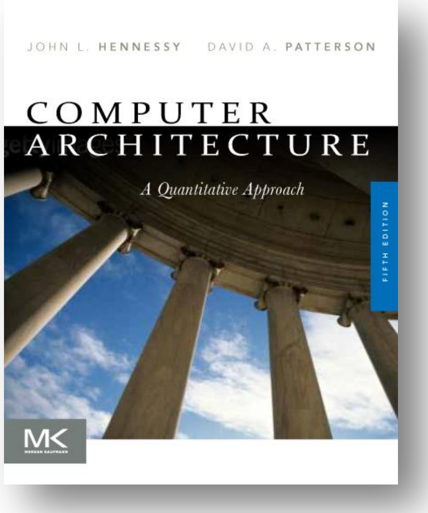


Chapter 2

Memory Hierarchy Design

Adapted by Prof. Gheith Abandah

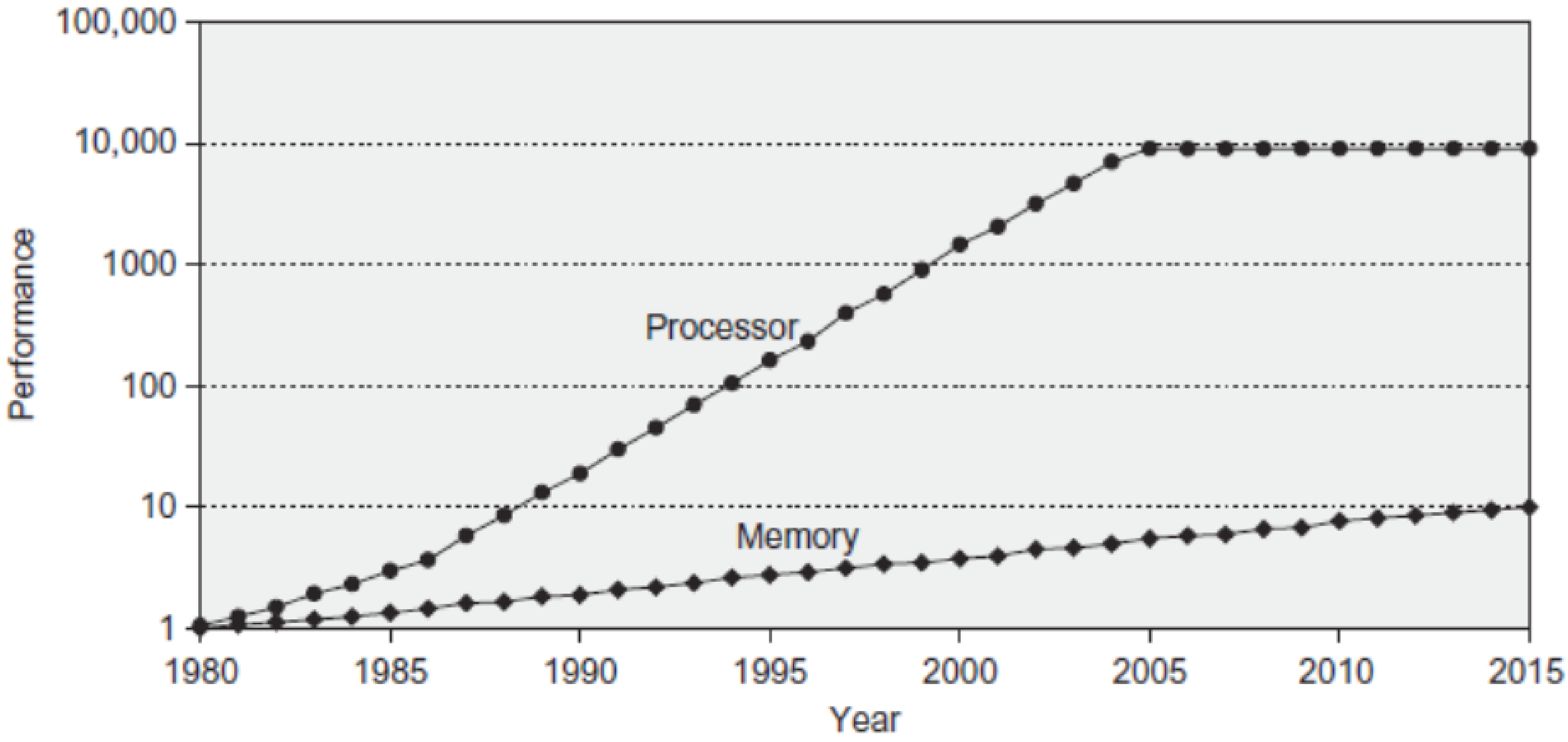
Contents

- Introduction
- Memory Technology and Optimizations
- Ten Advanced Optimizations of Cache Performance
- Virtual Memory and Virtual Machines
- ARM Cortex-A53 and Intel Core i7 6700
- Fallacies and Pitfalls

Contents

- Introduction
  - Memory Performance Gap
  - Memory Hierarchy
  - Memory Hierarchy Design
  - Memory Hierarchy Basics
    - Direct-Mapped Caches
    - Associative Caches
    - Cache Writing Strategies
    - Cache Misses
  - Six Basic Cache Optimizations

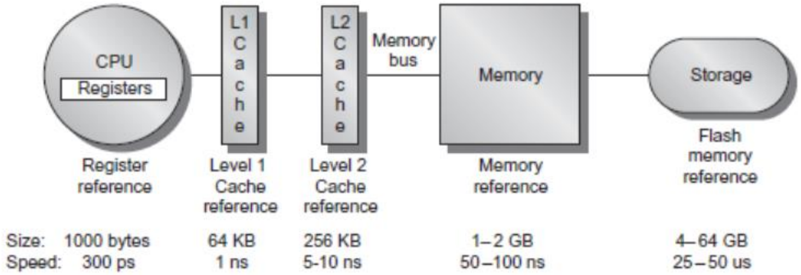
Memory Performance Gap



Memory Hierarchy

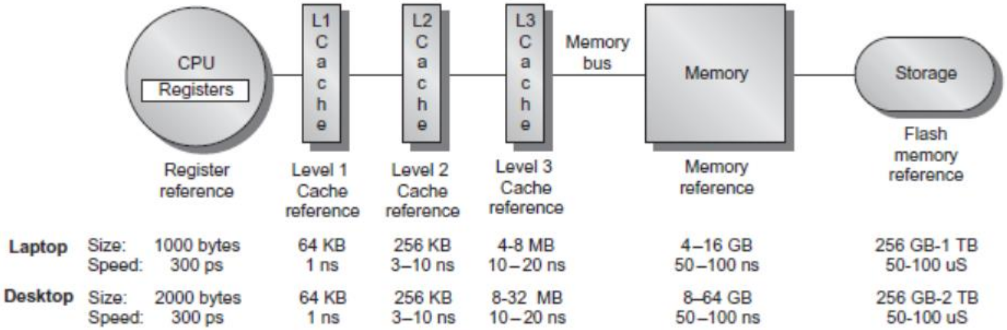
- Programmers want unlimited amounts of memory with low latency
- Fast memory technology is more expensive per bit than slower memory
- Solution: organize memory system into a hierarchy
  - Entire addressable memory space available in largest, slowest memory
  - Incrementally smaller and faster memories, each containing a subset of the memory below it, proceed in steps up toward the processor
- Temporal and spatial locality insures that nearly all references can be found in smaller memories
  - Gives the allusion of a large, fast memory being presented to the processor

Memory Hierarchy



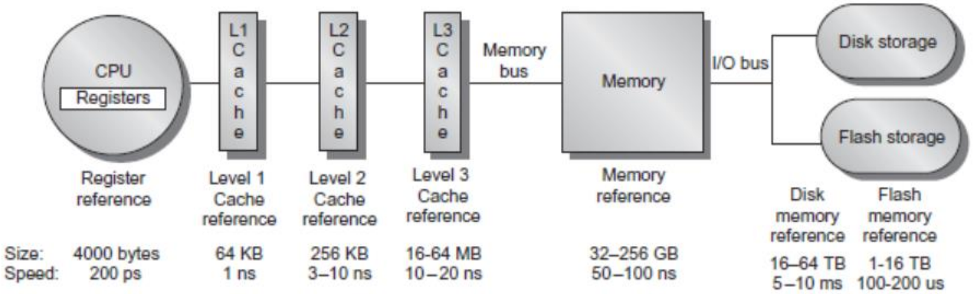
	Register reference	Level 1 Cache reference	Level 2 Cache reference	Memory reference	Flash memory reference
Size:	1000 bytes	64 KB	256 KB	1–2 GB	4–64 GB
Speed:	300 ps	1 ns	5-10 ns	50–100 ns	25–50 us

(A) Memory hierarchy for a personal mobile device



		Register reference	Level 1 Cache reference	Level 2 Cache reference	Level 3 Cache reference	Memory reference	Flash memory reference
Laptop	Size:	1000 bytes	64 KB	256 KB	4-8 MB	4–16 GB	256 GB-1 TB
	Speed:	300 ps	1 ns	3–10 ns	10–20 ns	50–100 ns	50-100 uS
Desktop	Size:	2000 bytes	64 KB	256 KB	8-32 MB	8–64 GB	256 GB-2 TB
	Speed:	300 ps	1 ns	3–10 ns	10–20 ns	50–100 ns	50-100 uS

(B) Memory hierarchy for a laptop or a desktop



	Register reference	Level 1 Cache reference	Level 2 Cache reference	Level 3 Cache reference	Memory reference	Disk memory reference	Flash memory reference
Size:	4000 bytes	64 KB	256 KB	16-64 MB	32–256 GB	16–64 TB	1-16 TB
Speed:	200 ps	1 ns	3–10 ns	10–20 ns	50–100 ns	5–10 ms	100-200 us

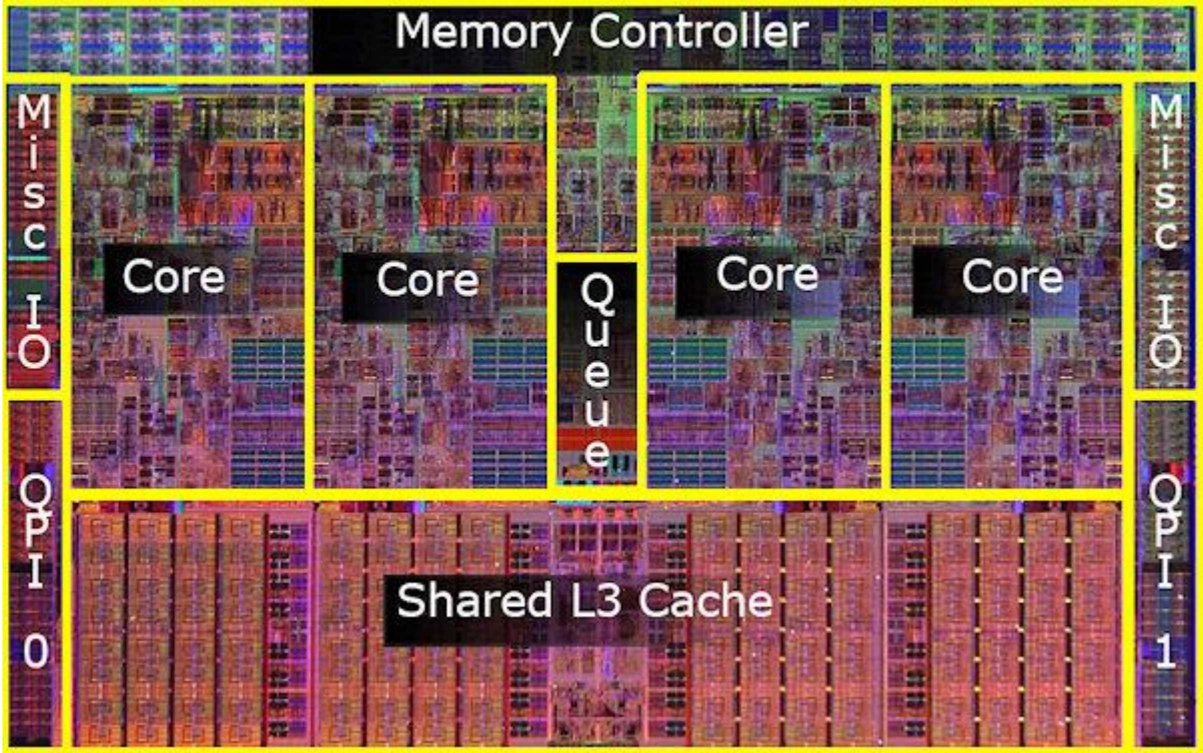
(C) Memory hierarchy for server

Memory Hierarchy Design

- Memory hierarchy design becomes more crucial with recent multi-core processors:
  - Aggregate peak bandwidth grows with # cores:
    - Intel Core i7 can generate two references per core per clock
    - Four cores and 3.2 GHz clock
      - 25.6 billion 64-bit data references/second +
      - 12.8 billion 128-bit instruction references/second
      - = 409.6 GB/s!
    - DRAM bandwidth is only 8% of this (34.1 GB/s)
    - Requires:
      - Multi-port, pipelined caches
      - Two levels of cache per core
      - Shared third-level cache on chip

