CS 61C:
Great Ideas in Computer Architecture
Cache Performance

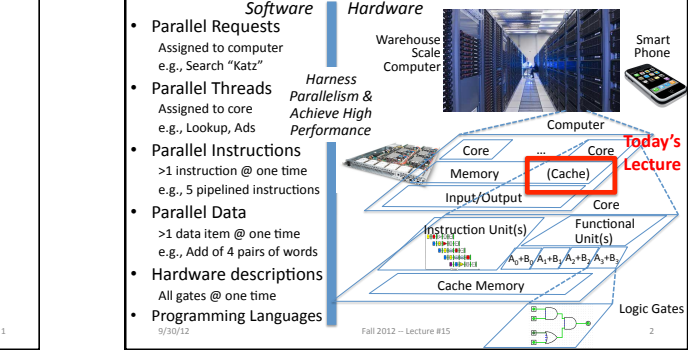
Instructors:
Krste Asanovic, Randy H. Katz
http://inst.eecs.Berkeley.edu/~cs61c/fa12

New-School Machine Structures (It's a bit more complicated!)

Software | *Hardware*

- Parallel Requests
  Assigned to computer
  e.g., Search "Katz"
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions
  All gates @ one time
- Programming Languages

Harness Parallelism & Achieve High Performance

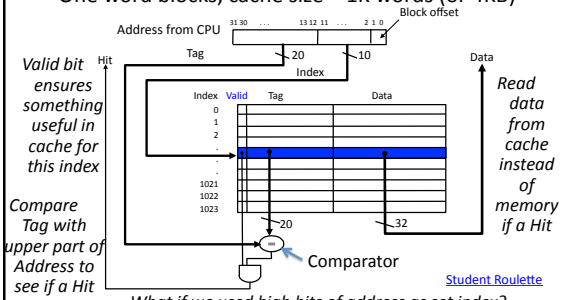


Review

- Memory hierarchy exploits temporal and spatial locality in instruction and data memory references from applications
- Almost as fast as small, expensive memory, while having capacity of large, cheap memory.
- Cache is hardware-managed, programmer-invisible structure to hold copies of recently-used memory locations
  - Cache hits serviced quickly
  - Cache misses need to go to memory, SLOW!

Review: Direct-Mapped Cache

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



Student Roulette

What if we used high bits of address as set index?

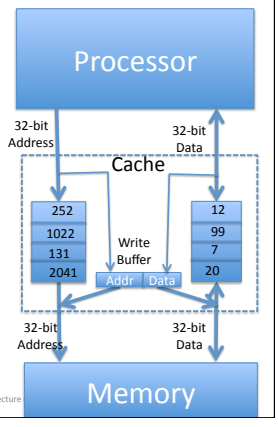
Handling Stores with Write-Through

- Store instructions write to memory, changing values
- Need to make sure cache and memory have same values on writes: 2 policies

1) Write-Through Policy: write cache and write *through* the cache to memory
   - Every write eventually gets to memory
   - Too slow, so include Write Buffer to allow processor to continue once data in Buffer
   - Buffer updates memory in parallel to processor

Write-Through Cache

- Write both values in cache and in memory
- Write buffer stops CPU from stalling if memory cannot keep up
- Write buffer may have multiple entries to absorb bursts of writes
- What if store misses in cache?

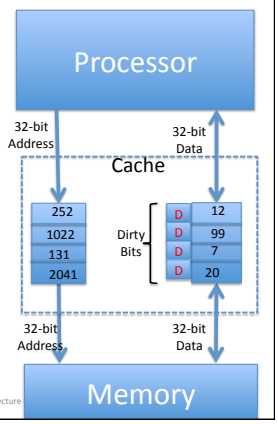


Handling Stores with Write-Back

2) Write-Back Policy: write only to cache and then write cache block *back* to memory when evict block from cache
  - Writes collected in cache, only single write to memory per block
  - Include bit to see if wrote to block or not, and then only write back if bit is set
    - Called "Dirty" bit (writing makes it "dirty")

Write-Back Cache

- Store/cache hit, write data in cache *only* & set dirty bit
  - Memory has stale value
- Store/cache miss, read data from memory, then update and set dirty bit
  - "Write-allocate" policy
- Load/cache hit, use value from cache
- On any miss, write back evicted block, only if dirty. Update cache with new block and clear dirty bit.



