CS 61C: Great Ideas in Computer Architecture (Machine Structures)

Instructors:
Randy H. Katz
David A. Patterson
http://inst.eecs.Berkeley.edu/~cs61c/fa10

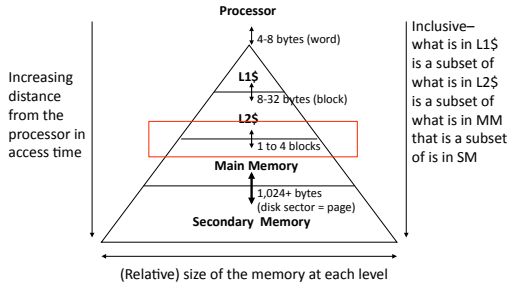
Agenda

- Direct Mapped Cache (continued)
- Administrivia
- Technology Break
- Cache-Memory Interface

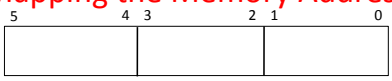
Agenda

- Direct Mapped Cache
- Administrivia
- Technology Break
- Cache-Memory Interface

Characteristics of the Memory Hierarchy



Mapping the Memory Address

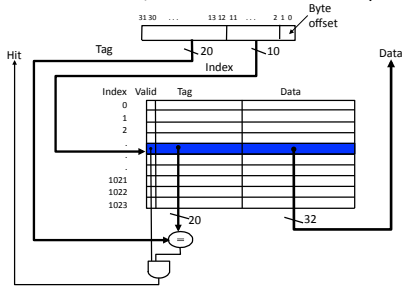


Mem Block Within $ Block *Tag* | Block Within $ *Index* | Byte Within Block (e.g., Word)

- Note: $ = Cache
- In example, block size is 4 bytes/1 word (it could be multi-word)
- Memory and cache blocks are the same size, unit of transfer between memory and cache
- # Memory blocks >> # Cache blocks
  - 16 Memory blocks/16 words/64 bytes/6 bits to address all bytes
  - 4 Cache blocks, 4 bytes (1 word) per block
  - 4 Memory blocks map to each cache block
- Byte within block: low order two bits, ignore! (nothing smaller than a block)
- Memory block to cache block, aka *index*: middle two bits
- Which memory block is in a given cache block, aka *tag*: top two bits

Direct Mapped Cache Example

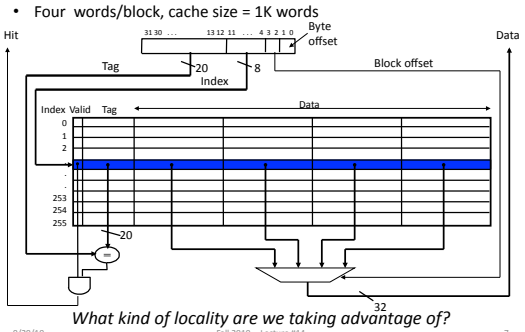
- One word blocks, cache size = 1K words (or 4KB)



What kind of locality are we taking advantage of?

Multiword Block Direct Mapped Cache

- Four words/block, cache size = 1K words



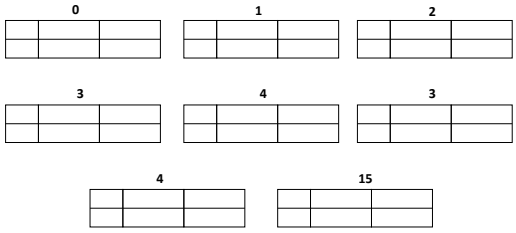
What kind of locality are we taking advantage of?

Taking Advantage of Spatial Locality

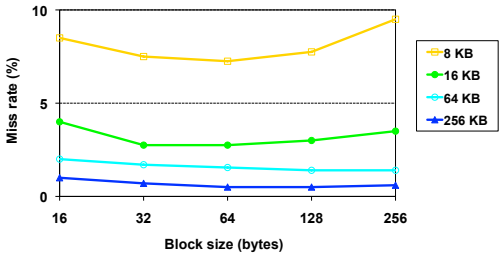
- Let cache block hold more than one word