Memory Hierarchy Design

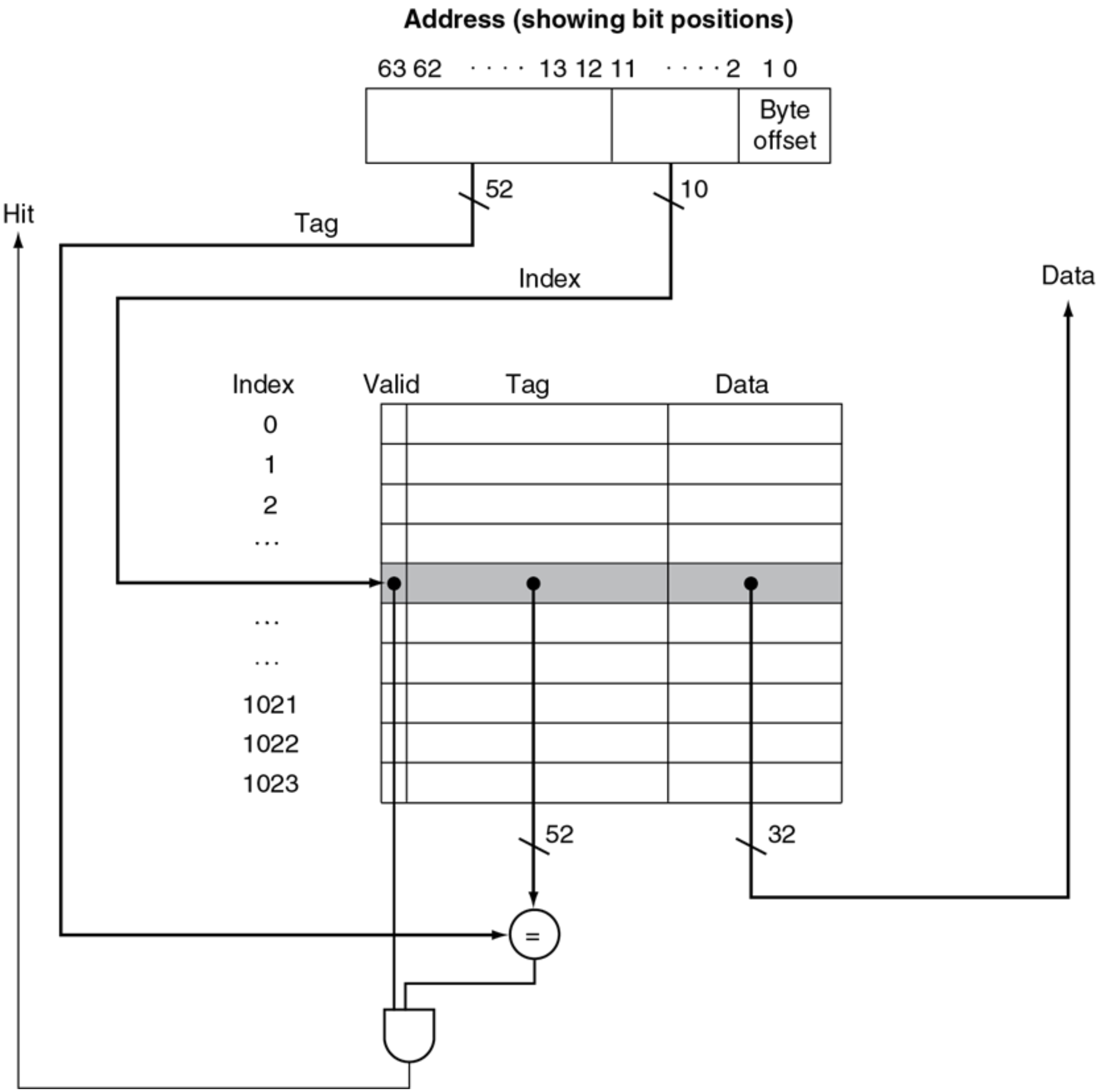
- High-end microprocessors have >10 MB on-chip cache
  - Consumes large amount of area and power budget



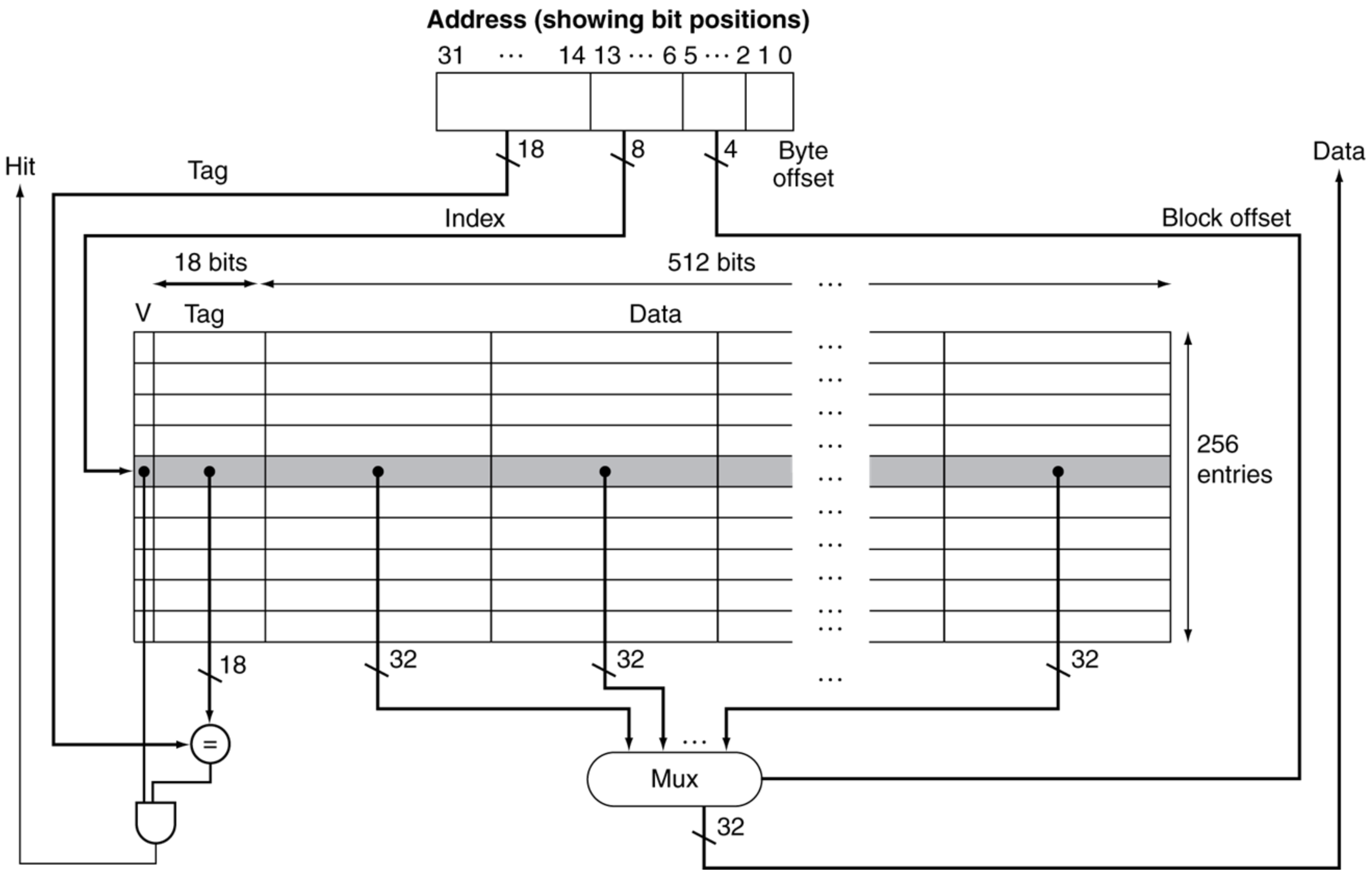
Memory Hierarchy Basics

- When a word is not found in the cache, a *miss* occurs:
  - Fetch word from lower level in hierarchy, requiring a higher latency reference
  - Lower level may be another cache or the main memory
  - Also fetch the other words contained within the *block*
    - Takes advantage of spatial locality
  - Place block into cache in any location within its *set*, determined by address
    - block address MOD number of sets in cache