Write-Through vs. Write-Back

- Write-Through:
  - Simpler control logic
  - More predictable timing simplifies processor control logic
  - Easier to make reliable, since memory always has copy of data
- Write-Back
  - More complex control logic
  - More variable timing (0,1,2 memory accesses per cache access)
  - Usually reduces write traffic
  - Harder to make reliable, sometimes cache has only copy of data

Average Memory Access Time (AMAT)

- Average Memory Access Time (AMAT) is the average to access memory considering both hits and misses in the cache

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

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

Given a 200 psec clock, a miss penalty of 50 clock cycles, a miss rate of 0.02 misses per instruction and a cache hit time of 1 clock cycle, what is AMAT?

- ☐ ≤200 psec
- ☐ 400 psec
- ☐ 600 psec
- ☐ ≥ 800 psec

Average Memory Access Time (AMAT)

- 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 200 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?

1 + 0.02 x 50 = 2 clock cycles
Or 2 x 200 = 400 psecs

Average Memory Access Time (AMAT)

- 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

- How calculate if separate instruction and data caches?

Impact of Cache on CPI

- Assume cache hit time included in normal CPU execution time, then
  CPU time = Instruction Count (IC) × Cycles Per Instruction (CPI) × Cycle Time (CT)
  = IC × (CPI_{ideal} + CPI_{miss}) × CT
  (CPI_{stalls})
- A simple model for cache miss impact on CPI

CPI_{miss} = accesses/instruction × miss rate × miss penalty

Impacts of Cache Performance

- Relative $ penalty increases as processor performance improves (faster clock rate and/or lower CPI)
  - When calculating CPI_{stalls}, cache miss penalty is measured in processor clock cycles needed to handle a miss
  - Lower the CPI_{ideal}, more pronounced impact of stalls
- 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
  - CPI_{miss} = 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? Student Roulette
- What if the D$ miss rate went up by 1%?

Impact of larger cache on AMAT?

- 1) Lower Miss rate
- 2) Longer Access time (Hit time): smaller is faster
  - Increase in hit time will likely add another stage to the pipeline
- At some point, increase in hit time for a larger cache may overcome the improvement in hit rate, yielding a decrease in performance
- Computer architects expend considerable effort optimizing organization of cache hierarchy – big impact on performance and power!

Administrivia

- Lab #5: MIPS Assembly
- HW #4 (of six), due Sunday
- Project 2a: MIPS Emulator, due Sunday
- Midterm, a week from Tuesday

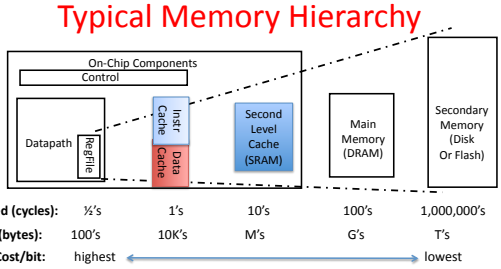
How to Reduce Miss Penalty?

- Could there be locality on misses from a cache?
- Use multiple cache levels!
- With Moore's Law, more room on die for bigger L1 caches and for second-level (L2) cache
- And in some cases even an L3 cache!
- IBM mainframes have ~1GB L4 cache off-chip.

Multiple Cache Levels

- E.g., CPI_{ideal} of 2,
  100 cycle miss penalty (to main memory),
  25 cycle miss penalty (to L2$),
  36% load/stores,
  a 2% (4%) L1 I$ (D$) miss rate,
  add a 0.5% L2$ miss rate
  - CPI_{stalls} = 2 + 0.02×25 + 0.36×0.04×25 + 0.005×100 + 0.36×0.005×100
    = 3.54 (vs. 5.44 with no L2$)

Typical Memory Hierarchy



- Principle of locality + memory hierarchy presents programmer with ≈ as much memory as is available in the *cheapest* technology at the ≈ speed offered by the *fastest* technology

Local vs. Global Miss Rates

- *Local miss rate* – the fraction of references to one level of a cache that miss
- Local Miss rate L2$ = $L2 Misses / L1$ Misses
- *Global miss rate* – the fraction of references that miss in all levels of a multilevel cache
  - L2$ local miss rate >> than the global miss rate
  - Often as high as 50% local miss rate – still useful?

For L1 cache

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

What is AMAT for system with L1 and L2 cache (L2 miss rate is *local* miss rate)?

