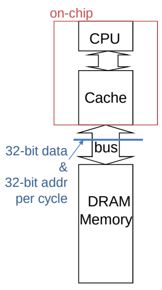
Memory System Issues

Memory Systems that Support Caches

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



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

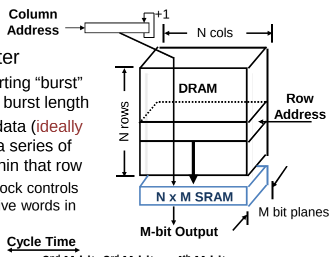
- Assume
  1. 1 memory bus clock cycle to send the addr
  2. 15 memory bus clock cycles to get the 1st word in the block from DRAM (row cycle time), 5 memory bus clock cycles for 2nd, 3rd, 4th words (column access time)
  3. 1 memory bus clock cycle to return a word of data
- Memory-Bus to Cache bandwidth
  - number of bytes accessed from memory and transferred to cache/CPU per memory bus clock cycle

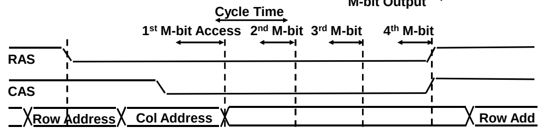
Advanced DRAM Operation

- Bits in a DRAM are organized as a rectangular array
  - DRAM accesses an entire row
  - Burst mode: supply successive words from a row with reduced latency
- Double data rate (DDR) DRAM
  - Transfer on rising and falling clock edges
- Quad data rate (QDR) DRAM
  - Separate DDR inputs and outputs

DRAM Operation

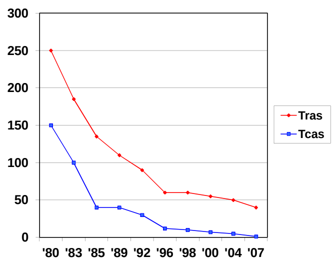
- After a row is read into the SRAM register
  - Input CAS as the starting "burst" address along with a burst length
  - Transfers a burst of data (ideally a cache block) from a series of sequential addr's within that row
    - The memory bus clock controls transfer of successive words in the burst



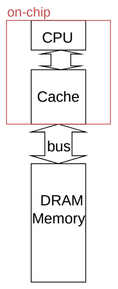


DRAM Generations

Year	Capacity	$/GB
1980	64Kbit	$1500000
1983	256Kbit	$500000
1985	1Mbit	$200000
1989	4Mbit	$50000
1992	16Mbit	$15000
1996	64Mbit	$10000
1998	128Mbit	$4000
2000	256Mbit	$1000
2004	512Mbit	$250
2007	1Gbit	$50



One Word Wide Bus, One Word Blocks

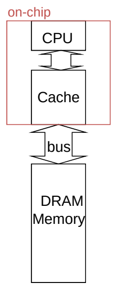


- If the block size is one word, then for a memory access due to a cache miss, the pipeline will have to stall for the number of cycles required to return one data word from memory

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

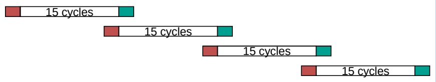
- Number of bytes transferred per clock cycle (bandwidth) for a single miss is
  4/17 = 0.235 bytes per memory bus clock cycle

One Word Wide Bus, Four Word Blocks



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

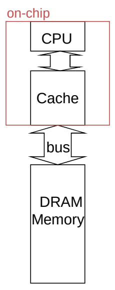
	1	cycle to send 1st address
4 x 15 =	60	cycles to read DRAM
	1	cycles to return last data word
	62	total clock cycles miss penalty



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

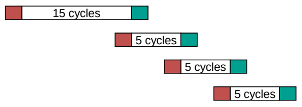
  (4 x 4)/62 = 0.258 bytes per clock

One Word Wide Bus, Four Word Blocks



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

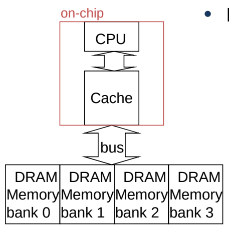
1	cycle to send 1st address
15 + 3*5 = 30	cycles to read DRAM
1	cycles to return last data word
32	total clock cycles miss penalty