Direct Mapped Cache

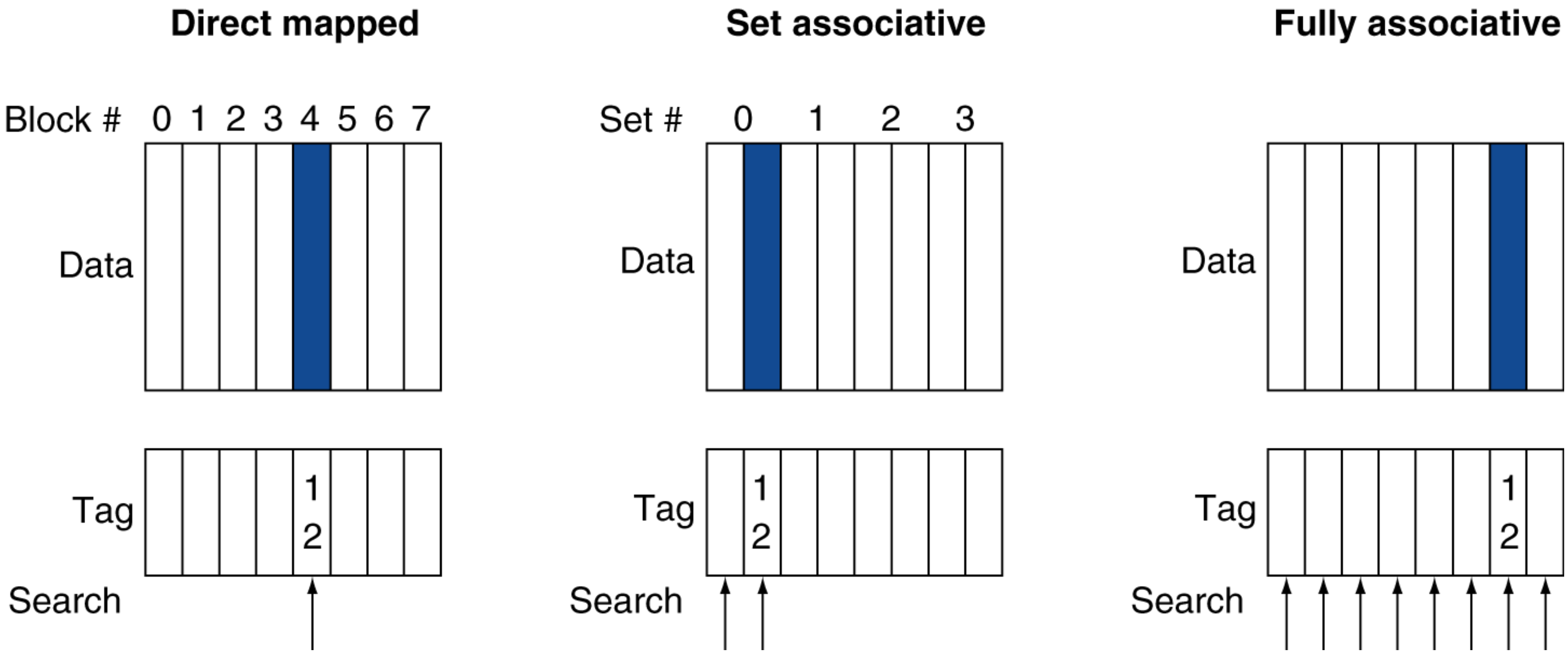


Direct Mapped Cache, Large Blocks

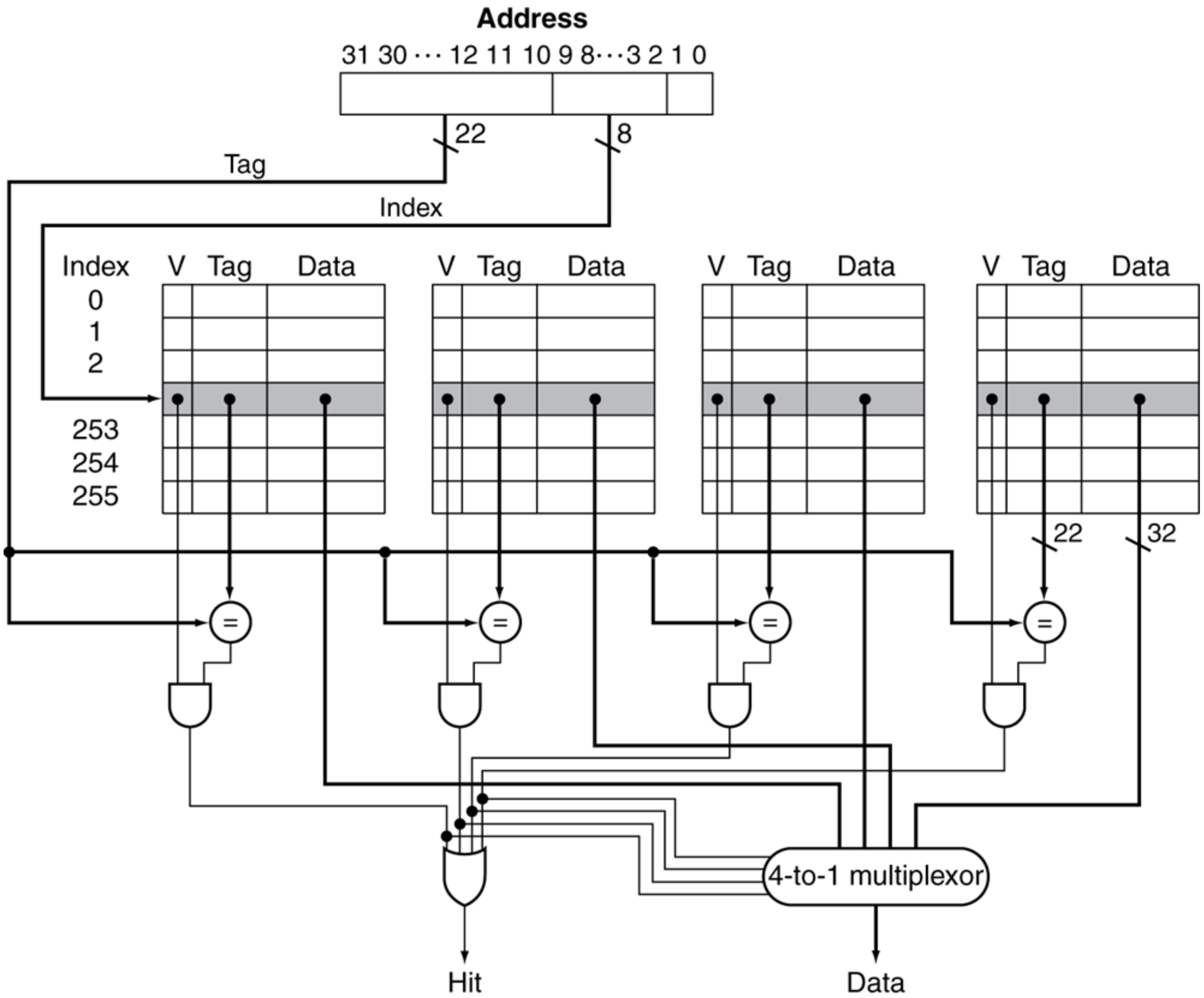


Memory Hierarchy Basics

- Associative Caches
  - *Direct-mapped cache =>* one block per set
  - *N-way set associative => n blocks per set*
  - *Fully associative* => one set. All blocks in one set



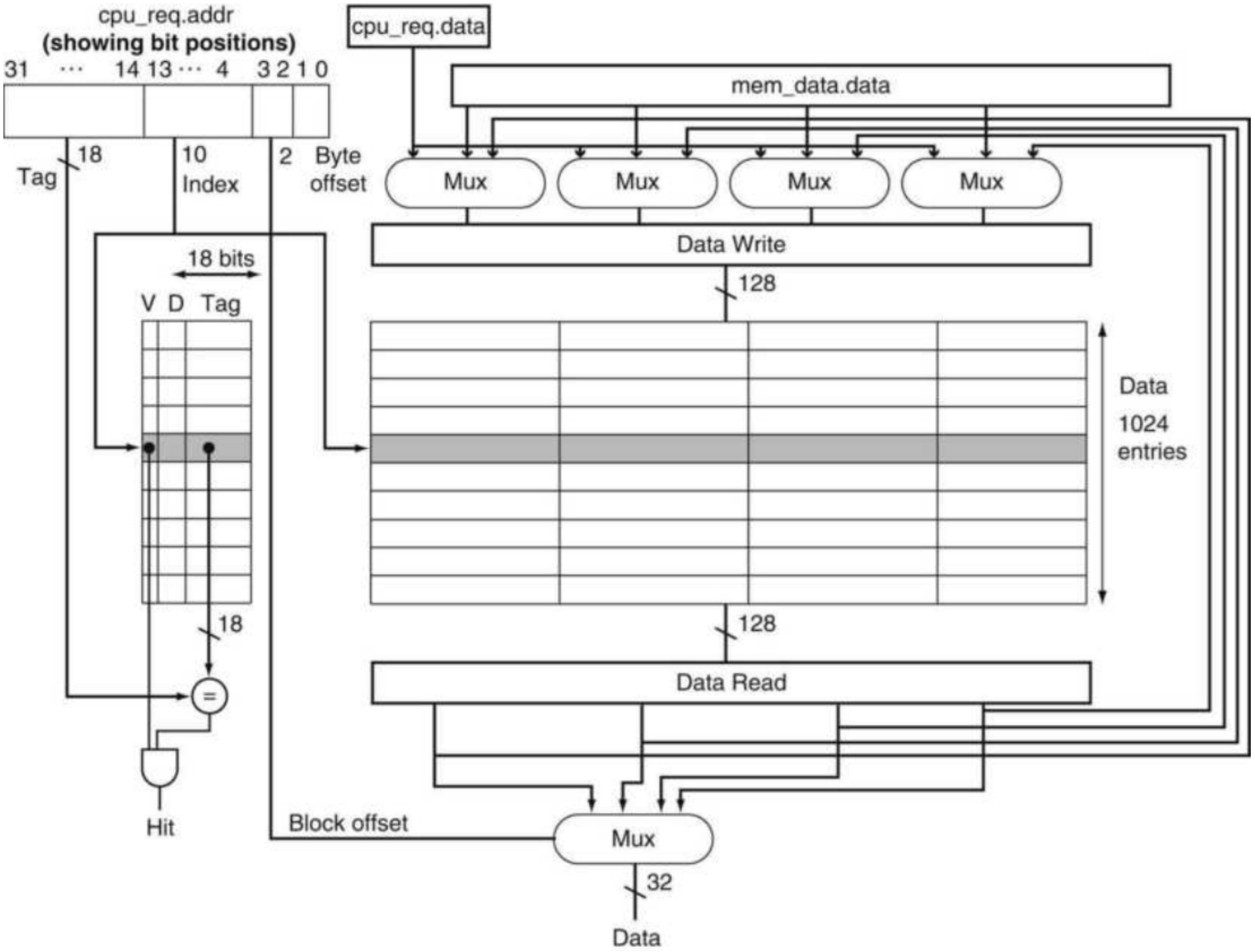
Four-Way Associative Cache



Memory Hierarchy Basics

- Writing to cache: two strategies
  - *Write-through*
    - Immediately update lower levels of hierarchy
  - *Write-back*
    - Only update lower levels of hierarchy when an updated block is replaced
  - Both strategies use *write buffer* to make writes asynchronous

Write-Back Direct Mapped Cache



Cache Misses

- Miss rate
  - Fraction of cache access that result in a miss
- Causes of misses
  - Compulsory
    - First reference to a block
  - Capacity
    - Blocks discarded and later retrieved
  - Conflict
    - Program makes repeated references to multiple addresses from different blocks that map to the same location in the cache

Cache Misses

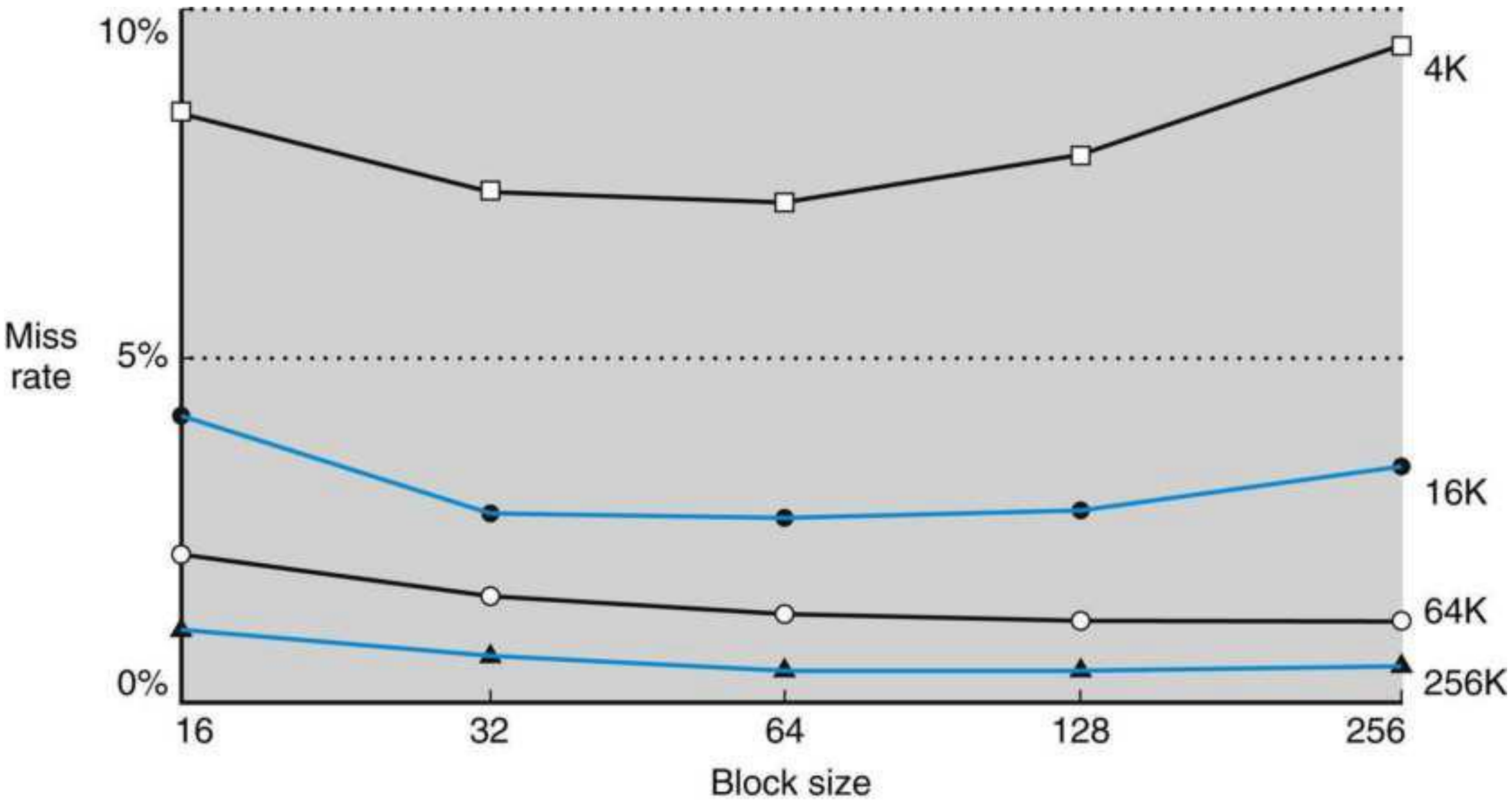
$$\frac{Misses}{Instruction} = \frac{Miss\,rate \times Memory\,accesses}{Instruction\,count} = Miss\,rate \times \frac{Memory\,accesses}{Instruction}$$

