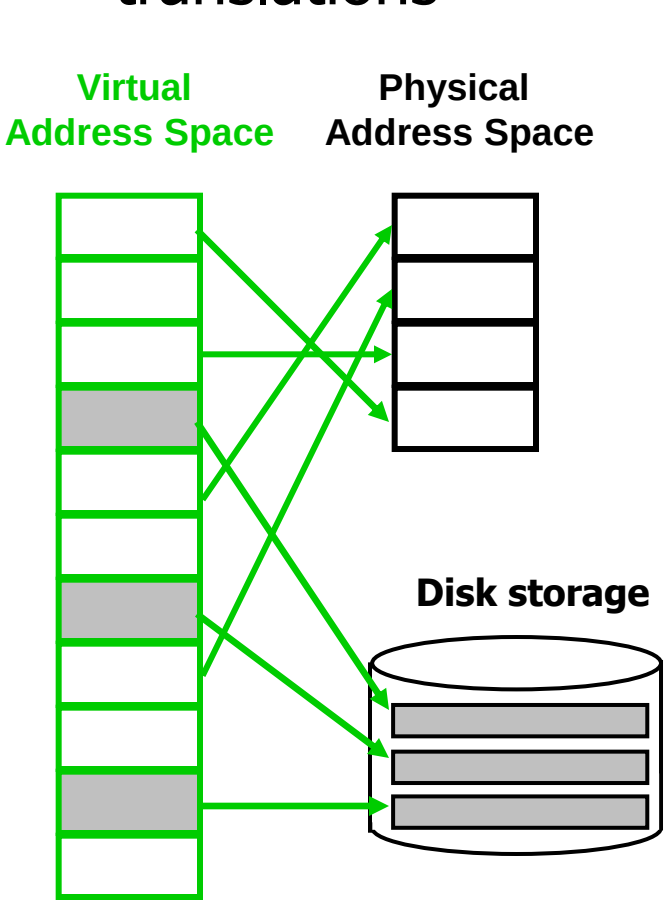
Caching Virtual Addresses

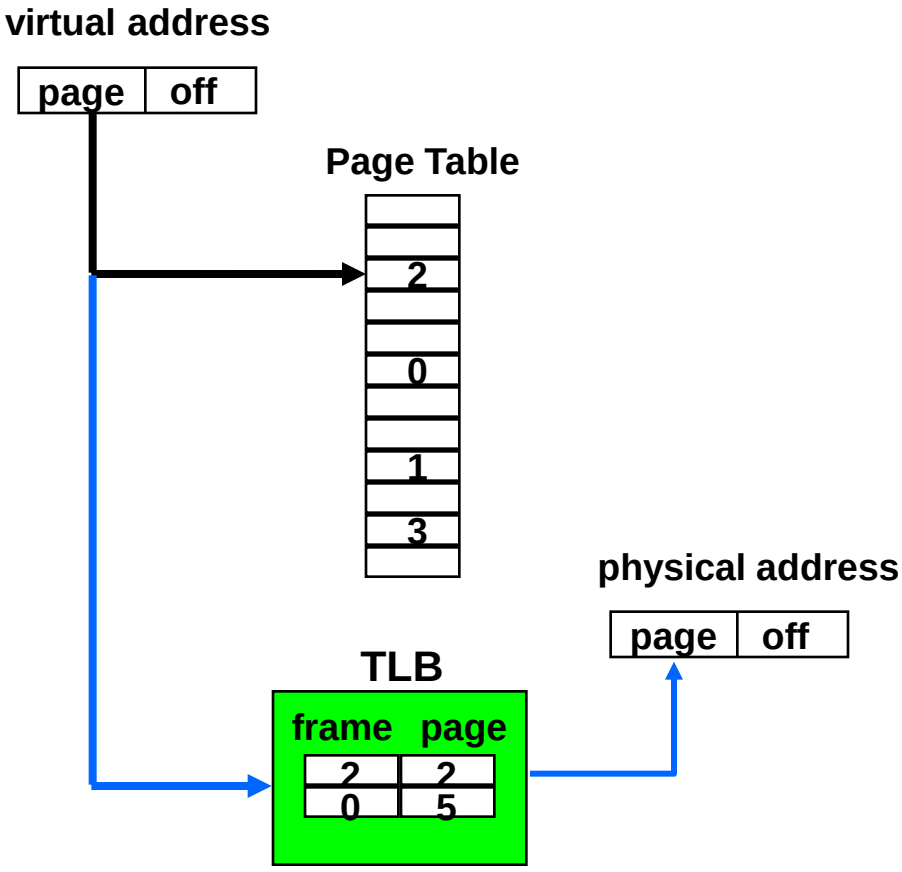


- Virtual memory seems to be really slow:
  - Must access memory on load/store -- even cache hits!
  - Worse, if translation is not completely in memory, may need to go to disk before hitting in cache!
- Solution: Caching!
