Computer Architecture
A Quantitative Approach, Sixth Edition

Chapter 2

Memory Hierarchy Design
Part II Performance

Copyright © 2019, Elsevier Inc. All rights Reserved 1

Memory Technology

Performance

- CPUtime = Instruction count x CPI x Clock cycle time
- CPIexecution = CPI with ideal memory
- CPI = CPIexecution + Mem Stall cycles per instruction
- Mem Stall cycles per instruction =
  Mem accesses per instruction x Miss rate x Miss penalty
- CPUtime = Instruction Count x (CPIexecution + Mem Stall cycles per instruction) x Clock cycle time
- CPUtime = IC x (CPIexecution + Mem accesses per instruction x Miss rate x Miss penalty) x Clock cycle time
- Misses per instruction = Memory accesses per instruction x Miss rate
- CPUtime = IC x (CPIexecution + Misses per instruction x Miss penalty) x Clock cycle time

Copyright © 2019, Elsevier Inc. All rights Reserved 2

Write Policy

Write Policy

1. Write Though: Data is written to both the cache block and to a block of main memory.
   - The lower level always has the most updated data; an important feature for I/O and multiprocessing.
   - Easier to implement than write back.
   - A write buffer is often used to reduce CPU write stall while data is written to memory.
2. Write back: Data is written or updated only to the cache block. The modified or dirty cache block is written to main memory when it's being replaced from cache.
   - Writes occur at the speed of cache
   - A status bit called a dirty or modified bit, is used to indicate whether the block was modified while in cache; if not the block is not written back to main memory when replaced.
   - Uses less memory bandwidth than write through.

3

Write Policy

Write Policy

Write Allocate:

The cache block is loaded on a write miss followed by write hit actions.

No-Write Allocate:

The block is modified in the lower level (lower cache level, or main memory) and not loaded into cache.

MK 4

Reploacement

Replacement

- Which block to replace if a cache miss?
  - Random
  - FIFO
  - LRU
  - Not MRU
  - Optimal ??
  - Least frequently used
- The cost of the algorithm (logic and number of bits)