Start with an empty cache - all blocks initially marked as not valid

0 1 2 3 4 3 4 15



Miss Rate vs Block Size vs Cache Size



- Miss rate goes up if the block size becomes a significant fraction of the cache size because the number of blocks that can be held in the same size cache is smaller (increasing capacity misses)

Cache Field Sizes

- Number of bits in a cache includes both the storage for data and for the tags
  - 32-bit byte address
  - For a direct mapped cache with 2n blocks, n bits are used for the index
  - For a block size of 2m words (2m+2 bytes), m bits are used to address the word within the block and 2 bits are used to address the byte within the word
- What is the size of the tag field?
- The total number of bits in a direct-mapped cache is then
  - 2n x (block size + tag field size + valid field size)
- How many total bits are required for a direct mapped cache with 16KB of data and 4-word blocks assuming a 32-bit address?

Handling Cache Hits

- Read hits (I$ and D$)
  - Hits are good in helping us go fast; misses are bad/slow us down
- Write hits (D$ only)
  - Require the cache and memory to be consistent
    - Always write the data into both the cache block and the next level in the memory hierarchy (write-through)
    - Writes run at the speed of the next level in the memory hierarchy – so slow! – or can use a write buffer and stall only if the write buffer is full
  - Allow cache and memory to be inconsistent
    - Write the data only into the cache block (write-back the cache block to the next level in the memory hierarchy when that cache block is "evicted")
    - Need a dirty bit for each data cache block to tell if it needs to be written back to memory when it is evicted – can use a write buffer to help "buffer" write-backs of dirty blocks

Sources of Cache Misses

- *Compulsory* (cold start or process migration, first reference):
  - First access to a block, "cold" fact of life, not a whole lot you can do about it. If you are going to run "millions" of instruction, compulsory misses are insignificant
  - Solution: increase block size (increases miss penalty; very large blocks could increase miss rate)
- *Capacity*:
  - Cache cannot contain all blocks accessed by the program
  - Solution: increase cache size (may increase access time)
- *Conflict* (collision):
  - Multiple memory locations mapped to the same cache location
  - Solution 1: increase cache size
  - Solution 2: increase associativity (stay tuned) (may increase access time)

Handling Cache Misses (Single Word Blocks)

- Read misses (I$ and D$)
  - Stall the pipeline, fetch the block from the next level in the memory hierarchy, install it in the cache and send the requested word to the processor, then let the pipeline resume
- Write misses (D$ only)
  - Stall the pipeline, fetch the block from next level in the memory hierarchy, install it in the cache (which may involve having to evict a dirty block if using a write-back cache), write the word from the processor to the cache, then let the pipeline resume

  or
  - Write allocate – just write the word into the cache updating both the tag and data, no need to check for cache hit, no need to stall

  or
  - No-write allocate – skip the cache write (but must invalidate that cache block since it will now hold stale data) and just write the word to the write buffer (and eventually to the next memory level), no need to stall if the write buffer isn't full

Multiword Block Considerations

- Read misses (I$ and D$)
  - Processed the same as for single word blocks – a miss returns the entire block from memory
  - Miss penalty grows as block size grows
    - Early restart – processor resumes execution as soon as the requested word of the block is returned
    - Requested word first – requested word is transferred from the memory to the cache (and processor) first
  - Nonblocking cache – allows the processor to continue to access the cache while the cache is handling an earlier miss
- Write misses (D$)
  - If using write allocate must first fetch the block from memory and then write the word to the block (or could end up with a "garbled" block in the cache (e.g., for 4 word blocks, a new tag, one word of data from the new block, and three words of data from the old block)

Agenda

- Direct Mapped Caches
- Administrivia
- Technology Break
- Cache-Memory Interface

Midterm!