$$Average\,memory\,access\,time = Hit\,time + Miss\,rate \times Miss\,penalty$$

- Speculative and multithreaded processors may execute other instructions during a miss
  - Reduces performance impact of misses

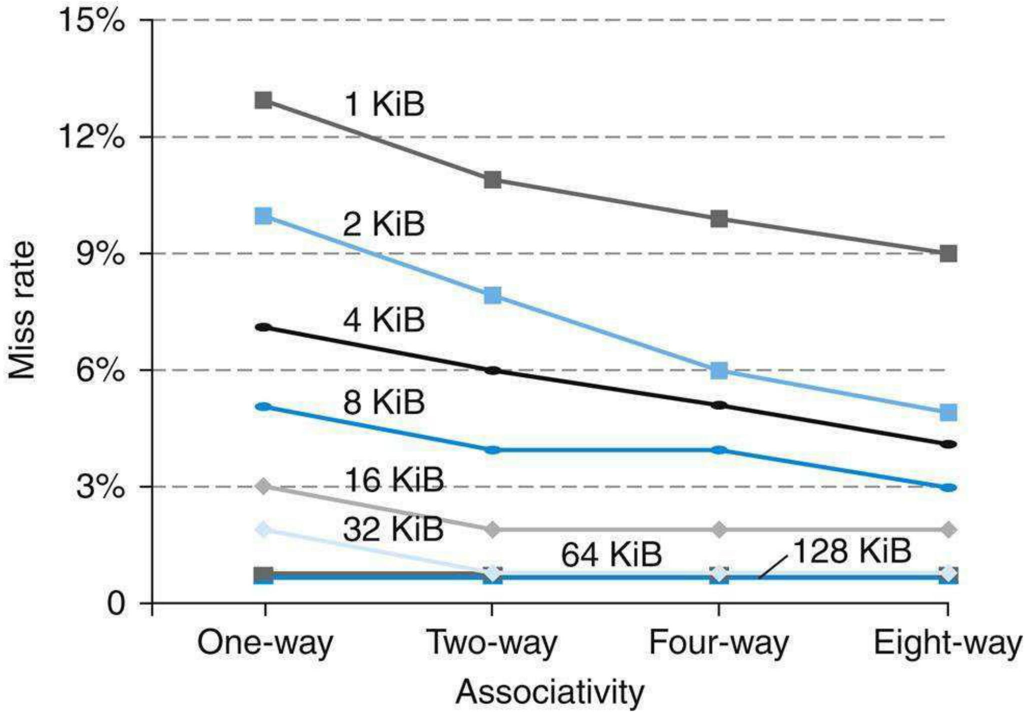
Six Basic Cache Optimizations

1. Larger block size
   - Reduces compulsory misses
   - Increases capacity and conflict misses, increases miss penalty



Six Basic Cache Optimizations

2. Larger total cache capacity to reduce miss rate
    - Increases hit time, increases power consumption
3. Higher associativity
    - Reduces conflict misses
    - Increases hit time, increases power consumption



Six Basic Cache Optimizations

4. Higher number of cache levels
    - Reduces overall memory access time

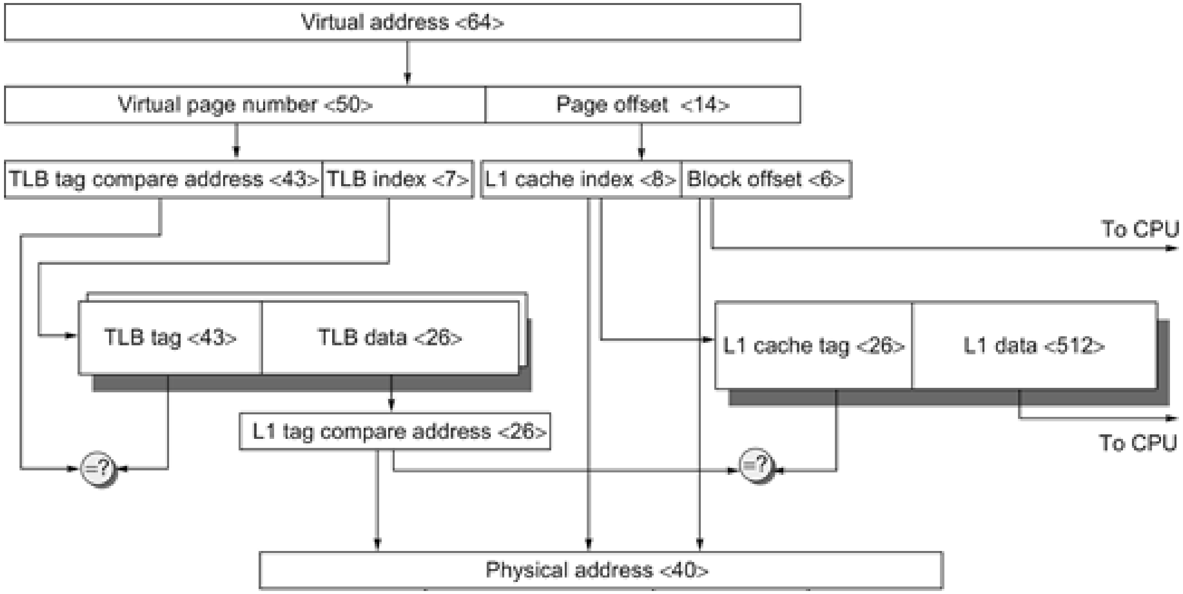
- Example 1: Find the average memory access time for a memory hierarchy of one cache and a main memory given the following:
    - Hit time = 1 cycle
    - Miss rate = 5%
    - Miss penalty = 200 cycles

- Example 2: Repeat Example 1 when an L2 is added with the following specs:
    - Hit time = 10 cycles
    - Miss rate = 2%
    - Miss penalty = 250 cycles

Six Basic Cache Optimizations

5. Giving priority to read misses over writes
   - Reduces miss penalty

6. Avoiding address translation in cache indexing
   - Reduces hit time



Contents

- Introduction
- Memory Technology and Optimizations
- Ten Advanced Optimizations of Cache Performance
- Virtual Memory and Virtual Machines
- ARM Cortex-A53 and Intel Core i7 6700
- Fallacies and Pitfalls

Contents

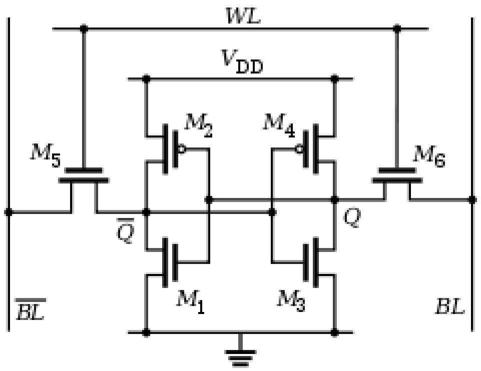
- Memory Technology and Optimizations
  - Introduction
  - Memory Technology
    - SRAM
    - DRAM
  - Memory Optimizations
  - Stacked/Embedded DRAMs
  - Flash Memory
  - Memory Dependability

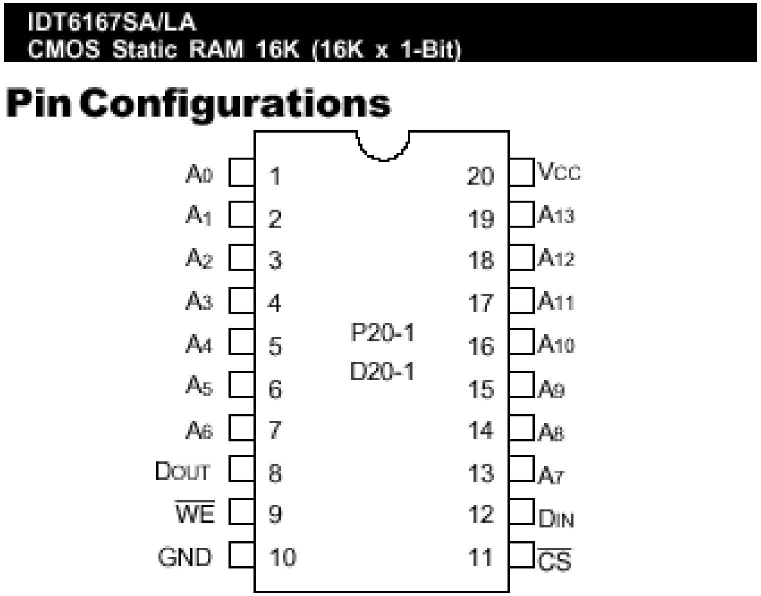
Introduction

- Performance metrics
    - Latency is concern of cache
    - Bandwidth is concern of multiprocessors and I/O
    - Access time
        - Time between read request and when desired word arrives
    - Cycle time
        - Minimum time between unrelated requests to memory
- SRAM memory has low latency, use for cache
- Organize DRAM chips into many banks for high bandwidth, use for main memory

Memory Technology

- SRAM
  - Requires low power to retain bit
  - Requires 6 transistors/bit

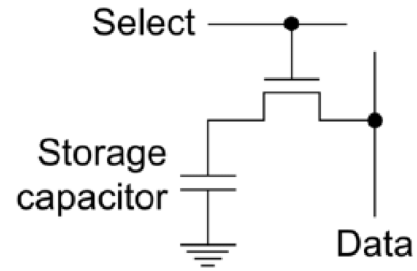



IDT6167SA/LA
CMOS Static RAM 16K (16K x 1-Bit)