- ☐ Time for L2 hit + L2 Miss rate x L2 Miss penalty
- ☐ Time for L1 hit + L1 Miss rate x L2 Miss rate x Miss penalty
- ☐ Time for L1 hit + L1 Miss rate x (Time for L2 hit + L2 Miss rate x Miss Penalty)
- ☐ **Time for L1 hit + L1 Miss rate x Miss penalty + Time for L2 hit + L2 Miss rate x Miss penalty**

Local vs. Global Miss Rates

- *Local miss rate* – the fraction of references to one level of a cache that miss
- Local Miss rate L2$ = $L2 Misses / L1$ Misses
- *Global miss rate* – the fraction of references that miss in all levels of a multilevel cache
  - L2$ local miss rate >> than the global miss rate
- Global Miss rate = L2$ Misses / Total Accesses
  = L2$ Misses / L1$ Misses x L1$ Misses / Total Accesses
  = Local Miss rate L2$ x Local Miss rate L1$
- AMAT = Time for a hit + Miss rate x Miss penalty
- AMAT = Time for a L1$ hit + (local) Miss rateL1$ x (Time for a L2$ hit + (local) Miss rate L2$ x L2$ Miss penalty)

Improving Cache Performance (1 of 3)

AMAT = Hit Time + Miss rate x Miss penalty

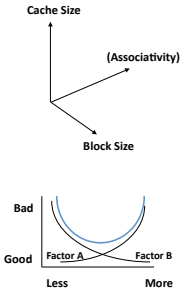
1. Reduce the time to hit in the cache
   - Smaller cache
2. Reduce the miss rate
   - Bigger cache
   - Larger blocks (16 to 64 bytes typical)
   - (Later in semester: More flexible placement by increasing associativity)

Improving Cache Performance (2 of 3)

3. Reduce the miss penalty
   - Smaller blocks
   - Use multiple cache levels
     - L2 cache size not tied to processor clock rate
   - Higher DRAM memory bandwidth (faster DRAMs)
   - Use a write buffer to hold dirty blocks being replaced so don't have to wait for the write to complete before reading

The Cache Design Space (3 of 3)

- Several interacting dimensions
  - Cache size
  - Block size
  - Write-through vs. write-back
  - Write allocation
  - (Later Associativity)
  - (Later Replacement policy)
- Optimal choice is a compromise
  - Depends on access characteristics
    - Workload
    - Use (I-cache, D-cache)
  - Depends on technology / cost
- Simplicity often wins



Multilevel Cache Design Considerations

- Different design considerations for L1$ and L2$
  - L1$ focuses on minimizing hit time for shorter clock cycle: Smaller $ with smaller block sizes
  - L2$(s) focus on reducing miss rate to reduce penalty of long main memory access times: Larger $ with larger block sizes
- Miss penalty of L1$ is significantly reduced by presence of L2$, so can be smaller/faster but with higher miss rate
- For the L2$, hit time is less important than miss rate
  - L2$ hit time determines L1$'s miss penalty

Characteristic	Intel Nehalem	AMD Opteron X4 (Barcelona)
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KB each for instructions/data per core	64 KB each for instructions/data per core
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L1 hit time (load-use)	Not Available	3 clock cycles
L2 cache organization	Unified (instruction and data) per core	Unified (instruction and data) per core
L2 cache size	256 KB (0.25 MB)	512 KB (0.5 MB)
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L2 hit time	Not Available	9 clock cycles
L3 cache organization	Unified (instruction and data)	Unified (instruction and data)
L3 cache size	8192 KB (8 MB), shared	2048 KB (2 MB), shared
L3 block size	64 bytes	64 bytes
L3 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L3 hit time	Not Available	38 (?)clock cycles

CPI/Miss Rates/DRAM Access SpecInt2006

		Data Only	Data Only	Instructions and Data
Name	CPI	L1 D cache misses/1000 instr	L2 D cache misses/1000 instr	DRAM accesses/1000 instr
perl	0.75	3.5	1.1	1.3
bzip2	0.85	11.0	5.8	2.5
gcc	1.72	24.3	13.4	14.8
mcf	10.00	106.8	88.0	88.5
go	1.09	4.5	1.4	1.7
hmmer	0.80	4.4	2.5	0.6
sjeng	0.96	1.9	0.6	0.8
libquantum	1.61	33.0	33.1	47.7
h264avc	0.80	8.8	1.6	0.2
omnetpp	2.94	30.9	27.7	29.8
astar	1.79	16.3	9.2	8.2
xalancbmk	2.70	38.0	15.8	11.4
Median	1.35	13.6	7.5	5.4

….and in Conclusion

- Write-through versus write-back caches
- Larger caches reduce Miss rate via Temporal and Spatial Locality, but can increase Hit time
- AMAT helps balance Hit time, Miss rate, Miss penalty
- Multilevel caches help Miss penalty