  - Keep track of most common translations and place them in a "**Translation Lookaside Buffer**" (TLB)

Making address translation practical: TLB

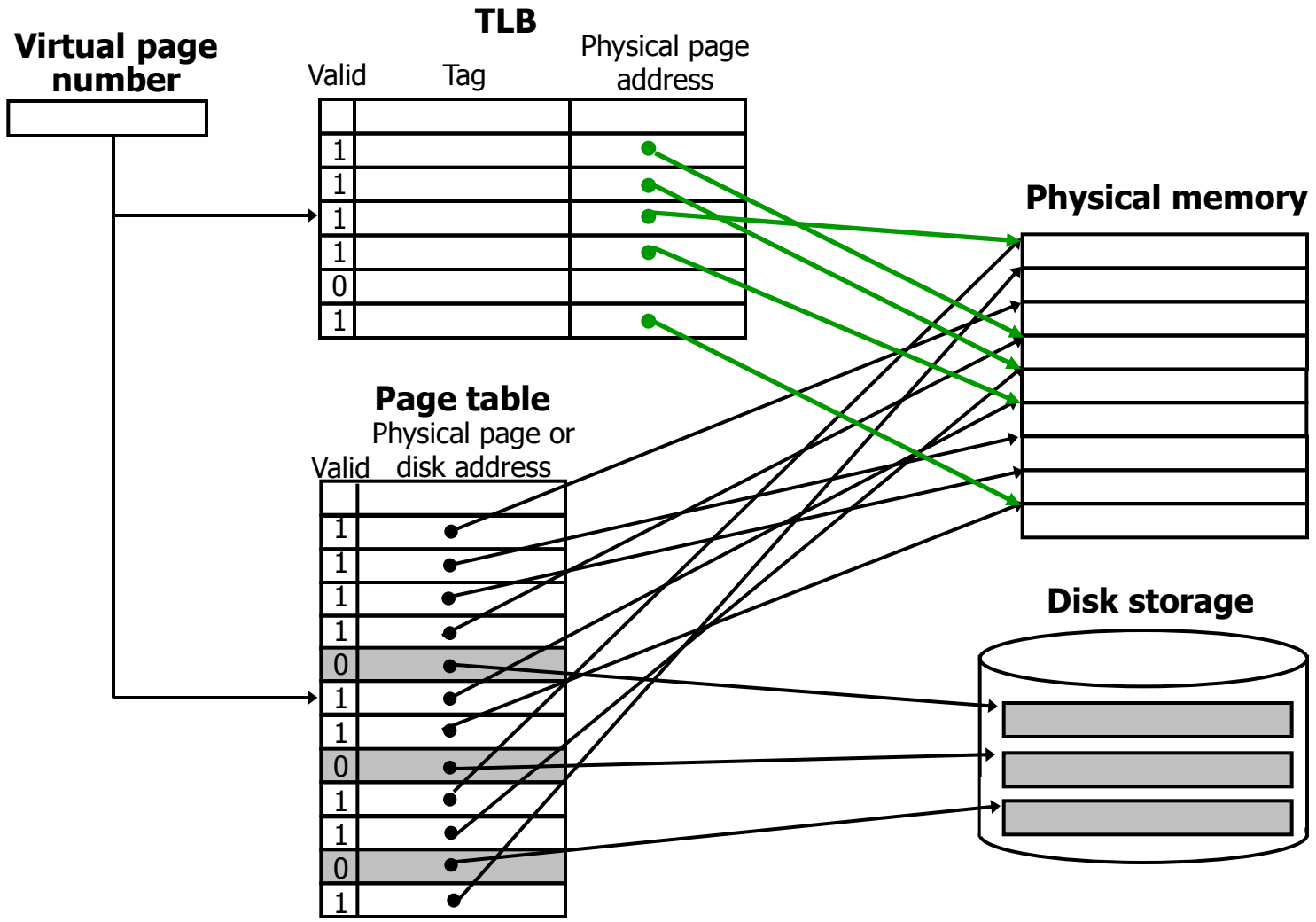
- Virtual memory => memory acts like a cache for the disk
- Page table maps virtual page numbers to physical frames
- Translation Look-aside Buffer (TLB) is a cache for translations