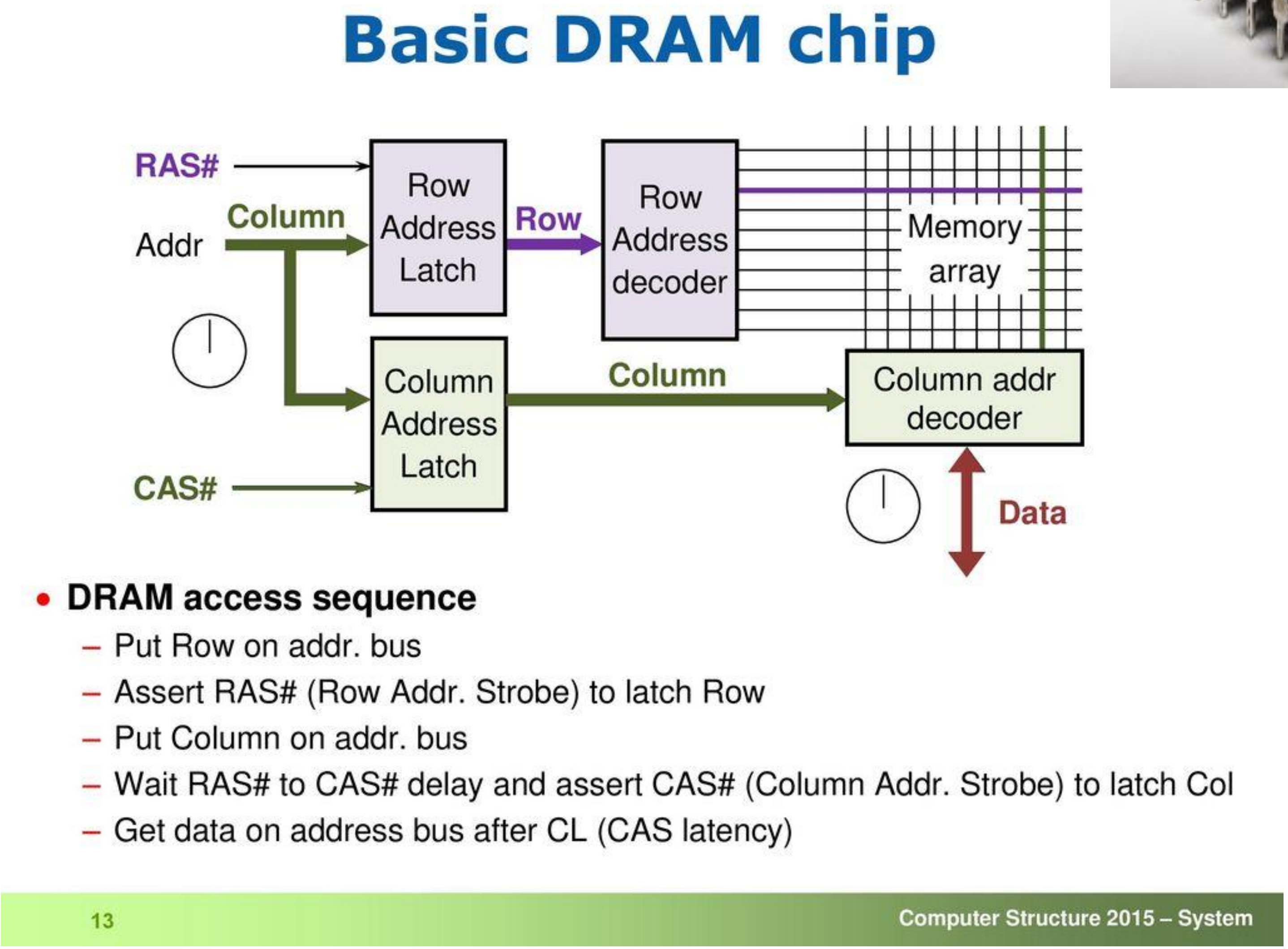
Pin Configurations

Memory Technology

- DRAM
  - Must be re-written after being read
  - Must also be periodically refreshed
    - Every ~ 8 ms (roughly 5% of time)
    - Each row can be refreshed simultaneousl
  - One transistor/bit
  - Address lines are multiplexed:
    - Upper half of address: row access strobe (RAS)
    - Lower half of address: column access strobe (CAS)



Classic DRAM



Basic DRAM chip

- **DRAM access sequence**
  - Put Row on addr. bus
  - Assert RAS# (Row Addr. Strobe) to latch Row
  - Put Column on addr. bus
  - Wait RAS# to CAS# delay and assert CAS# (Column Addr. Strobe) to latch Col
  - Get data on address bus after CL (CAS latency)

