CS 61C: Great Ideas in Computer Architecture

Virtual Memory III

Instructor: Dan Garcia

1

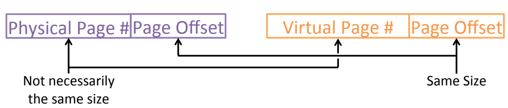
Virtual Memory Mapping Function

- How large is main memory? Disk?
  - Don't know! Designed to be interchangeable components
  - Need a system that works regardless of sizes
- Use lookup table (*page table*) to deal with arbitrary mapping
  - Index lookup table by # of pages in VM (not all entries will be used/valid)
  - Size of PM will affect size of stored translation

6

Address Mapping

- Pages are aligned in memory
  - Border address of each page has same lowest bits
  - Page size is same in VM and PM, so denote lowest O = $\log_2$(page size/byte) bits as *page offset*
- Use remaining upper address bits in mapping
  - Tells you which page you want (similar to Tag)



7

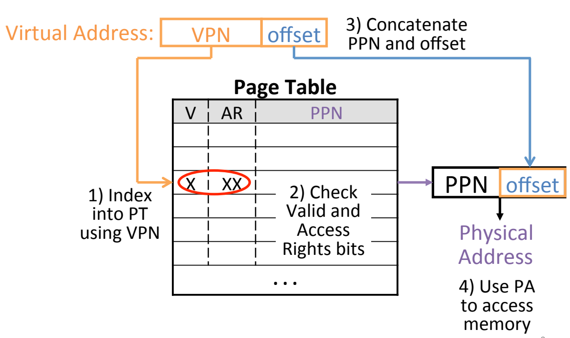
Address Mapping: Page Table

- **Page Table functionality:**
  - Incoming request is Virtual Address (VA), want Physical Address (PA)
  - Physical Offset = Virtual Offset (page-aligned)
  - So just swap Virtual Page Number (VPN) for Physical Page Number (PPN)

- **Implementation?**
  - Use VPN as index into PT
  - Store PPN and management bits (Valid, Access Rights)
  - Does NOT store actual data (the data sits in PM)

8

Page Table Layout



9

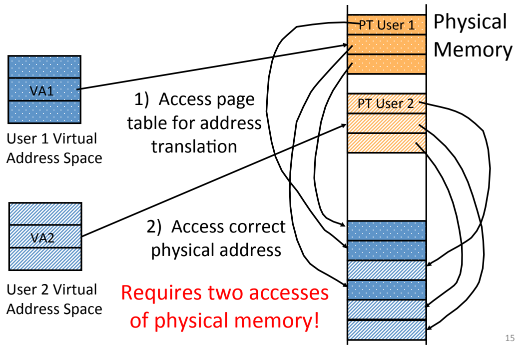
Question: How many bits wide are the following fields?

- 16 KiB pages
- 40-bit virtual addresses
- 64 GiB physical memory

	VPN	PPN
A)	26	26
B)	24	20
C)	22	22
D)	26	22

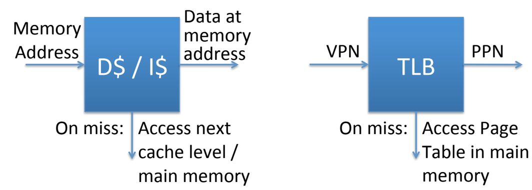
14

Retrieving Data from Memory



15

TLBs vs. Caches

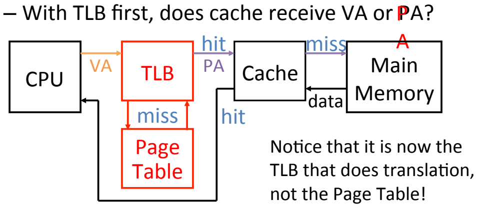


- TLBs usually small, typically 16 – 512 entries
- TLB access time comparable to cache (« main memory)
- TLBs can have associativity
  - Usually fully/highly associative

17

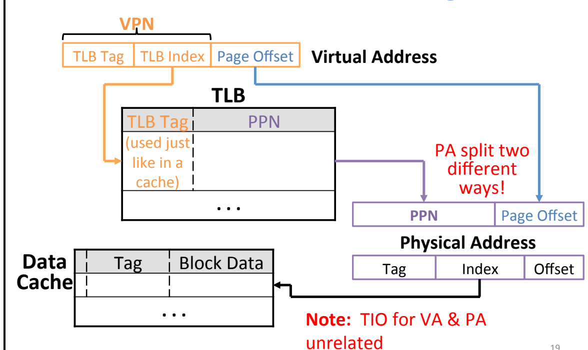
Where Are TLBs Located?

- Which should we check first: Cache or TLB?
  - Can cache hold requested data if corresponding page is not in physical memory? No
  - With TLB first, does cache receive VA or PA?



18

Address Translation Using TLB



19

Question: How many bits wide are the following?

- 16 KiB pages
- 40-bit virtual addresses
- 64 GiB physical memory
- 2-way set associative TLB with 512 entries

Valid	Dirty	Ref	Access Rights	TLB Tag	PPN
X	X	X	XX		

	TLB Tag	TLB Index	TLB Entry
A)	12	14	38
B)	18	8	45
C)	14	12	40
D)	17	9	43

21

Administrivia

- TAs?