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

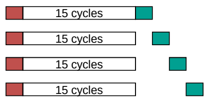
  (4 x 4)/32 = 0.5 bytes per clock

Interleaved Memory, One Word Wide Bus



- For a block size of four words

1	cycle to send 1st address
15	cycles to read DRAM banks
4*1 = 4	cycles to return last data word
20	total clock cycles miss penalty



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

  (4x4)/20 = 0.8 bytes per clock

DRAM Memory System Summary

- 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

Measuring Cache Performance

- Assuming cache hit costs are included as part of the normal CPU execution cycle, then

$$\text{CPU time} = \text{IC} \times \text{CPI} \times \text{CC}$$
$$= \text{IC} \times \underbrace{(\text{CPI}_{ideal} + \text{Memory-stall cycles})}_{\text{CPI}_{stall}} \times \text{CC}$$

- Memory-stall cycles come from cache misses (a sum of read-stalls and write-stalls)

  Read-stall cycles = reads/program × read miss rate × read miss penalty

  Write-stall cycles = (writes/program × write miss rate × write miss penalty) + write buffer stalls

- For write-through caches, we can simplify this to

  Memory-stall cycles = accesses/program × miss rate × miss penalty

Impacts of Cache Performance

- Relative cache penalty increases as processor performance improves (faster clock rate and/or lower CPI)
  - The memory speed is unlikely to improve as fast as processor cycle time. When calculating CPI_{stall}, the cache miss penalty is measured in *processor* clock cycles needed to handle a miss
  - The lower the CPI_{ideal}, the more pronounced the impact of stalls
- A processor with a CPI_{ideal} of 2, a 100 cycle miss penalty, 36% load/store instr's, and 2% I$ and 4% D$ miss rates

  Memory-stall cycles = 2% × 100 + 36% × 4% × 100 = 3.44

  So CPI_{stalls} = 2 + 3.44 = **5.44**

  more than twice the CPI_{ideal} !
- What if the CPI_{ideal} is reduced to 1? 0.5? 0.25?
- What if the D$ miss rate went up 1%? 2%?
- What if the processor clock rate is doubled (doubling the miss penalty)?

Average Memory Access Time (AMAT)

- A larger cache will have a longer access time. An increase in hit time will likely add another stage to the pipeline. At some point the increase in hit time for a larger cache will overcome the improvement in hit rate leading to a decrease in performance.
- Average Memory Access Time (AMAT) is the average to access memory considering both hits and misses

AMAT = Time for a hit + Miss rate x Miss penalty

- What is the AMAT for a processor with a 20 psec clock, a miss penalty of 50 clock cycles, a miss rate of 0.02 misses per instruction and a cache access time of 1 clock cycle?

Reducing Cache Miss Rates #2

2. Use multiple levels of caches

- With advancing technology have more than enough room on the die for bigger L1 caches *or* for a second level of caches – normally a unified L2 cache (i.e., it holds both instructions and data) and in some cases even a unified L3 cache
- For our example, CPI_{ideal} of 2, 100 cycle miss penalty (to main memory) and a 25 cycle miss penalty (to UL2\$), 36% load/stores, a 2% (4%) L1 I\$ (D\$) miss rate, add a 0.5% UL2\$ miss rate

$CPI_{stalls} = 2 + .02\times25 + .36\times.04\times25 + .005\times100 + .36\times.005\times100 = 3.54$
(as compared to 5.44 with no L2\$)