13

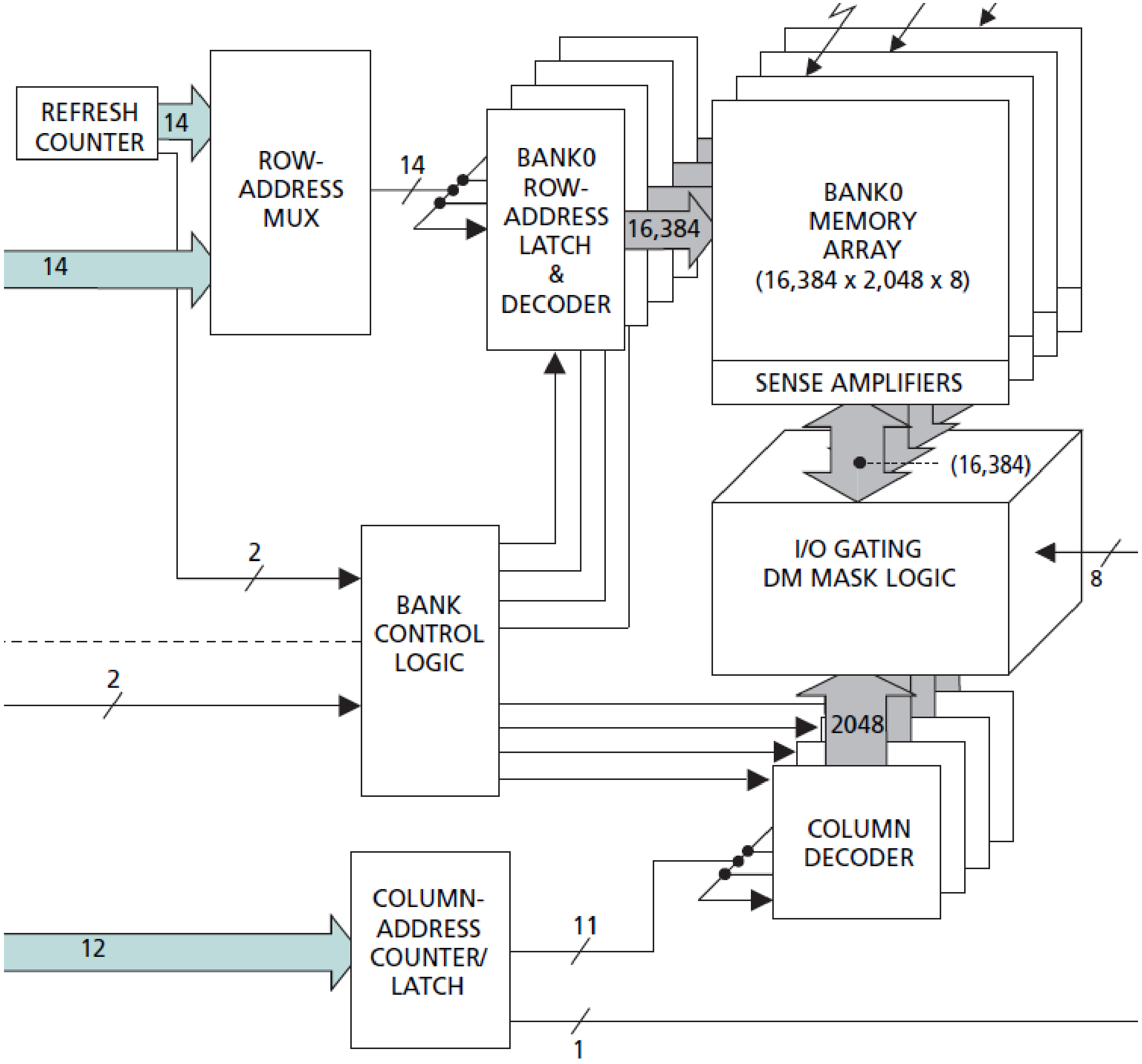
Computer Structure 2015 – System

Memory Optimizations

- 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

Micron DDR-SDRAM

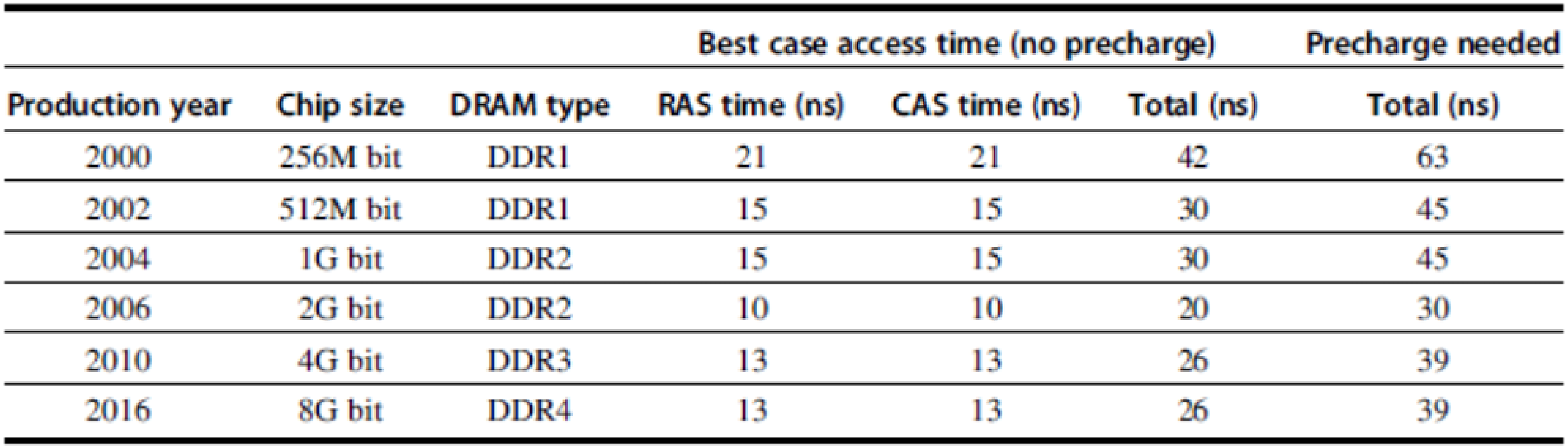


Memory Optimizations

- 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

Memory Optimizations

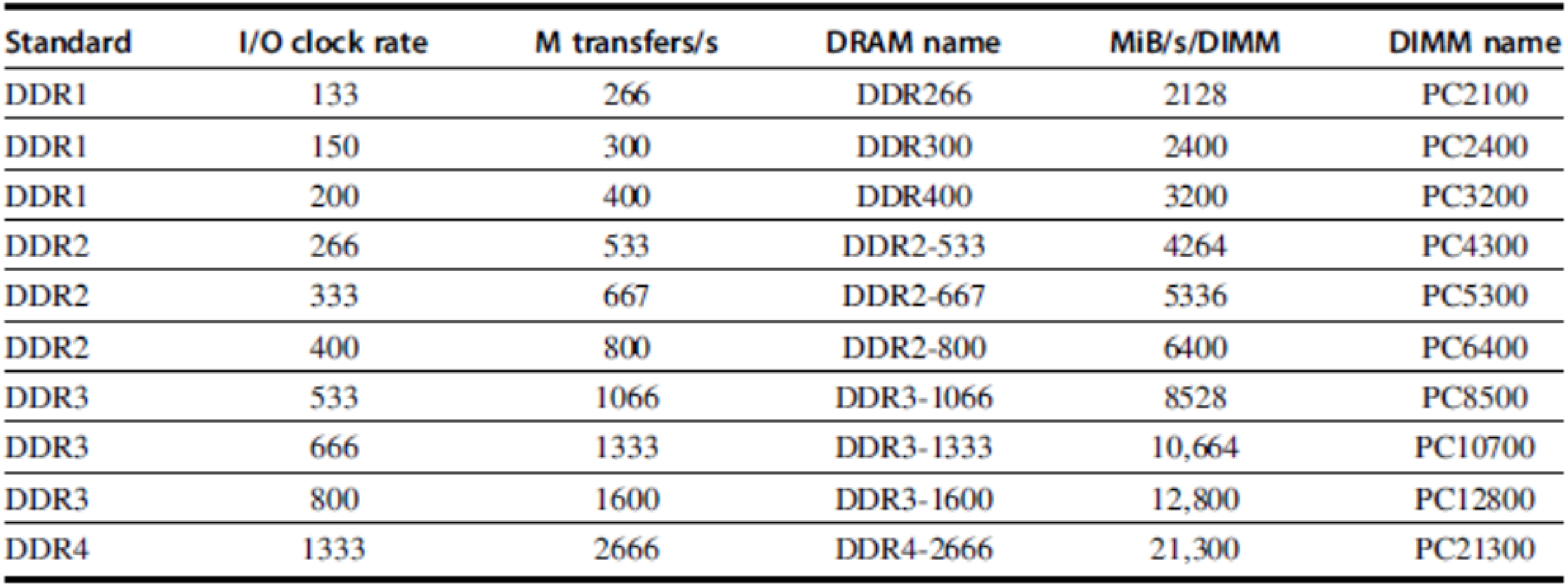
DDR-SDRAM Chips



			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

Memory Optimizations

DDR-SDRAM DIMMs



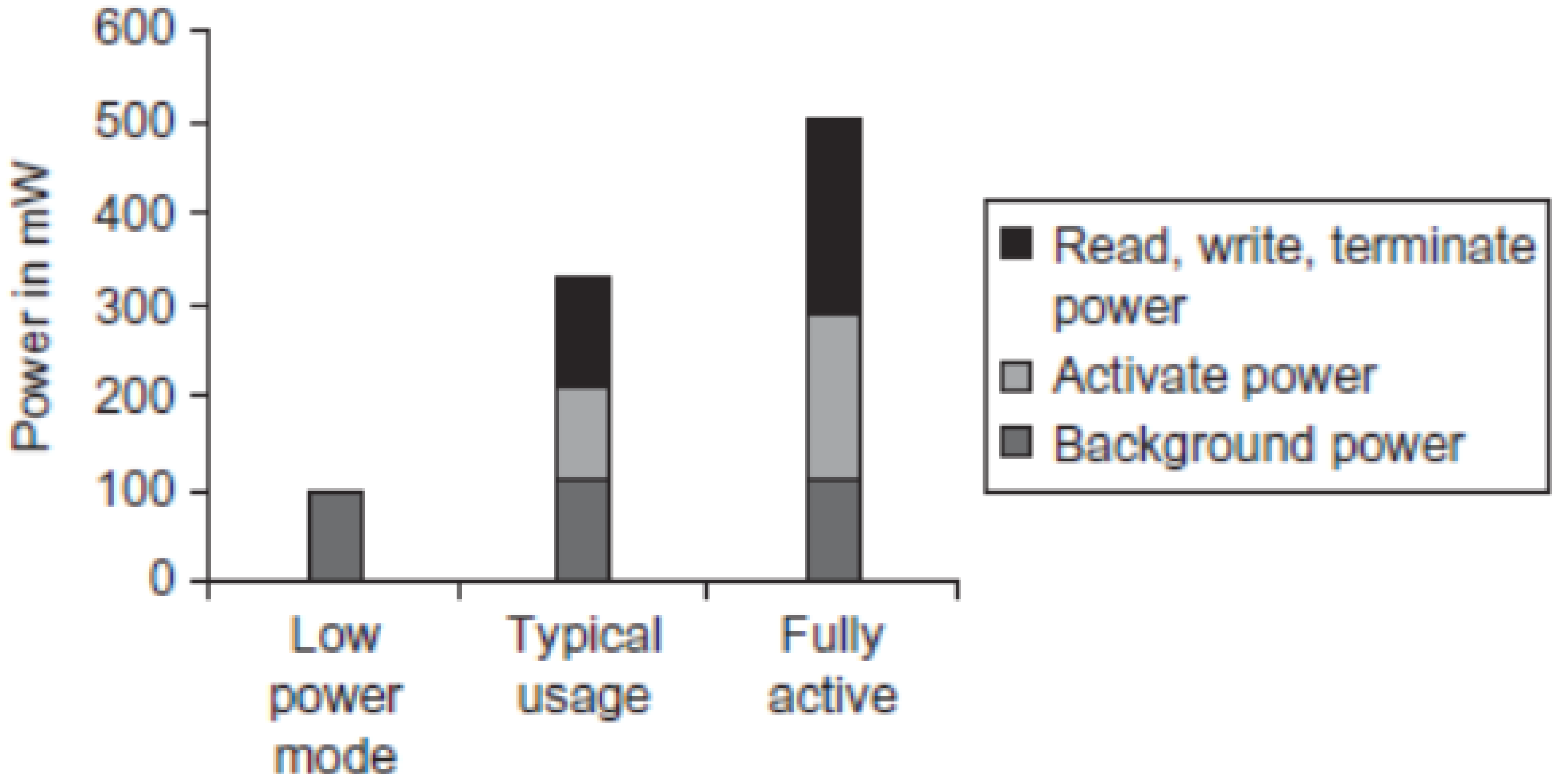
Standard	I/O clock rate	M transfers/s	DRAM name	MiB/s/DIMM	DIMM name
DDR1	133	266	DDR266	2128	PC2100
DDR1	150	300	DDR300	2400	PC2400
DDR1	200	400	DDR400	3200	PC3200
DDR2	266	533	DDR2-533	4264	PC4300
DDR2	333	667	DDR2-667	5336	PC5300
DDR2	400	800	DDR2-800	6400	PC6400
DDR3	533	1066	DDR3-1066	8528	PC8500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

Memory Optimizations

- Reducing power in SDRAMs:
    - Lower voltage
    - Low power mode (ignores clock, continues to refresh)

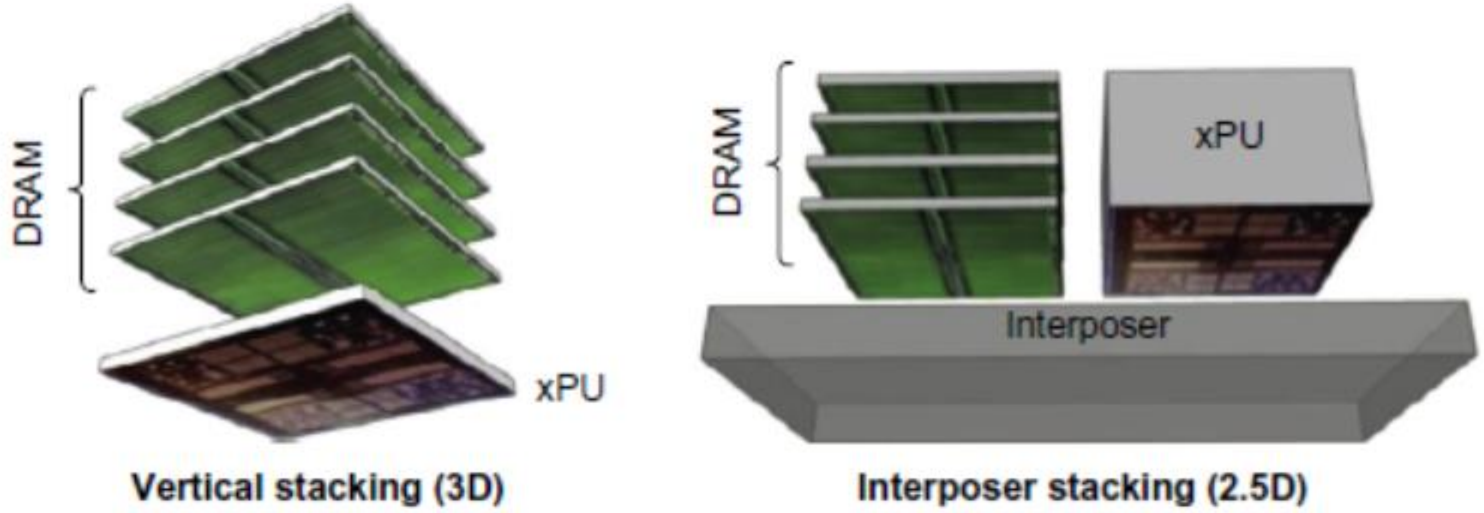
- Graphics memory:
    - Achieve 2-5 X bandwidth per DRAM vs. DDR3
        - Wider interfaces (32 vs. 16 bit)
        - Higher clock rate
            - Possible because they are attached via soldering instead of socketed DIMM modules

Memory Power Consumption



Stacked/Embedded DRAMs

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



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 µs vs 500 ns for SDRAM, 150X slower
  - 300 to 500X faster than magnetic disk

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 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

Contents

- Introduction
- Memory Technology and Optimizations
- Ten Advanced Optimizations of Cache Performance
- Virtual Memory and Virtual Machines
- ARM Cortex-A53 and Intel Core i7 6700
- Fallacies and Pitfalls

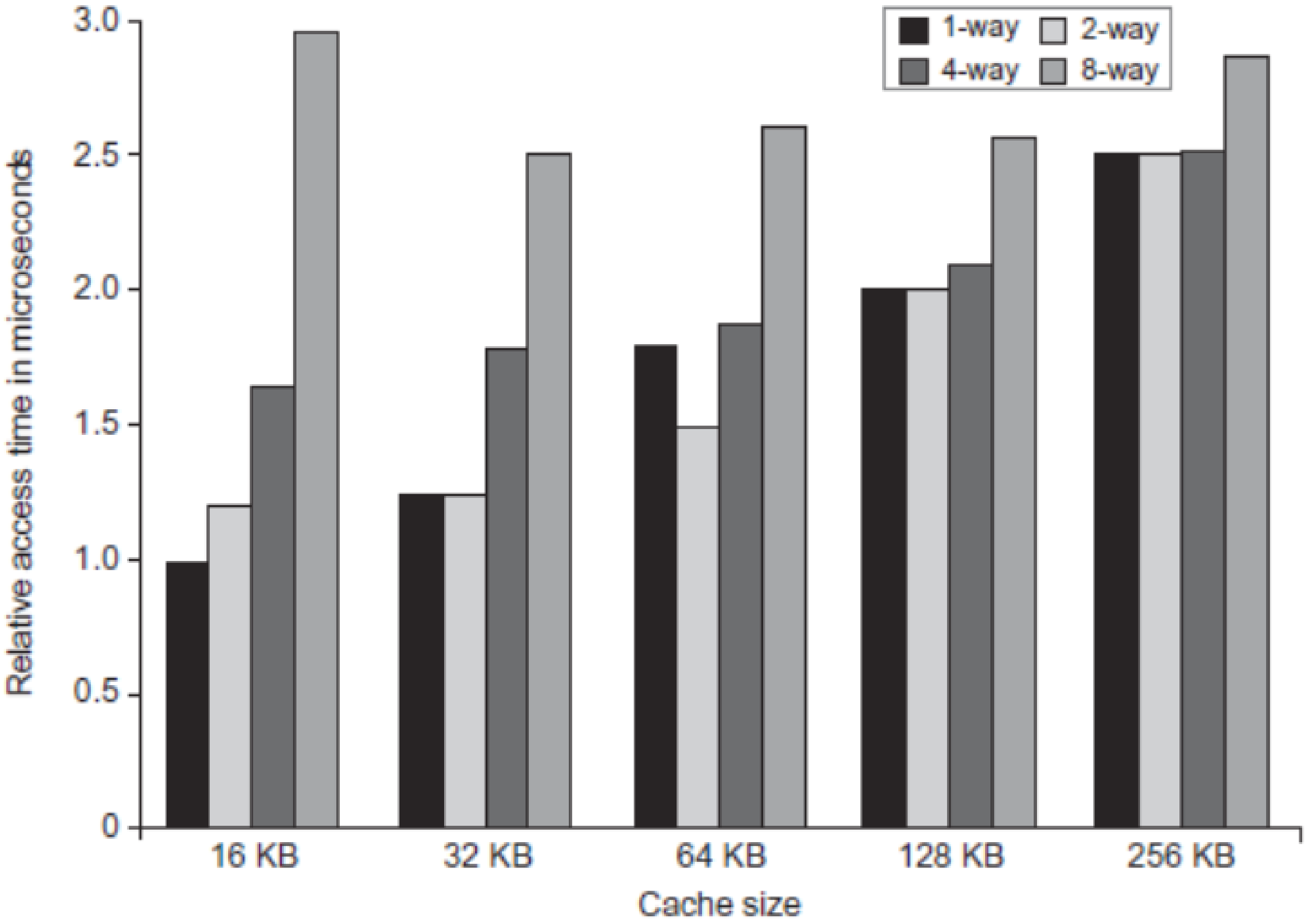
Contents

- Ten Advanced Optimizations of Cache Performance
  - Reduce hit time
    - (1) Small and simple first-level caches
    - (2) Way prediction
  - Increase bandwidth
    - (3) Pipelined, (3) multibanked, or (4) non-blocking caches
  - Reduce miss penalty
    - (5) Critical word first, (6) merging write buffers
  - Reduce miss rate
    - (7) Compiler optimizations
  - Reduce miss penalty or miss rate via parallelization
    - (8) Hardware or (9) compiler prefetching
  - (10) Using HBM to Extend Memory Hierarchy