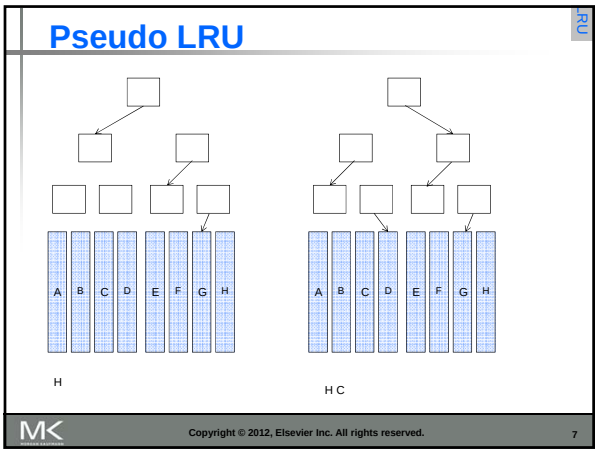
MK 5

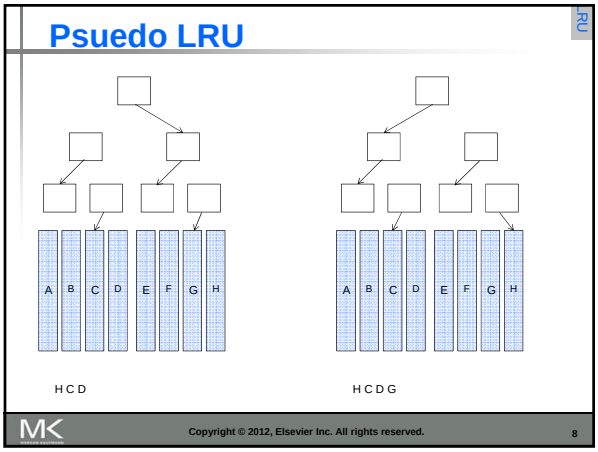
LRU

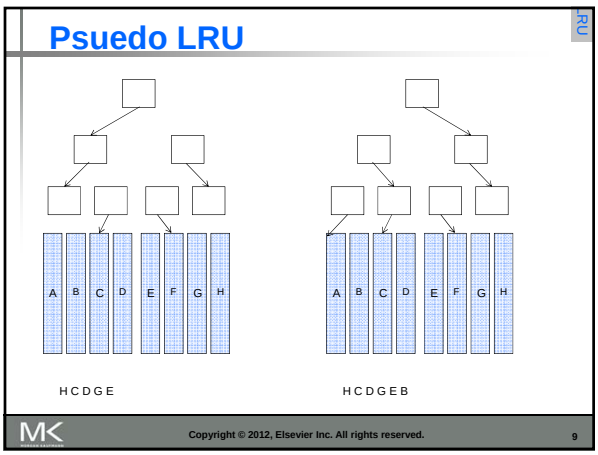
LRU

- A list to keep track of the order of access to every block in the set.
- The least recently used block is replaced (if needed).
- How many bits we need for that?

MK Copyright © 2012, Elsevier Inc. All rights reserved. 6

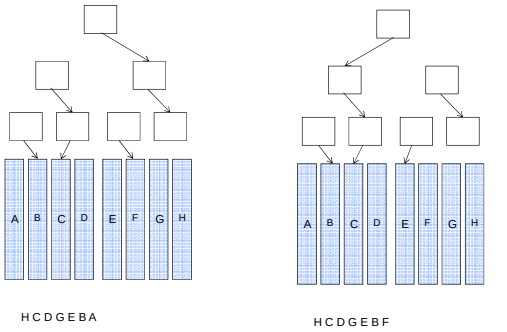

Pseudo LRU
LRU
A B C D E F G H
H
A B C D E F G H
HC
Copyright © 2012, Elsevier Inc. All rights reserved.
7


Psuedo LRU
LRU
A B C D E F G H
HCD
A B C D E F G H
HCDG
Copyright © 2012, Elsevier Inc. All rights reserved.
8


Psuedo LRU
LRU
A B C D E F G H
HCDGE
A B C D E F G H
HCDGEB
Copyright © 2012, Elsevier Inc. All rights reserved.
9

Psuedo LRU

LRU



Copyright © 2012, Elsevier Inc. All rights reserved.

10

Cache Performance

Performance

- Assuming the following execution and cache parameters:
  - Cache miss penalty = 50 cycles
  - Normal instruction execution CPI ignoring memory stalls = 2.0 cycles
  - Miss rate = 2%
  - Average memory references/instruction = 1.33
- CPU time = IC × [CPI execution + Memory accesses/instruction × Miss rate × Miss penalty] × Clock cycle time
- CPUtime with cache = IC x (2.0 + (1.33 x 2% x 50)) x clock cycle time
- = IC x 3.33 x Clock cycle time
- *Lower CPI execution increases the impact of cache miss clock cycles*

11

Cache Performance

Performance

- Suppose a CPU executes at Clock Rate = 200 MHz (5 ns per cycle) with a single level of cache.
- CPIexecution = 1.1
- Instruction mix: 50% arith/logic, 30% load/store, 20% control
- Assume a cache miss rate of 1.5% and a miss penalty of 50 cycles.
- CPI = $CPI_{execution}$ + mem stalls per instruction
- Mem Stalls per instruction =
- Mem accesses per instruction x Miss rate x Miss penalty
- Mem accesses per instruction = 1 + .3 = 1.3
- Mem Stalls per instruction = 1.3 x .015 x 50 = 0.975
- CPI = 1.1 + .975 = 2.075
- The ideal memory CPU with no misses is 2.075/1.1 = 1.88 times faster

12

Cache Performance

Performance

- Suppose for the previous example we double the clock rate to 400 MHZ, how much faster is this machine, assuming similar miss rate, instruction mix?
- Since memory speed is not changed, the miss penalty takes more CPU cycles:
- Miss penalty = 50 x 2 = 100 cycles.
- CPI = 1.1 + 1.3 x .015 x **100** = 1.1 + 1.95 = 3.05
- Speedup = (CPIold x Cold)/ (CPInew x Cnew)
- = 2.075 x 2 / 3.05 = 1.36
- The new machine is only 1.36 times faster rather than 2

times faster due to the increased effect of cache misses.