Multilevel Cache Design Considerations

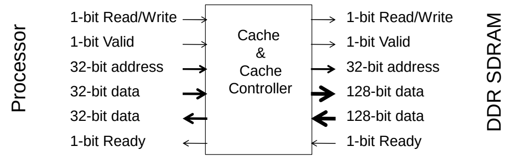
- Design considerations for L1 and L2 caches are very different
  - Primary cache should focus on minimizing hit time in support of a shorter clock cycle
    - Smaller with smaller block sizes
  - Secondary cache(s) should focus on reducing miss rate to reduce the penalty of long main memory access times
    - Larger with larger block sizes
    - Higher levels of associativity
- The miss penalty of the L1 cache is significantly reduced by the presence of an L2 cache – so it can be smaller (i.e., faster) but have a higher miss rate
- For the L2 cache, hit time is less important than miss rate
  - The L2\$ hit time determines L1\$'s miss penalty
  - L2\$ local miss rate >> than the global miss rate

Two Machines' Cache Parameters

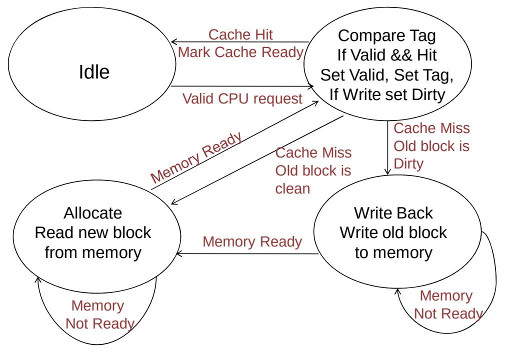
	Intel Nehalem	AMD Barcelona
L1 cache organization & size	Split I\$ and D\$; 32KB for each per core; 64B blocks	Split I\$ and D\$; 64KB for each per core; 64B blocks
L1 associativity	4-way (I), 8-way (D) set assoc.; ~LRU replacement	2-way set assoc.; LRU replacement
L1 write policy	write-back, write-allocate	write-back, write-allocate
L2 cache organization & size	Unified; 256KB (0.25MB) per core; 64B blocks	Unified; 512KB (0.5MB) per core; 64B blocks
L2 associativity	8-way set assoc.; ~LRU	16-way set assoc.; ~LRU
L2 write policy	write-back	write-back
L2 write policy	write-back, write-allocate	write-back, write-allocate
L3 cache organization & size	Unified; 8192KB (8MB) shared by cores; 64B blocks	Unified; 2048KB (2MB) shared by cores; 64B blocks
L3 associativity	16-way set assoc.	32-way set assoc.; evict block shared by fewest cores
L3 write policy	write-back, write-allocate	write-back; write-allocate

FSM Cache Controller

- Key characteristics for a simple L1 cache
  - Direct mapped
  - Write-back using write-allocate
  - Block size of 4 32-bit words (so 16B); Cache size of 16KB (so 1024 blocks)
  - 18-bit tags, 10-bit index, 2-bit block offset, 2-bit byte offset, dirty bit, valid bit, LRU bits (if set associative)

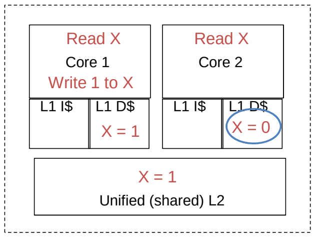


Four State Cache Controller



Cache Coherence in Multicores

- In multicore processors the cores *share* a common physical address space, causing a cache coherence problem



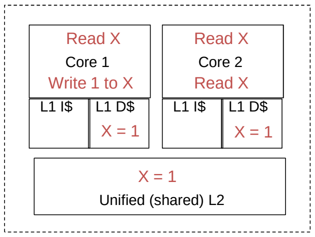
A Coherent Memory System

- Any read of a data item should return the most recently written value of the data item
  - Coherence – defines what values can be returned by a read
    - Writes to the same location are serialized (two writes to the same location must be seen in the same order by all cores)
  - Consistency – determines when a written value will be returned by a read
- To enforce coherence, caches must provide
  - Replication of shared data items in multiple cores' caches
    - Replication reduces both latency and contention for a read shared data item
  - Migration of shared data items to a core's local cache
    - Migration reduces the latency of the access to the data and the bandwidth demand on the shared memory (L2 in our example)

