Caches & Virtual Memory

Scott Beamer (cs61c-ta)

Cache Misses

- *Compulsory*: first time block has been accessed
  - i.e. would still happen with infinite sized caches
- *Conflict*: due to collision caused by block placement policy
  - i.e. would not happen under full associativity
- *Capacity*: due to cache being too small
  - i.e. would still happen with full associativity and perfect replacement policy

Cache Associativity

- *Fully Associative*: cache blocks can go anywhere in the cache (no index bits)
- *N-Way Set Associativity*: cache block can go in any of N ways per set - thus:
  - # of sets = (cache capacity) / (block size) / (set associativity)
  - Fully associative cache is N-way set associative, where N is number of blocks
  - Direct mapped cache is 1-way set associative
- Increased associativity reduces conflict misses, but needs more comparators and logic

Cache Details

- *Average Memory Access Time (AMAT)*: how long it takes to access memory
  - AMAT = hit time + (miss time)(miss penalty)
- *Write Through*: every write happens at the current level as well as level above
- *Write Back*: writes only happen at current level, and are written back at eviction

Cache Problem

- For the given cache parameters, guess what the AMAT would be (inverse of Lab 12)
- 32KB cache with 32B blocks that is direct mapped
- Cache hit time of 10ns and cache miss time (memory access time) of 64ns

Array Size \Stride	4B	8B	16B	32B	64B	128B
16KB						
32KB						
64KB						
128KB						

- For when there are any cache hits above, which locality (spatial or temporal) is it?

Virtual Memory Overview

- *Virtual Memory* (commonly and in this course) actually refers to three concepts:
  - *Translation*: map one address space to another to give illusion of dedicated space
  - *Virtual Memory*: use backing store (disk) to give illusion of larger memory
  - *Protection*: only allow access to parts of memory for certain programs
- *Sharing*: allow multiple programs to access same memory

How Virtual Memory Works

- OS adds level of indirection to every memory access made by a program
- Each program requests a *virtual address* that is mapped to a *physical address* by VM
- *Pages:* The unit of transfer for VM (like block for caches)
  - Done to reduce the internal fragmentation problems of base & bound
- Each address breaks into an offset (for within the page) and page number

Virtual Address	
Virtual Page Number (VPN)	Page Offset

Physical Address	
Physical Page Number (PPN)	Page Offset

- Virtual address -> physical address: OS keeps page offset, and turns VPN into PPN
- *Page Table*: A direct map for each program that translates VPNs to PPNs and stores relevant information such as: valid, dirty, access rights, and if the page is in memory or disk. It is a software data structure maintained by the OS.
- *Translation Lookaside Buffer (TLB)*: Hardware that stores a fraction of the Page Table for quicker access (like a cache).

Qualitative VM Questions

- Why is the TLB placed between the CPU and cache?

- What aspects of virtual memory do you need for multiprogramming?

- Sharing can be implemented using which two aspects?

VM Parameters Problem

- For each change, indicate how it would affect the VM performance

	Page Table Size	TLB Hit Rate	Internal Fragmentation
Increase Page Size			
Increase Virtual Address Size			
Decrease Physical Address Size			