CSE 490/590 Computer Architecture

Cache II

Steve Ko
Computer Sciences and Engineering
University at Buffalo

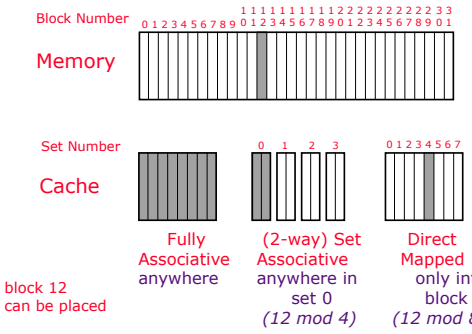
Last time…

- Dynamic RAM (DRAM) is main form of main memory storage in use today
  - Holds values on small capacitors, need refreshing (hence dynamic)
  - Slow multi-step access: precharge, read row, read column
- Static RAM (SRAM) is faster but more expensive
  - Used to build on-chip memory for caches
- Cache holds small set of values in fast memory (SRAM) close to processor
  - Need to develop search scheme to find values in cache, and replacement policy to make space for newly accessed locations
- Caches exploit two forms of predictability in memory reference streams
  - Temporal locality, same location likely to be accessed again soon
  - Spatial locality, neighboring location likely to be accessed soon

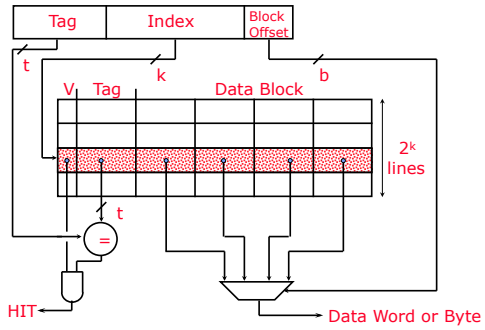
Some Basics (Again)

- Block: the unit of access/storage in cache
- Word: the unit of access by CPU
- A block contains multiple words.
  - Why multiple words?
- On cache miss,
  - Memory access
  - Cache block (re)placement
  - Why keep it?
- Five things to decide
  - After fetching a block from the memory, where do we place it inside the cache?
  - If the line is taken or the cache is full already, which block to evict?
  - How many words per block?
  - How big?
  - What happens on write?

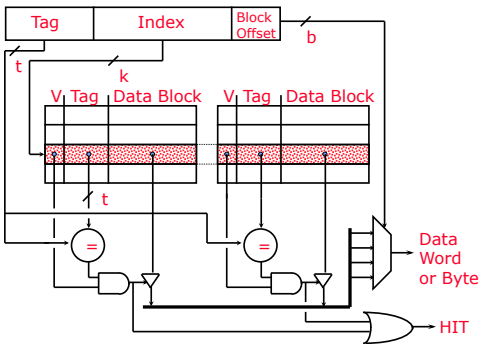
Placement Policy



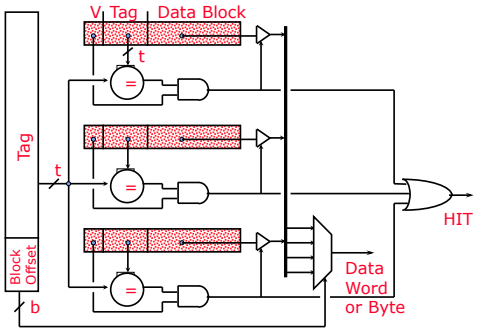
Direct-Mapped Cache



2-Way Set-Associative Cache



Fully Associative Cache



Replacement Policy

In an associative cache, which block from a set should be evicted when the set becomes full?

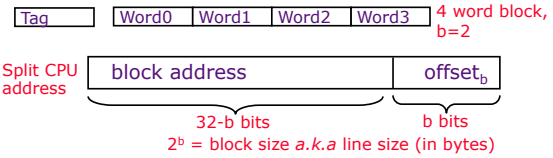
- Random
- Least Recently Used (LRU)
  - LRU cache state must be updated on every access
  - true implementation only feasible for small sets (2-way)
  - pseudo-LRU binary tree often used for 4-8 way
- First In, First Out (FIFO) a.k.a. Round-Robin
  - used in highly associative caches
- Not Most Recently Used (NMRU)
  - FIFO with exception for most recently used block or blocks

This is a second-order effect. Why?

Replacement only happens on misses

Block Size and Spatial Locality

Block is unit of transfer between the cache and memory

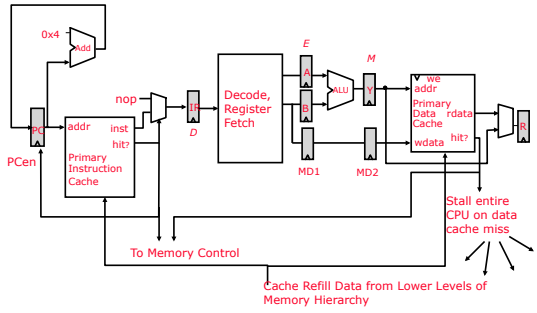


Larger block size has distinct hardware advantages
- less tag overhead
- exploit fast burst transfers from DRAM
- exploit fast burst transfers over wide busses

What are the disadvantages of increasing block size?

Fewer blocks => more conflicts. Can waste bandwidth.

CPU-Cache Interaction
(5-stage pipeline)



Improving Cache Performance

Average memory access time =
Hit time + Miss rate x Miss penalty

To improve performance:
- reduce the hit time
- reduce the miss rate
- reduce the miss penalty

What is the simplest design strategy?

Biggest cache that doesn't increase hit time past 1-2 cycles (approx 8-32KB in modern technology)

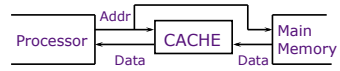
[design issues more complex with out-of-order superscalar processors]

Serial-versus-Parallel Cache and Memory access

α is HIT RATIO: Fraction of references in cache

1 - α is MISS RATIO: Remaining references

Average access time for serial search: $t_{cache} + (1 - \alpha)\, t_{mem}$



Average access time for parallel search: $\alpha\, t_{cache} + (1 - \alpha)\, t_{mem}$

- Savings are usually small, $t_{mem} >> t_{cache}$, hit ratio α high
- High bandwidth required for memory path
- Complexity of handling parallel paths can slow t_{cache}

CSE 490/590 Administrivia

- Feedback on lectures
  - If you have any feedback/concern, please send it along to me
  - Thanks to those who already did
  - Please ask questions if things are not clear
  - Or you can simply scream, "TOO FAST!"
  - Please utilize my office hours (I will change to sometime in the afternoon)
- Very important to attend
  - Recitations this week & next week
- Quiz 1
  - Fri, 2/11
  - Closed book, in-class
  - Includes lectures until last Monday (1/31)
  - Review: Wed (2/9)

Acknowledgements

- These slides heavily contain material developed and copyright by
  - Krste Asanovic (MIT/UCB)
  - David Patterson (UCB)
- And also by:
  - Arvind (MIT)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
- MIT material derived from course 6.823
- UCB material derived from course CS252