Cache Coherence Protocols

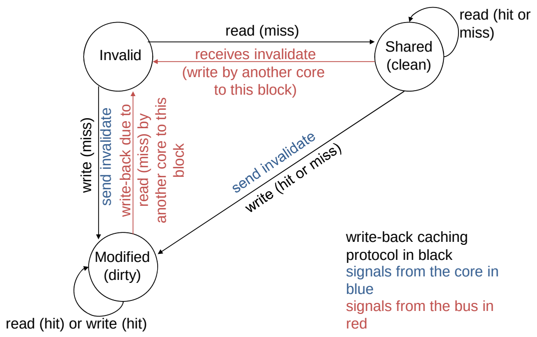
- Need a hardware protocol to ensure cache coherence. The most popular of which is snooping
  - The cache controllers monitor (snoop) on the broadcast medium (e.g., bus) with duplicate address tag hardware (so they don't interfere with core's access to the cache) to determine if their cache has a copy of a block that is requested
- Write invalidate protocol – writes require exclusive access and invalidate *all* other copies
  - Exclusive access ensures that no other readable or writable copies of an item exists
- If two processors attempt to write the same data at the same time, one of them wins the race causing the other core's copy to be invalidated. For the other core to complete, it must obtain a new copy of the data which must now contain the updated value – thus enforcing write serialization

Example of Snooping Invalidation



- When the second miss by Core 2 occurs, Core 1 responds with the value canceling the response from the L2 cache (and also updating the L2 copy)

A Write-Invalidate CC Protocol

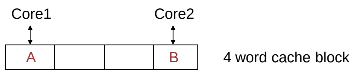


Invalid
Shared (clean)
Modified (dirty)
read (miss)
read (hit or miss)
receives invalidate (write by another core to this block)
write (miss)
send invalidate
write-back due to read (miss) by another core to this block
send invalidate
write (hit or miss)
read (hit) or write (hit)

write-back caching protocol in black
signals from the core in blue
signals from the bus in red

Block Size Effects

- Writes to one word in a multi-word block mean that the full block is invalidated
- Multi-word blocks can also result in false sharing: when two cores are writing to two different variables that happen to fall in the same cache block
  - With write-invalidate false sharing increases cache miss rates



- Compilers can help reduce false sharing by allocating highly correlated data to the same cache block

Summary: Improving Cache Performance

0. Reduce the time to hit in the cache
   - smaller cache
   - direct mapped cache
   - smaller blocks
   - for writes
     - no write allocate – no “hit” on cache, just write to write buffer
     - write allocate – to avoid two cycles (first check for hit, then write) pipeline writes via a delayed write buffer to cache

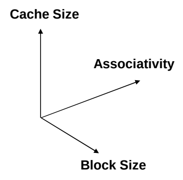
1. Reduce the miss rate
   - bigger cache
   - more flexible placement (increase associativity)
   - larger blocks (16 to 64 bytes typical)
   - victim cache – small buffer holding most recently discarded blocks

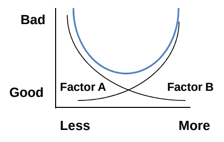
Summary: Improving Cache Performance

2. Reduce the miss penalty
   - smaller blocks
   - use a write buffer to hold dirty blocks being replaced so don’t have to wait for the write to complete before reading
   - check write buffer (and/or victim cache) on read miss – may get lucky
   - for large blocks fetch critical word first
   - use multiple cache levels – L2 cache not tied to CPU clock rate
   - faster backing store/improved memory bandwidth
     - wider buses
     - memory interleaving, DDR SDRAMs

Summary: The Cache Design Space

- Several interacting dimensions
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back
  - write allocation
- The optimal choice is a compromise
  - depends on access characteristics
    - workload
    - use (I-cache, D-cache, TLB)
  - depends on technology / cost
- Simplicity often wins





End of Lecture