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

(1) L1 Hit Time



Access time vs. size and associativity

(2) 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
- Extend to predict block as well
  - “Way selection”
  - Increases misprediction penalty

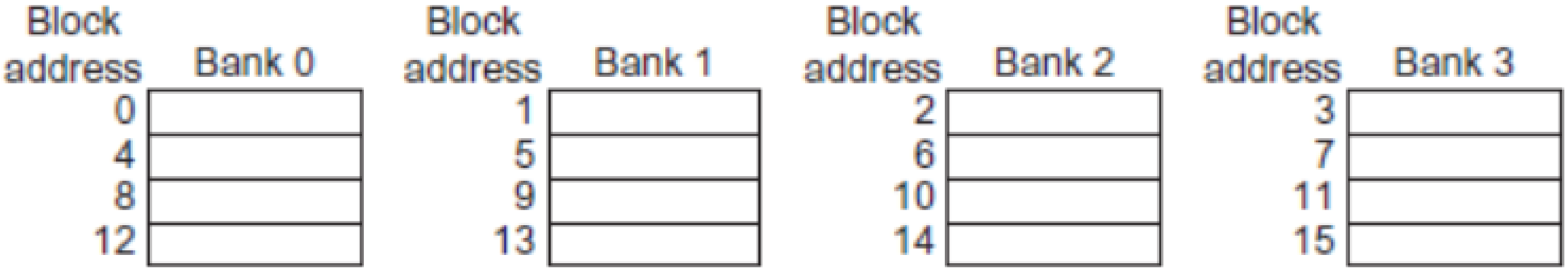
(3) Pipelined Caches

- Pipeline cache access to improve bandwidth
  - Examples:
    - Pentium: 1 cycle
    - Pentium Pro – Pentium III: 2 cycles
    - Pentium 4 – Core i7: 4 cycles
- Increases branch misprediction penalty
- Makes it easier to increase associativity

(3) Multibanked Caches

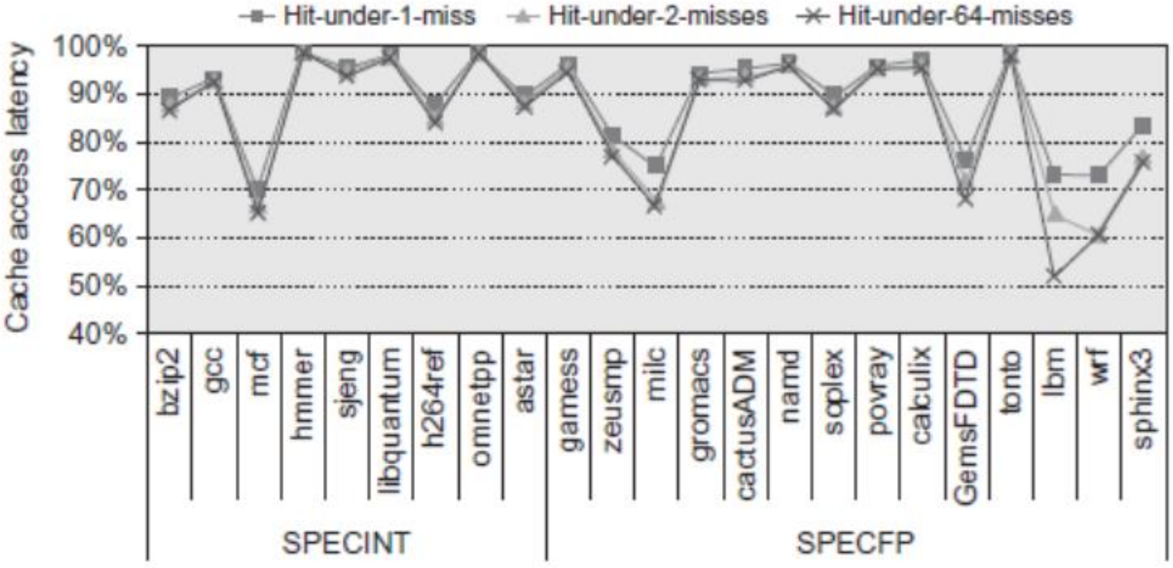
- Organize cache as independent banks to support simultaneous access
  - ARM Cortex-A8 supports 1-4 banks for L2
  - Intel i7 supports 4 banks for L1 and 8 banks for L2

- Interleave banks according to block address



(4) Nonblocking Caches

- 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



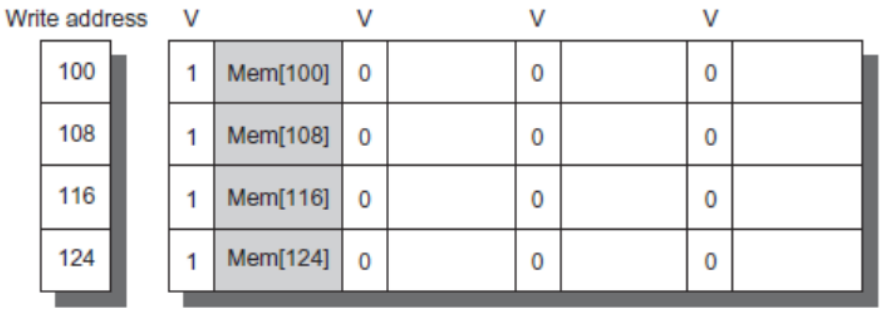
Reduce Miss Penalty

- (5) Critical word first
  - Request missed word from memory first
  - Send it to the processor as soon as it arrives
- (5) 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

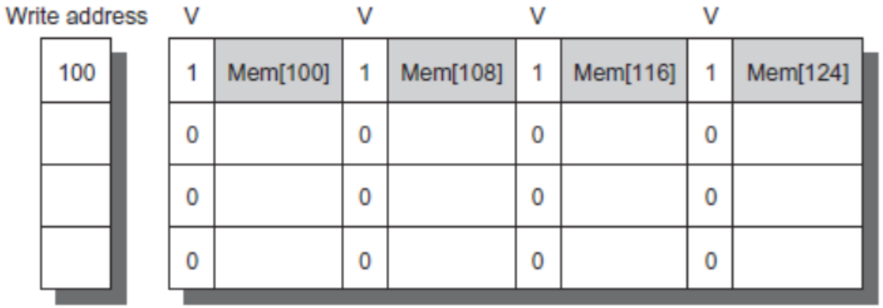
(6) Merging Write Buffer

- 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



Write address	V		V		V		V	
100	1	Mem[100]	0		0		0	
108	1	Mem[108]	0		0		0	
116	1	Mem[116]	0		0		0	
124	1	Mem[124]	0		0		0	

No write buffering

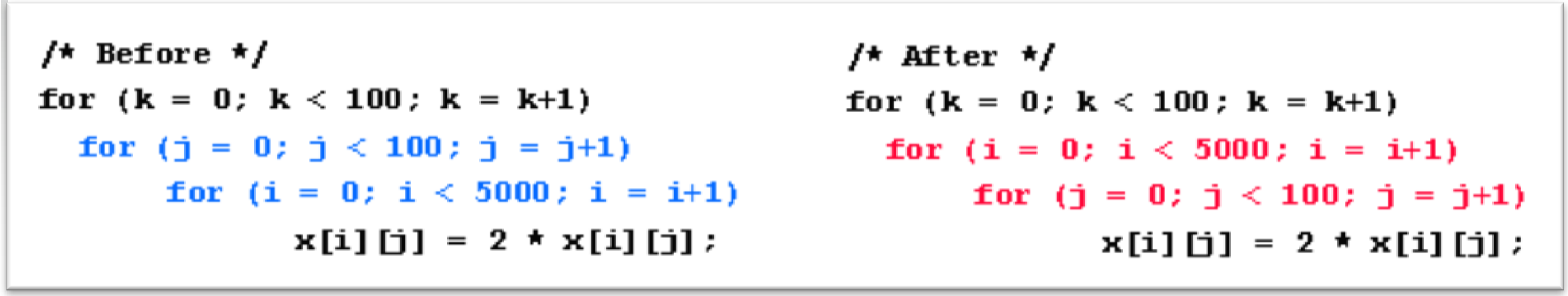


Write address	V		V		V		V	
100	1	Mem[100]	1	Mem[108]	1	Mem[116]	1	Mem[124]
	0		0		0		0	
	0		0		0		0	
	0		0		0		0	

Write buffering

(7) Compiler Optimizations

- Loop Interchange
  - Swap nested loops to access memory in sequential order



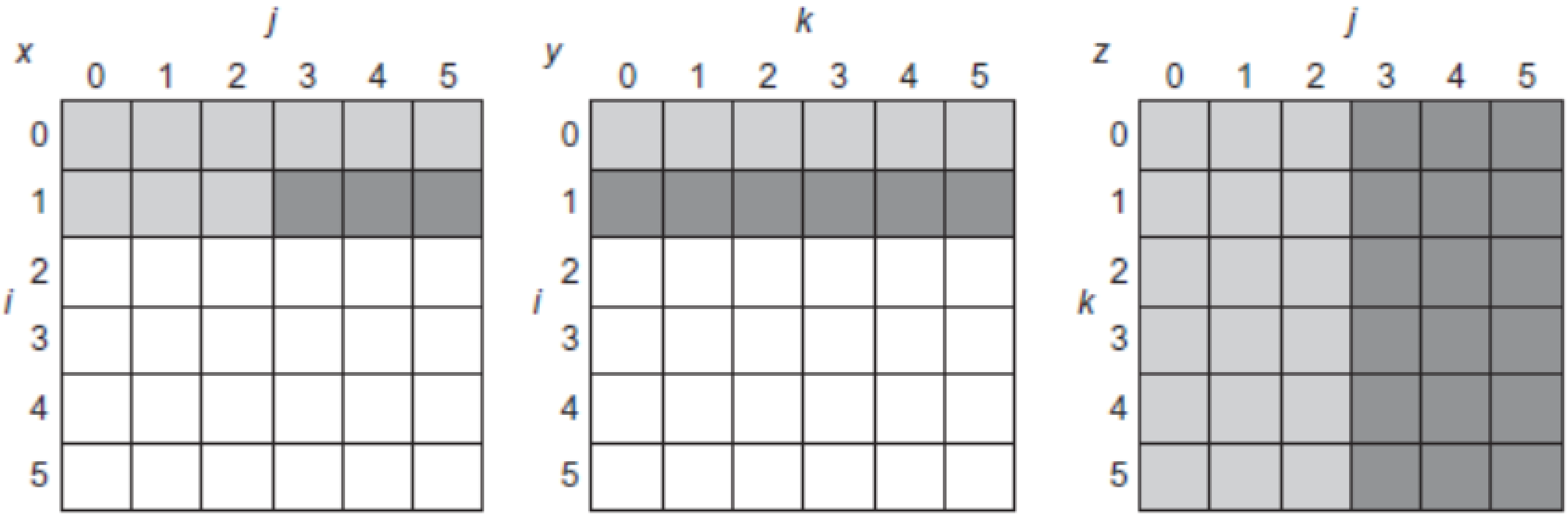
```
/* Before */
for (k = 0; k < 100; k = k+1)
   for (j = 0; j < 100; j = j+1)
       for (i = 0; i < 5000; i = i+1)
            x[i][j] = 2 * x[i][j];
```

```
/* After */
for (k = 0; k < 100; k = k+1)
   for (i = 0; i < 5000; i = i+1)
       for (j = 0; j < 100; j = j+1)
            x[i][j] = 2 * x[i][j];
```