- *CPUs with higher clock rate, have more cycles per cache miss and more memory impact on CPI.*

MK 13

Cache Performance

Performance

- Suppose a CPU uses separate level one (L1) caches for instructions and data (Harvard memory architecture) with different miss rates for instruction and data access:
  - $CPI_{execution}$ = 1.1
  - Instruction mix: 50% arith/logic, 30% load/store, 20% control
  - Assume a cache miss rate of 0.5% for instruction fetch and a cache data miss rate of 6%.
  - A cache hit incurs no stall cycles while a cache miss incurs 200 stall cycles for both memory reads and writes. Find the resulting CPI using this cache? How much faster is the CPU with ideal memory?

CPI = $CPI_{execution}$ + mem stalls per instruction

Mem Stall cycles per instruction = Instruction Fetch Miss rate x Miss Penalty + Data Memory Accesses Per Instruction x Data Miss Rate x Miss Penalty

Mem Stall cycles per instruction = 1 x 0.5/100 x 200 + 0.3 x 6/100 x 200 = 1 + 3.6 = 4.6

CPI = $CPI_{execution}$ + mem stalls per instruction = 1.1 + 4.6 = 5.7

The CPU with ideal cache (no misses) is 5.7/1.1 = 5.18 times faster

With no cache the CPI would have been = 1.1 + 1.3 X 200 = 261.1

MK 14

Memory Hierarchy Basics

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

MK Copyright © 2012, Elsevier Inc. All rights reserved. 15

Ten (11) Advanced Optimizations

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
- Sectored cache

Copyright © 2012, Elsevier Inc. All rights reserved. 16

Small and Simple

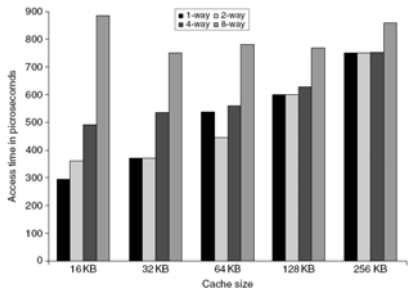
Advanced Optimizations

- 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. 17

L1 Size and Associativity

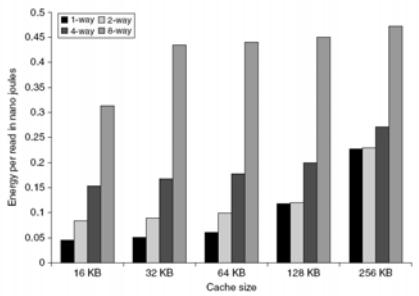
Advanced Optimizations



Access time vs. size and associativity

Copyright © 2012, Elsevier Inc. All rights reserved. 18

L1 Size and Associativity



Energy per read vs. size and associativity

Copyright © 2012, Elsevier Inc. All rights reserved. 19

Way Prediction

- To improve hit time, predict the way to pre-set mux
  - Misprediction 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

Copyright © 2012, Elsevier Inc. All rights reserved. 20

Pipelining Cache

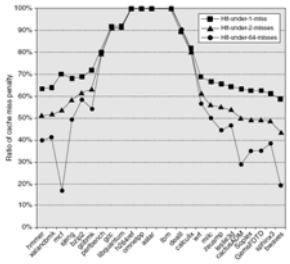
- Pipeline cache access to improve bandwidth
  - Examples:
    - Pentium: 1 cycle
    - Pentium Pro – Pentium III: 2 cycles
    - Pentium 4 – Core i7: 4 cycles
- Increases branch miss-prediction penalty (longer pipeline).
- Makes it easier to increase associativity

