Memory Hierarchy Basics

- Six basic cache optimizations:
  - Larger block size
    - Reduces compulsory misses
    - Increases capacity and conflict misses, increases miss penalty
  - Larger total cache capacity to reduce miss rate
    - Increases hit time, increases power consumption
  - Higher associativity
    - Reduces conflict misses
    - Increases hit time, increases power consumption
  - Higher number of cache levels
    - Reduces overall memory access time
  - Giving priority to read misses over writes
    - Reduces miss penalty
  - Avoiding address translation in cache indexing
    - Reduces hit time
  - Victim cache

Copyright © 2012, Elsevier Inc. All rights reserved.

Ten Advanced Optimizations

- Small and simple first level caches
- Way Prediction
- Pipelined caches
- Non-blocking cache
- Multibanked cache
- Critical word first
- Merging write buffer
- Compiler optimization
- Hardware prefetching
- Compiler prefetching

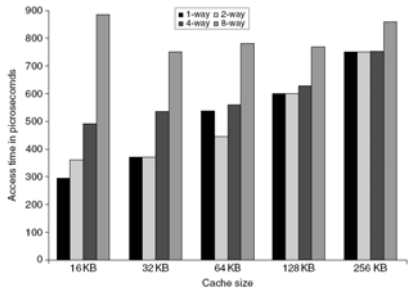
Copyright © 2012, Elsevier Inc. All rights reserved.

Small and Simple

- No mux in the critical path of a direct mapped cache.
- Bigger cache means more energy.
- CACTI – An idea for the project/paper review
- Many processors takes at least 2 clock cycles to access the cache, longer hit time may not be that critical
- The use of a virtual index cache, limits the cache size to page size × associativity (recently a trend to increase associativity).

Copyright © 2012, Elsevier Inc. All rights reserved.

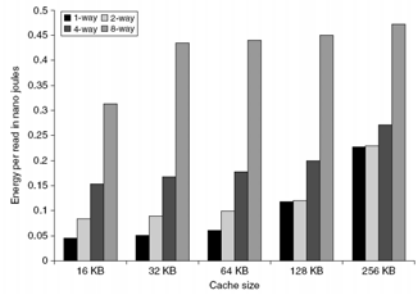
L1 Size and Associativity



Access time vs. size and associativity

Copyright © 2012, Elsevier Inc. All rights reserved.

L1 Size and Associativity



Energy per read vs. size and associativity

Copyright © 2012, Elsevier Inc. All rights reserved.

Way Prediction

- To improve hit time, predict the way to pre-set mux
  - Mis-prediction gives longer hit time
  - Prediction accuracy
    - > 90% for two-way
    - > 80% for four-way
    - I-cache has better accuracy than D-cache
  - First used on MIPS R10000 in mid-90s
  - Used on ARM Cortex-A8
- Extend to predict block as well
  - "Way selection"
  - Increases mis-prediction penalty

Copyright © 2012, Elsevier Inc. All rights reserved.

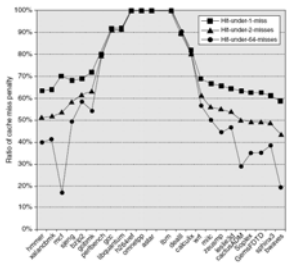
Pipelining Cache

- Pipeline cache access to improve bandwidth
  - Examples:
    - Pentium: 1 cycle
    - Pentium Pro – Pentium III: 2 cycles
    - Pentium 4 – Core i7: 4 cycles
- Increases branch miss-prediction penalty (longer pipeline).
- Makes it easier to increase associativity

Copyright © 2012, Elsevier Inc. All rights reserved.

Nonblocking Caches

- For out-of-order execution (later on this point).
- Allow hits before previous misses complete
  - "Hit under miss"
  - "Hit under multiple miss"
- L2 must support this
- In general, processors can hide L1 miss penalty but not L2 miss penalty



Single core i7 using SPEC2006

Copyright © 2012, Elsevier Inc. All rights reserved.

