

# CS 61C: Great Ideas in Computer Architecture

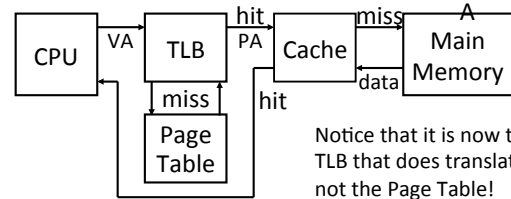
## Virtual Memory III

Miki Lustig

1

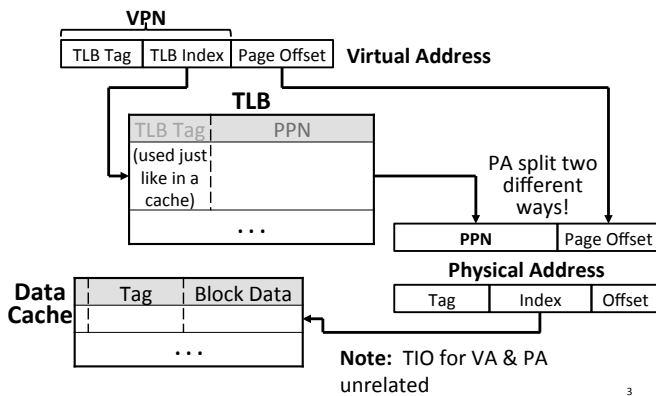
## Where Are TLBs Located?

- 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?



2

## Address Translation Using TLB



3



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

4

## 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

7

## 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”

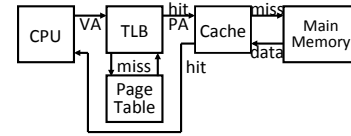
8

## 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

9

## 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

10

**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

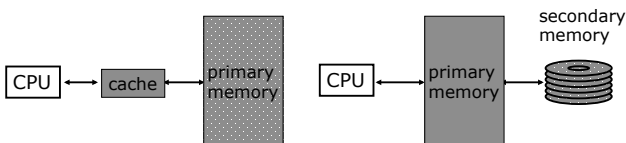
11

## 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

13

## 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)

14

## 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):
  - $1 + 5\% \times 10 + 5\% \times 40\% \times 200 = 5.5$  clock cycles
- Average Memory Access Time (with paging):
  - 5.5 (AMAT with no paging) + ?

15

## Impact of Paging on AMAT (2/2)

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

16

## Impact of TLBs on Performance

- Each TLB miss to Page Table  $\sim$  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
  - $128 \times 4 \text{ KiB} = 512 \text{ KiB} = \text{just } 0.5 \text{ MiB}$
- What can you do to have better performance?
  - Multi-level TLBs  $\leftarrow$  Conceptually same as multi-level caches
  - Variable page size (segments)
  - Special situationally-used “superpages” } Not covered here

17

## 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

18

## 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  $\sim$  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

19