Copyright © 2012, Elsevier Inc. All rights reserved. 21

Nonblocking Caches

Advanced Optimizations

- 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. 22

Multibanked Caches

Advanced Optimizations

- 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

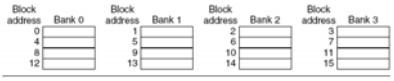


Figure 2.6 Four-way interleaved cache banks using block addressing. Assuming 64 bytes per blocks, each of these addresses would be multiplied by 64 to get byte addressing.

Copyright © 2012, Elsevier Inc. All rights reserved. 23

Critical Word First, Early Restart

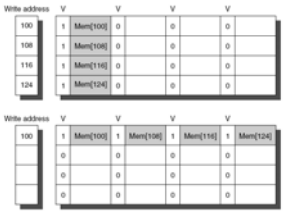
Advanced Optimizations

- 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. 24

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.

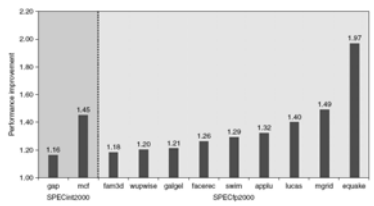
Compiler Optimizations

- 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. 28

Sectored Cache

- Divide the block into subblocks
- Ach subblock has its own V and D bits
- How is that helpful?

Copyright © 2012, Elsevier Inc. All rights reserved. 29

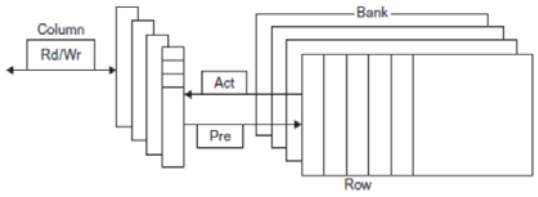
Summary

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. 30

Internal Organization of DRAM



Copyright © 2019, Elsevier Inc. All rights Reserved

Memory Technology

- Amdahl:
  - Memory capacity should grow linearly with processor speed
  - Unfortunately, memory capacity and speed has not kept pace with processors
- Some optimizations:
  - Multiple accesses to same row
  - Synchronous DRAM
    - Added clock to DRAM interface
    - Burst mode with critical word first
  - Wider interfaces
  - Double data rate (DDR)
  - Multiple banks on each DRAM device

Copyright © 2019, Elsevier Inc. All rights Reserved

Memory Optimizations

			Best case access time (no precharge)			Precharge needed
Production year	Chip size	DRAM type	RAS time (ns)	CAS time (ns)	Total (ns)	Total (ns)
2000	256M bit	DDR1	21	21	42	63
2002	512M bit	DDR1	15	15	30	45
2004	1G bit	DDR2	15	15	30	45
2006	2G bit	DDR2	10	10	20	30
2010	4G bit	DDR3	13	13	26	39
2016	8G bit	DDR4	13	13	26	39

Copyright © 2019, Elsevier Inc. All rights Reserved

Memory Optimizations

Standard	I/O clock rate	M transfers/s	DRAM name	MiB/s/DIMM	DIMM name
DDR1	133	266	DDR266	2128	PC2100
DDR1	150	300	DDR300	2400	PC2400
DDR1	200	400	DDR400	3200	PC3200
DDR2	266	533	DDR2-533	4264	PC4300
DDR2	333	667	DDR2-667	5336	PC5300
DDR2	400	800	DDR2-800	6400	PC6400
DDR3	533	1066	DDR3-1066	8528	PC8500
DDR3	666	1333	DDR3-1333	10,664	PC10700
DDR3	800	1600	DDR3-1600	12,800	PC12800
DDR4	1333	2666	DDR4-2666	21,300	PC21300