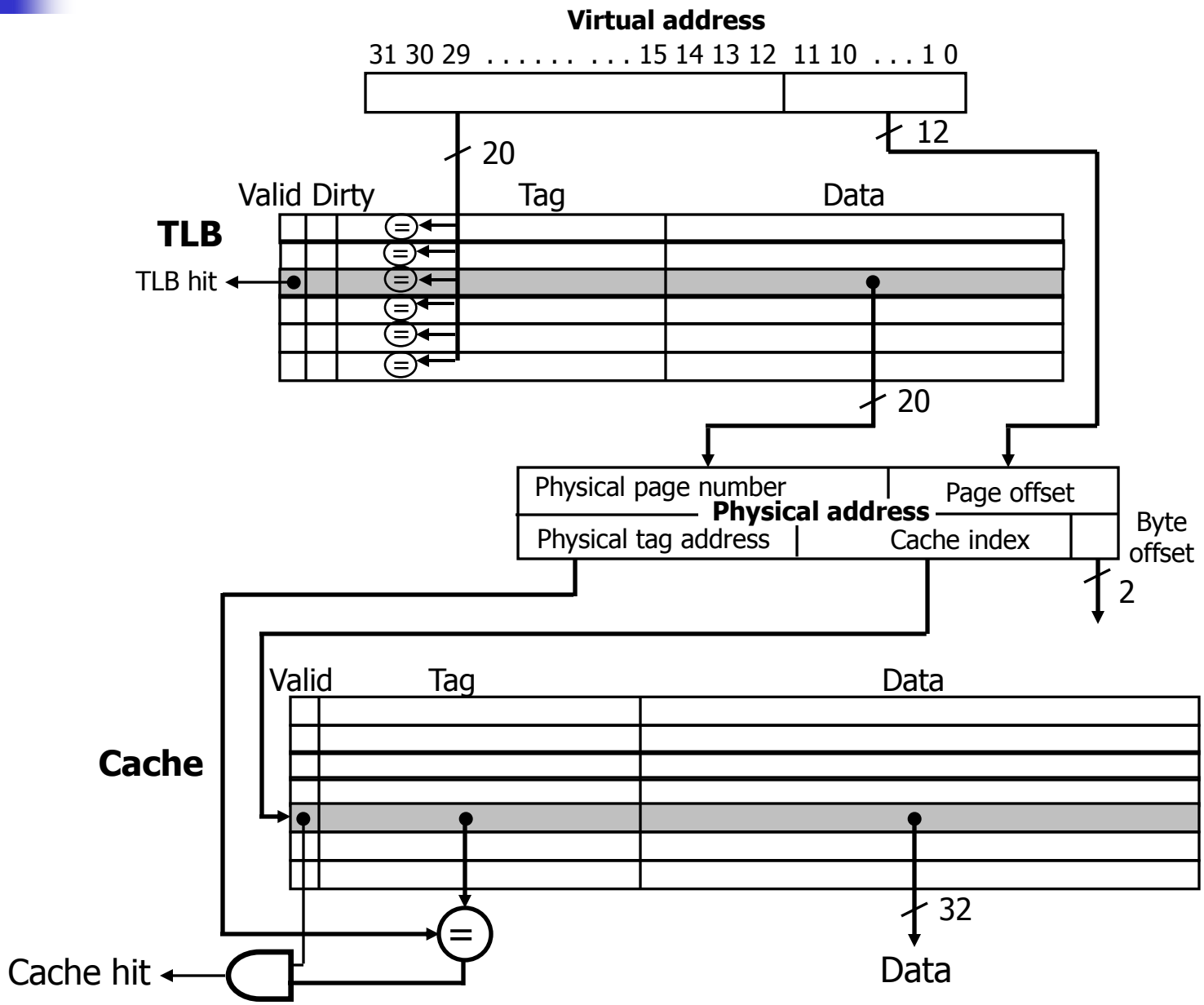


Translation-Lookaside Buffer (TLB)

- A TLB acts as a cache for the page table, by storing physical addresses of pages that have been recently accessed.