23

Fetching Data on a Memory Read

1) Check TLB (input: VPN, output: PPN)
   - *TLB Hit:* Fetch translation, return PPN
   - *TLB Miss:* Check page table (in memory)
     - *Page Table Hit:* Load page table entry into TLB
     - *Page Table Miss (Page Fault):* Fetch page from disk to memory, update corresponding page table entry, then load entry into TLB
2) Check cache (input: PPN, output: data)
   - *Cache Hit:* Return data value to processor
   - *Cache Miss:* Fetch data value from memory, store it in cache, return it to processor

25

Page Faults

- Load the page off the disk into a free page of memory
  - Switch to some other process while we wait
- Interrupt thrown when page loaded and the process' page table is updated
  - When we switch back to the task, the desired data will be in memory
- If memory full, replace page (LRU), writing back if necessary, and update *both* page table entries
  - Continuous swapping between disk and memory called "thrashing"

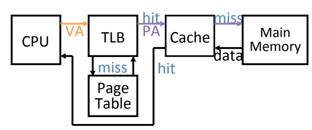
26

Performance Metrics

- VM performance also uses Hit/Miss Rates and Miss Penalties
  - *TLB Miss Rate:* Fraction of TLB accesses that result in a TLB Miss
  - *Page Table Miss Rate:* Fraction of PT accesses that result in a page fault
- Caching performance definitions remain the same
  - Somewhat independent, as TLB will always pass PA to cache regardless of TLB hit or miss

27

Data Fetch Scenarios



- Are the following scenarios for a single data access possible?
  - TLB Miss, Page Fault — Yes
  - TLB Hit, Page Table Hit — No
  - TLB Miss, Cache Hit — Yes
  - Page Table Hit, Cache Miss — Yes
  - Page Fault, Cache Hit — No

28

Question: A program tries to load a word at X that causes a TLB miss but not a page fault. Are the following statements TRUE or FALSE?

1) The page table does not contain a valid mapping for the virtual page corresponding to the address X
2) The word that the program is trying to load is present in physical memory

	1	2
A)	F	F
B)	**F**	**T**
C)	T	F
D)	T	T

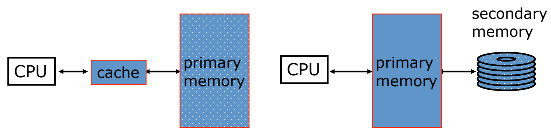
29

VM Performance

- Virtual Memory is the level of the memory hierarchy that sits *below* main memory
  - TLB comes *before* cache, but affects transfer of data from disk to main memory
  - Previously we assumed main memory was lowest level, now we just have to account for disk accesses
- Same CPI, AMAT equations apply, but now treat main memory like a mid-level cache

30

Typical Performance Stats



Caching
cache entry
cache block (≈32 bytes)
cache miss rate (1% to 20%)
cache hit (≈1 cycle)
cache miss (≈100 cycles)

Demand paging
page frame
page (≈4Ki bytes)
page miss rate (<0.001%)
page hit (≈100 cycles)
page miss (≈5M cycles)

31

Impact of Paging on AMAT (1/2)

- Memory Parameters:
  - L1 cache hit = 1 clock cycles, hit 95% of accesses
  - L2 cache hit = 10 clock cycles, hit 60% of L1 misses
  - DRAM = 200 clock cycles (≈100 nanoseconds)
  - Disk = 20,000,000 clock cycles (≈10 milliseconds)
- Average Memory Access Time (no paging):
  - 1 + 5%×10 + 5%×40%×200 = 5.5 clock cycles
- Average Memory Access Time (with paging):
  - 5.5 (AMAT with no paging) + ?

32

Impact of Paging on AMAT (2/2)

- Average Memory Access Time (with paging) =
  - 5.5 + 5%×40%× (1-HR_{Mem})×20,000,000
- AMAT if HR_{Mem} = 99%?
  - 5.5 + 0.02×0.01×20,000,000 = 4005.5 (≈728x slower)
  - 1 in 20,000 memory accesses goes to disk: 10 sec program takes 2 hours!
- AMAT if HR_{Mem} = 99.9%?
  - 5.5 + 0.02×0.001×20,000,000 = 405.5
- AMAT if HR_{Mem} = 99.9999%
  - 5.5 + 0.02×0.000001×20,000,000 = 5.9

33

Impact of TLBs on Performance

- Each TLB miss to Page Table ~ L1 Cache miss
- *TLB Reach:* Amount of virtual address space that can be simultaneously mapped by TLB:
  - TLB typically has 128 entries of page size 4-8 KiB
  - 128 × 4 KiB = 512 KiB = just 0.5 MiB
- What can you do to have better performance?
  - Multi-level TLBs ← Conceptually same as multi-level caches
  - Variable page size (segments)
  - Special situationally-used "superpages"

  (Variable page size and superpages: Not covered here)

34

Aside: Context Switching

- How does a single processor run many programs at once?
- *Context switch:* Changing of internal state of processor (switching between processes)
  - Save register values (and PC) and change value in Page Table Base register
- What happens to the TLB?
  - Current entries are for different process
  - Set all entries to invalid on context switch

35

Virtual Memory Summary

- User program view:
  - Contiguous memory
  - Start from some set VA
  - "Infinitely" large
  - Is the only running program
- Reality:
  - Non-contiguous memory
  - Start wherever available memory is
  - Finite size
  - Many programs running simultaneously
- Virtual memory provides:
  - Illusion of contiguous memory
  - All programs starting at same set address
  - Illusion of ~ infinite memory (2^{32} or 2^{64} bytes)
  - Protection, Sharing
- Implementation:
  - Divide memory into chunks (pages)
  - OS controls page table that maps virtual into physical addresses
  - memory as a cache for disk
  - TLB is a cache for the page table

36