- HW #3: Posted, Due SUNDAY@23:59:59
- Exam Review on Monday, 100 GPB 6-8 PM
- NO lecture next Wednesday, 6 October
- Exam, 6-9 PM, 1 Pimentel
  - Closed book, notes
  - No calculator
  - One 8.5" x 11" crib sheet

Agenda

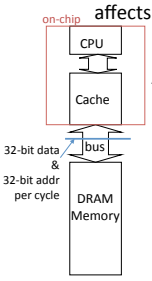
- Direct Mapped Caches
- Administrivia
- Technology Break
- Caches-Memory Interface

Agenda

- Cache Hits and Misses
- Administrivia
- Technology Break
- Caches-Memory Interface

Memory Systems that Support Caches

- The off-chip interconnect and memory architecture affects overall system performance in dramatic ways



One word wide organization (one word wide bus and one word wide memory)

Assume

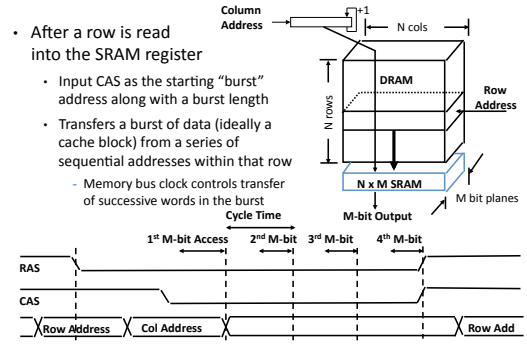
- 1 memory bus clock cycle to send address
- 15 memory bus clock cycles to get the 1st word in the block from DRAM (row cycle time), 5 memory bus clock cycles for 2nd, 3rd, 4th words (subsequent column *access* time)—note effect of latency!
- 1 memory bus clock cycle to return a word of data

Memory-Bus to Cache bandwidth

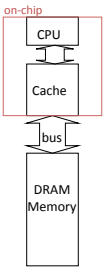
- Number of bytes accessed from memory and transferred to cache/CPU per memory bus clock cycle

(DDR) SDRAM Operation

- After a row is read into the SRAM register
  - Input CAS as the starting "burst" address along with a burst length
  - Transfers a burst of data (ideally a cache block) from a series of sequential addresses within that row
    - Memory bus clock controls transfer of successive words in the burst



One Word Wide Bus, One Word Blocks



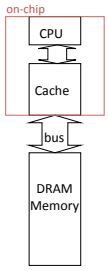
- If block size is one word, then for a memory access due to a cache miss, the pipeline will have to *stall* for the number of cycles required to return one data word from memory

  memory bus clock cycle to send address
  memory bus clock cycles to read DRAM
  memory bus clock cycle to return data
  total clock cycles miss penalty

- Number of bytes transferred per clock cycle (bandwidth) for a single miss is

  bytes per memory bus clock cycle

One Word Wide Bus, Four Word Blocks



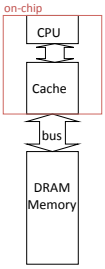
- What if the block size is four words and each word is in a different DRAM row?

  cycle to send 1st address
  cycles to read DRAM
  cycles to return last data word
  total clock cycles miss penalty

- Number of bytes transferred per clock cycle (bandwidth) for a single miss is

  bytes per clock

One Word Wide Bus, Four Word Blocks



- What if block size is four words and all words are in the same DRAM row?

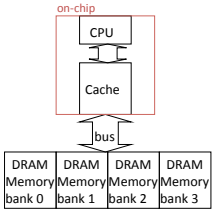
  cycle to send 1st address
  cycles to read DRAM
  cycles to return last data word
  total clock cycles miss penalty

- Number of bytes transferred per clock cycle (bandwidth) for a single miss is

  bytes per clock

Interleaved Memory, One Word Wide Bus

- For a block size of four words



  cycle to send 1st address
  cycles to read DRAM banks
  cycles to return last data word
  total clock cycles miss penalty

- Number of bytes transferred per clock cycle (bandwidth) for a single miss is

  bytes per clock

DRAM Memory System Observations

- Its important to match the cache characteristics
  - Caches access one block at a time (usually more than one word)
- With the DRAM characteristics
  - Use DRAMs that support fast multiple word accesses, preferably ones that match the block size of the cache
- With the memory-bus characteristics
  - Make sure the memory-bus can support the DRAM access rates and patterns
  - With the goal of increasing the Memory-Bus to Cache bandwidth

Summary

- Hits in caches hide the long access latencies to main memory
- Misses suffer from the high latency of going to main memory—process may have to wait for memory in these cases unless other tricks are used (future parallelism discussions!)
- Mitigate the effect of the latency by exploiting memory bandwidth and parallelism to move a block from memory to cache in the hope that locality will increase future hits