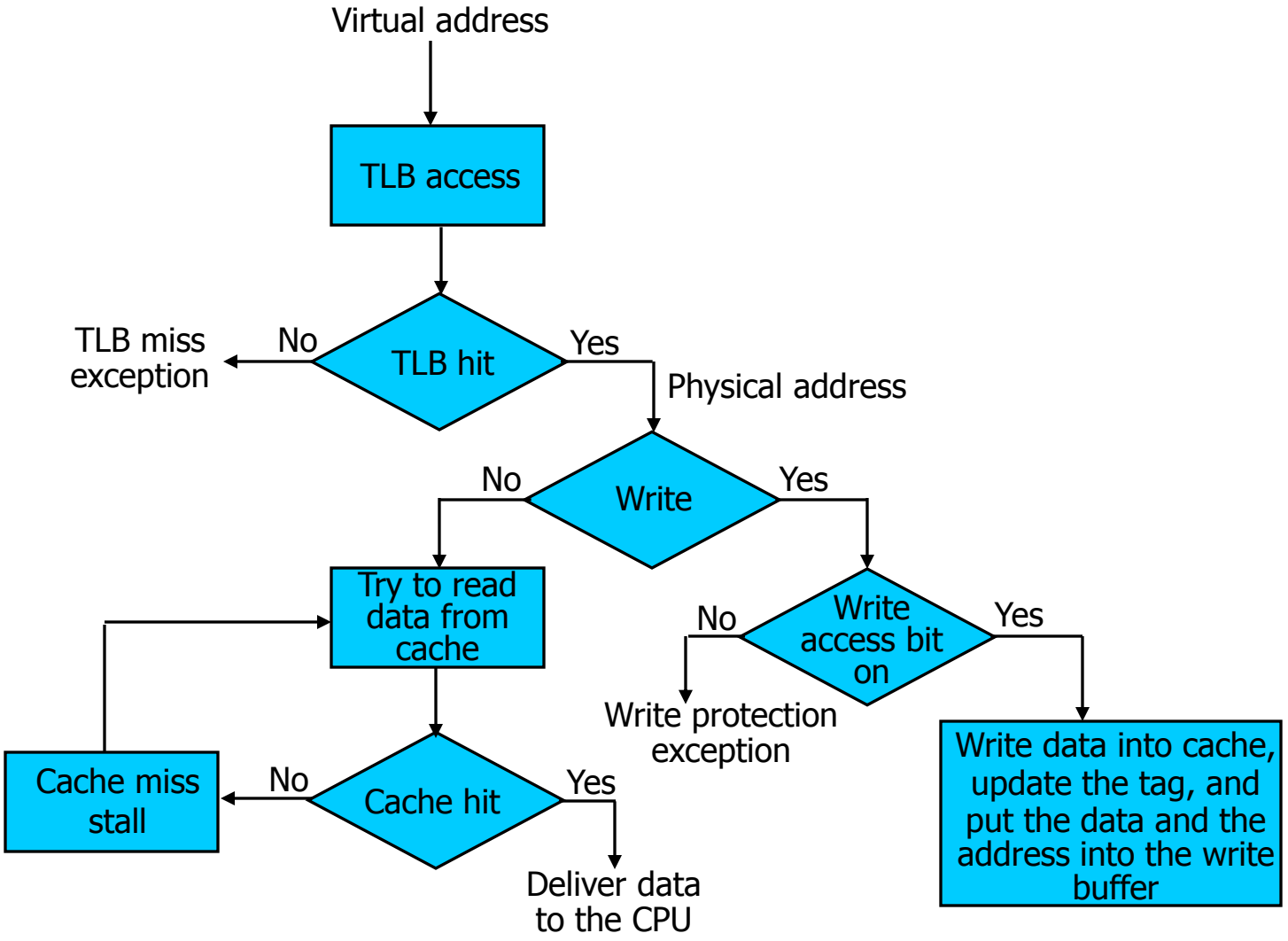


MIPS R2000 TLB and Cache (See Figure 7.24 on page 525)



TLB and Cache Operation

- On a memory access, the following operations occur.