Copyright © 2019, Elsevier Inc. All rights Reserved

Memory Optimizations

- DDR:
  - DDR2
    - Lower power (2.5 V -> 1.8 V)
    - Higher clock rates (266 MHz, 333 MHz, 400 MHz)
  - DDR3
    - 1.5 V
    - 800 MHz
  - DDR4
    - 1-1.2 V
    - 1333 MHz
- GDDR5 is graphics memory based on DDR3

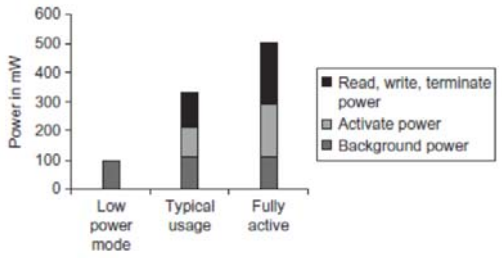
Copyright © 2019, Elsevier Inc. All rights Reserved

Memory Optimizations

- Reducing power in SDRAMs:
  - Lower voltage
  - Low power mode (ignores clock, continues to refresh)
- Graphics memory:
  - Achieve 2-5 X bandwidth per DRAM vs. DDR3
    - Wider interfaces (32 vs. 16 bit)
    - Higher clock rate
      - Possible because they are attached via soldering instead of socketted DIMM modules

Copyright © 2019, Elsevier Inc. All rights Reserved

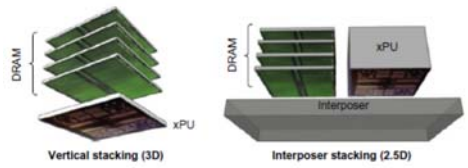
Memory Power Consumption



MK
Copyright © 2019, Elsevier Inc. All rights Reserved

Stacked/Embedded DRAMs

- Stacked DRAMs in same package as processor
  - High Bandwidth Memory (HBM)



MK
Copyright © 2019, Elsevier Inc. All rights Reserved

Flash Memory

- Type of EEPROM
- Types: NAND (denser) and NOR (faster)
- NAND Flash:
  - Reads are sequential, reads entire page (.5 to 4 KiB)
  - 25 us for first byte, 40 MiB/s for subsequent bytes
  - SDRAM: 40 ns for first byte, 4.8 GB/s for subsequent bytes
  - 2 KiB transfer: 75 uS vs 500 ns for SDRAM, 150X slower
  - 300 to 500X faster than magnetic disk

MK
Copyright © 2019, Elsevier Inc. All rights Reserved

NAND Flash Memory

- Must be erased (in blocks) before being overwritten
- Nonvolatile, can use as little as zero power
- Limited number of write cycles (~100,000)
- $2/GiB, compared to $20-40/GiB for SDRAM and $0.09 GiB for magnetic disk
- Phase-Change/Memrister Memory
  - Possibly 10X improvement in write performance and 2X improvement in read performance

Memory Technology and Optimizations

MK Copyright © 2019, Elsevier Inc. All rights Reserved 40

Memory Dependability

- Memory is susceptible to cosmic rays
- *Soft errors*: dynamic errors
  - Detected and fixed by error correcting codes (ECC)
- *Hard errors*: permanent errors
  - Use spare rows to replace defective rows
- Chipkill: a RAID-like error recovery technique

Memory Technology and Optimizations

MK Copyright © 2019, Elsevier Inc. All rights Reserved 41

Advanced Optimizations

- Reduce hit time
  - Small and simple first-level caches
  - Way prediction
- Increase bandwidth
  - Pipelined caches, multibanked caches, non-blocking caches
- Reduce miss penalty
  - Critical word first, merging write buffers
- Reduce miss rate
  - Compiler optimizations
- Reduce miss penalty or miss rate via parallelization
  - Hardware or compiler prefetching

Advanced Optimizations

MK Copyright © 2019, Elsevier Inc. All rights Reserved 42