Multibanked Caches

- Organize cache as independent banks to support simultaneous access
  - ARM Cortex-A8 supports 1-4 banks for L2
  - Intel i7 supports 4 banks for L1 and 8 banks for L2
- Interleave banks according to block address

Block address	Bank 0	Block address	Bank 1	Block address	Bank 2	Block address	Bank 3
0		1		2		3	
4		5		6		7	
8		9		10		11	
12		13		14		15	

Figure 2.6 Four-way interleaved cache banks using block addressing. Assuming 64 bytes per blocks, each of these addresses would be multiplied by 64 to get byte addressing.

Copyright © 2012, Elsevier Inc. All rights reserved.

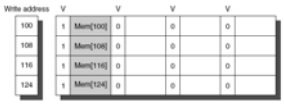
Critical Word First, Early Restart

- Critical word first
  - Request missed word from memory first
  - Send it to the processor as soon as it arrives
- Early restart
  - Request words in normal order
  - Send missed work to the processor as soon as it arrives
- Effectiveness of these strategies depends on block size and likelihood of another access to the portion of the block that has not yet been fetched

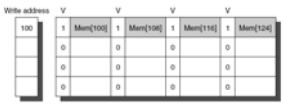
Copyright © 2012, Elsevier Inc. All rights reserved.

Merging Write Buffer

- When storing to a block that is already pending in the write buffer, update write buffer
- Reduces stalls due to full write buffer
- Do not apply to I/O addresses



No write buffering



Write buffering

Copyright © 2012, Elsevier Inc. All rights reserved.

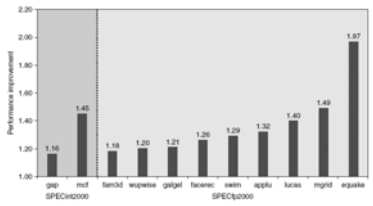
Compiler Optimizations

- Loop Interchange
  - Swap nested loops to access memory in sequential order (row major access)
- Blocking
  - Instead of accessing entire rows or columns, subdivide matrices into blocks
  - Requires more memory accesses but improves locality of accesses

Copyright © 2012, Elsevier Inc. All rights reserved.

Hardware Prefetching

- Fetch two blocks on miss (include next sequential block) (the 2nd one goes to instruction stream buffer, must be checked if found do not go to cache).



Pentium 4 Pre-fetching

Copyright © 2012, Elsevier Inc. All rights reserved.

Compiler Prefetching

- Insert prefetch instructions before data is needed
- Non-faulting: prefetch doesn't cause exceptions
- Register prefetch
  - Loads data into register
- Cache prefetch
  - Loads data into cache
- Combine with loop unrolling and software pipelining

Copyright © 2012, Elsevier Inc. All rights reserved.

Summary

Technique	Hit time	Band-width	Miss penalty	Miss rate	Power consumption	Hardware cost/ complexity	Comment
Small and simple caches	+			–	+	0	Trivial; widely used
Way-predicting caches	+				+	1	Used in Pentium 4
Pipelined cache access	–	+				1	Widely used
Nonblocking caches		+	+			3	Widely used
Banked caches		+			+	1	Used in L2 of both i7 and Cortex-A8
Critical word first and early restart			+			2	Widely used
Merging write buffer			+			1	Widely used with write through
Compiler techniques to reduce cache misses				+		0	Software is a challenge, but many compilers handle common linear algebra calculations
Hardware prefetching of instructions and data			+	+	–	2 instr., 3 data	Most provide prefetch instructions; modern high-end processors also automatically prefetch in hardware.
Compiler-controlled prefetching			+	+		3	Needs nonblocking cache; possible instruction overhead; in many CPUs

Figure 2.11 Summary of 10 advanced cache optimizations showing impact on cache performance, power consumption, and complexity. Although generally a technique helps only one factor, prefetching can reduce misses if done sufficiently early; if not, it can reduce miss penalty. + means that the technique improves the factor, – means it hurts that factor, and blank means it has no impact. The complexity measure is subjective, with 0 being the easiest and 3 being a challenge.

Copyright © 2012, Elsevier Inc. All rights reserved.