TLB Misses

- If page is in memory
  - Load the PTE from memory and retry
  - Could be handled in hardware
    - Can get complex for more complicated page table structures
  - Or in software
    - Raise a special exception, with optimized handler
- If page is not in memory (page fault)
  - OS handles fetching the page and updating the page table
  - Then restart the faulting instruction

TLB Miss Handler

- TLB miss indicates
  - Page present, but PTE not in TLB
  - Page not preset
- Must recognize TLB miss before destination register overwritten
  - Raise exception
- Handler copies PTE from memory to TLB
  - Then restarts instruction
  - If page not present, page fault will occur

Handling TLB Misses and Page Faults

- When a TLB miss occurs either
  - Page is present in memory and update the TLB
    - occurs if valid bit of page table is set
  - Page is not present in memory and O.S. gets control to handle a page fault
- If a page fault occur, the operating system
  - Access the page table to determine the physical location of the page on disk
  - Chooses a physical page to replace - if the replaced page is dirty it is written to disk
  - Reads a page from disk into the chosen physical page in main memory.
- Since the disk access takes so long, another process is typically allowed to run during a page fault.

Page Fault Handler

- Use faulting virtual address to find PTE
- Locate page on disk
- Choose page to replace
  - If dirty, write to disk first
- Read page into memory and update page table
- Make process runnable again
  - Restart from faulting instruction

Cache and Main Memory Parameters

Parameter	L1 Cache	Main Memory
Block (page) size	16-128 bytes	4096-65,536 bytes
Hit time	1-2 cycles	40-100 cycles
Miss Penalty	8-100 cycles	1 to 6 million cycles
Miss rate	0.5-10%	0.00001-0.001%
Memory size	16 KB to 1 MB	16 MB to 8 GB

4 Qs for Virtual Memory

- Q1: Where can a block be placed in the upper level?
  - Miss penalty for virtual memory is very high
  - Have software determine location of block while accessing disk
  - Allow blocks to be place anywhere in memory (fully associative) to reduce miss rate.
- Q2: How is a block found if it is in the upper level?
  - Address divided into page number and page offset
  - Page table and translation buffer used for address translation
- Q3: Which block should be replaced on a miss?
  - Want to reduce miss rate & can handle in software
  - Least Recently Used typically used
- Q4: What happens on a write?
  - Writing to disk is very expensive
  - Use a write-back strategy

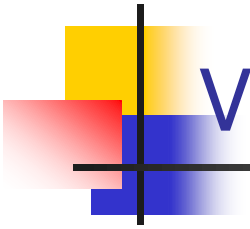
TLB organization: include protection

Virtual Address	Physical Address	Dirty	Ref	Valid	Access	ASID
0xFA00	0x0003	Y	N	Y	R/W	34
0x0040	0x0010	N	Y	Y	R	0
0x0041	0x0011	N	Y	Y	R	0

- TLB usually organized as fully-associative cache
  - Lookup is by Virtual Address
  - Returns Physical Address + other info
- Dirty => Page modified (Y/N)?
  Ref => Page touched (Y/N)?
  Valid => TLB entry valid (Y/N)?
  Access => Read? Write?
  ASID => Which User?

Memory Protection

- With multiprogramming, a computer is shared by several programs or processes running concurrently
  - Need to provide protection
  - Need to allow sharing
- Mechanisms for providing protection
  - Provide both user and supervisor (operating system) modes
  - Provide CPU state that the user can read, but cannot write
    - user/supervisor bit, page table pointer, and TLB
  - Provide method to go from user to supervisor mode and vice versa
    - system call or exception : user to supervisor
    - system or exception return : supervisor to user
  - Provide permissions for each page in memory
  - Store page tables in the operating systems address space - can't be accessed directly by user.



Virtual Memory Summary

- Virtual memory (VM) allows main memory (DRAM) to act like a cache for secondary storage (magnetic disk).
- Page tables and TLBs are used to translate the virtual address to a physical address
- The large miss penalty of virtual memory leads to different strategies from cache
  - Fully associative
  - LRU or LRU approximation
  - Write-back
  - Done by software