- Blocking
  - Instead of accessing entire rows or columns, subdivide matrices into blocks
  - Requires more memory accesses but improves locality of accesses

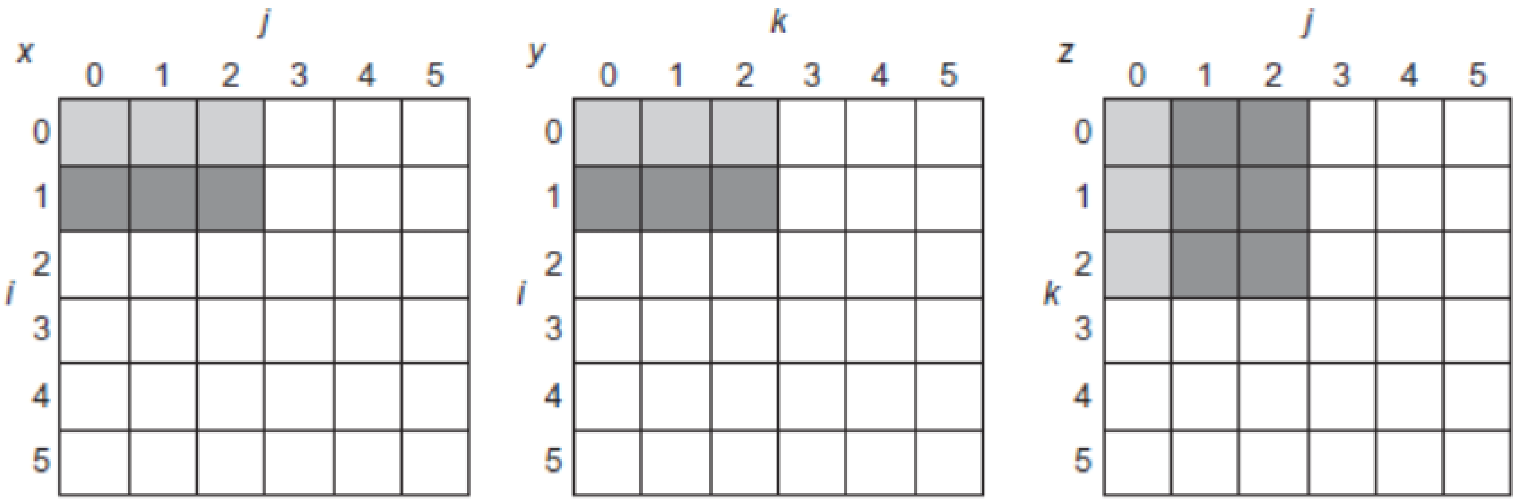
Blocking

```
for (i = 0; i < N; i = i + 1)
  for (j = 0; j < N; j = j + 1)
  {
    r = 0;
    for (k = 0; k < N; k = k + 1)
      r = r + y[i][k]*z[k][j];
    x[i][j] = r;
};
```



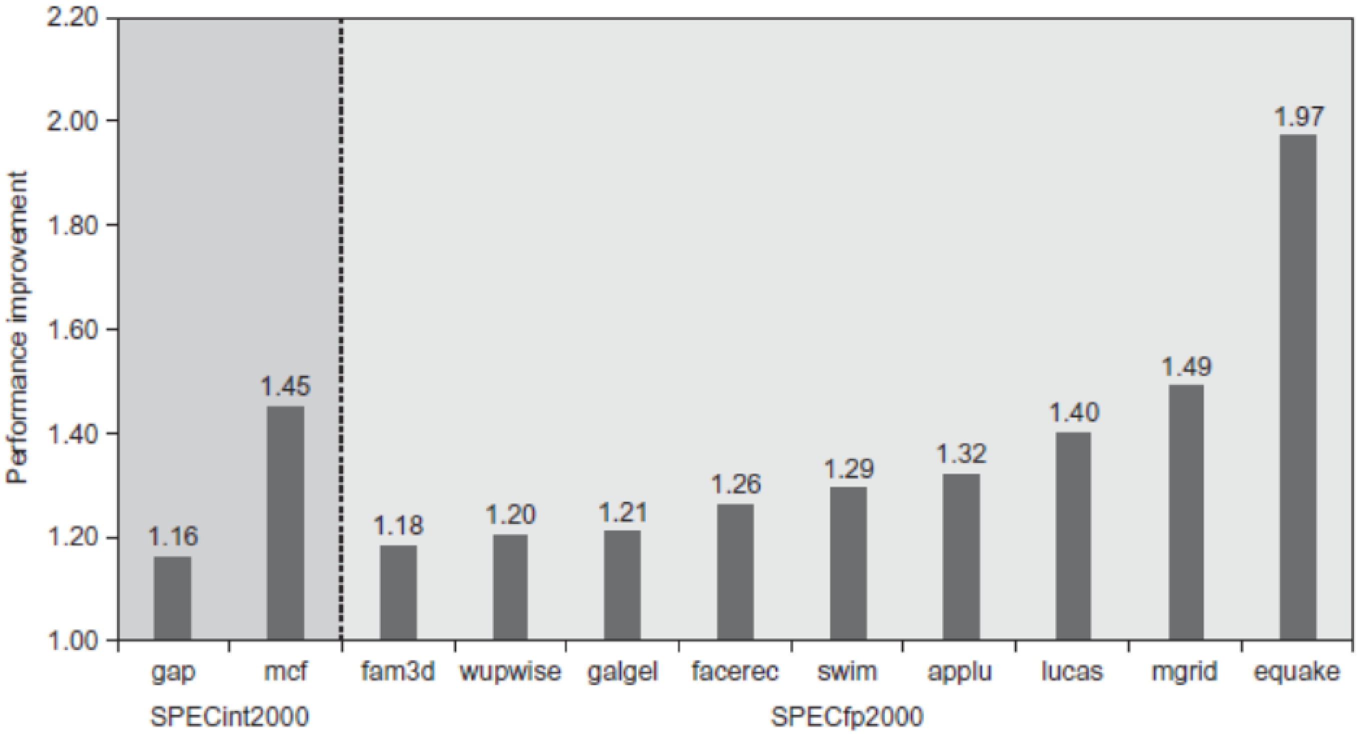
Blocking

```
for (jj = 0; jj < N; jj = jj + B)
  for (kk = 0; kk < N; kk = kk + B)
    for (i = 0; i < N; i = i + 1)
      for (j = jj; j < min(jj + B,N); j = j + 1)
      {
        r = 0;
        for (k = kk; k < min(kk + B,N); k = k + 1)
          r = r + y[i][k]*z[k][j];
        x[i][j] = x[i][j] + r;
};
```



(8) Hardware Prefetching

- Fetch two blocks on miss (include next sequential block)



Pentium 4 Pre-fetching

(9) 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

(10) Use HBM to Extend Hierarchy

- 128 MiB to 1 GiB
- Smaller blocks require substantial tag storage
- Larger blocks are potentially inefficient

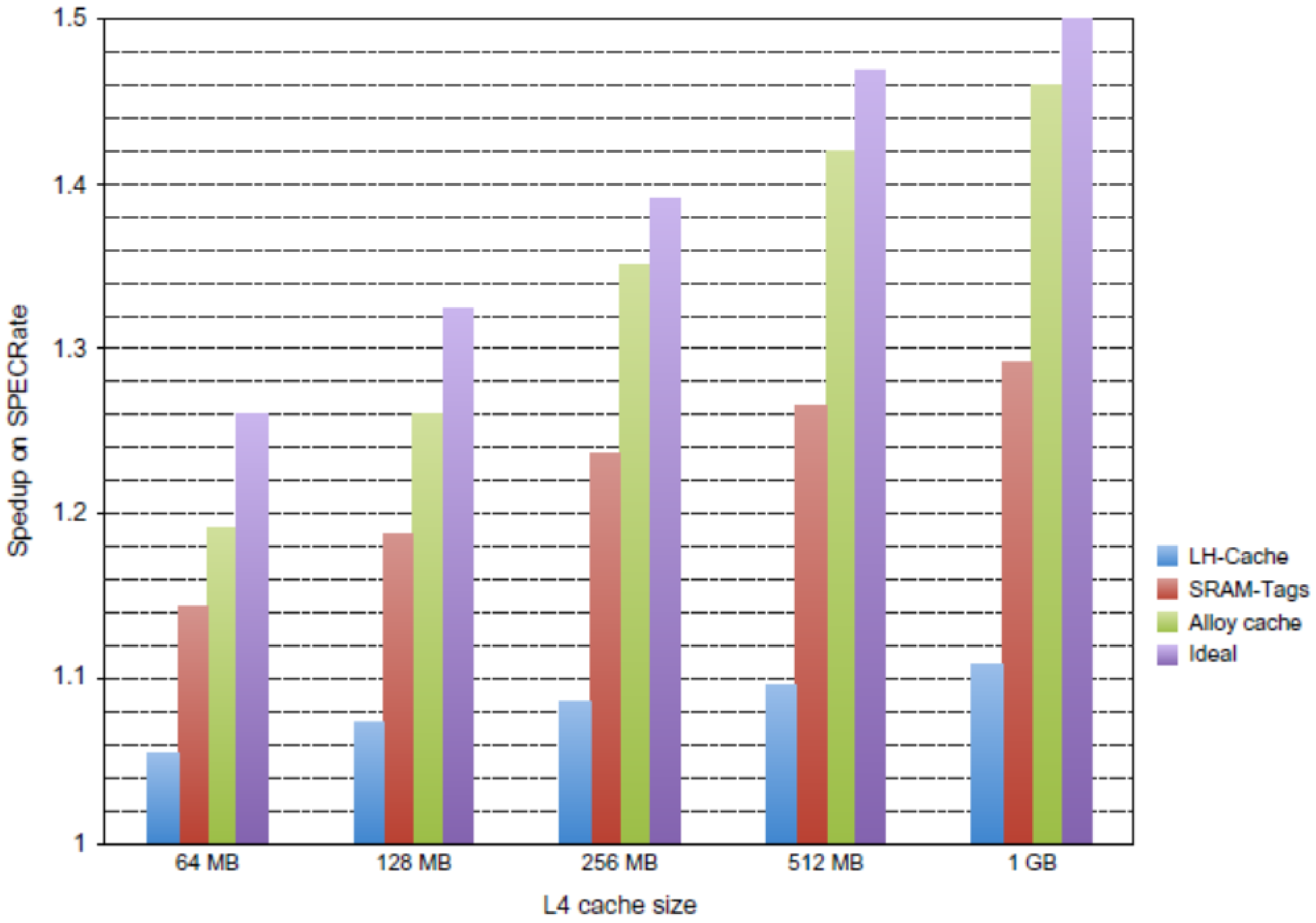
- One approach (L-H):
  - Each SDRAM row is a block index
  - Each row contains set of tags and 29 data segments
  - 29-set associative
  - Hit requires a CAS

Use HBM to Extend Hierarchy

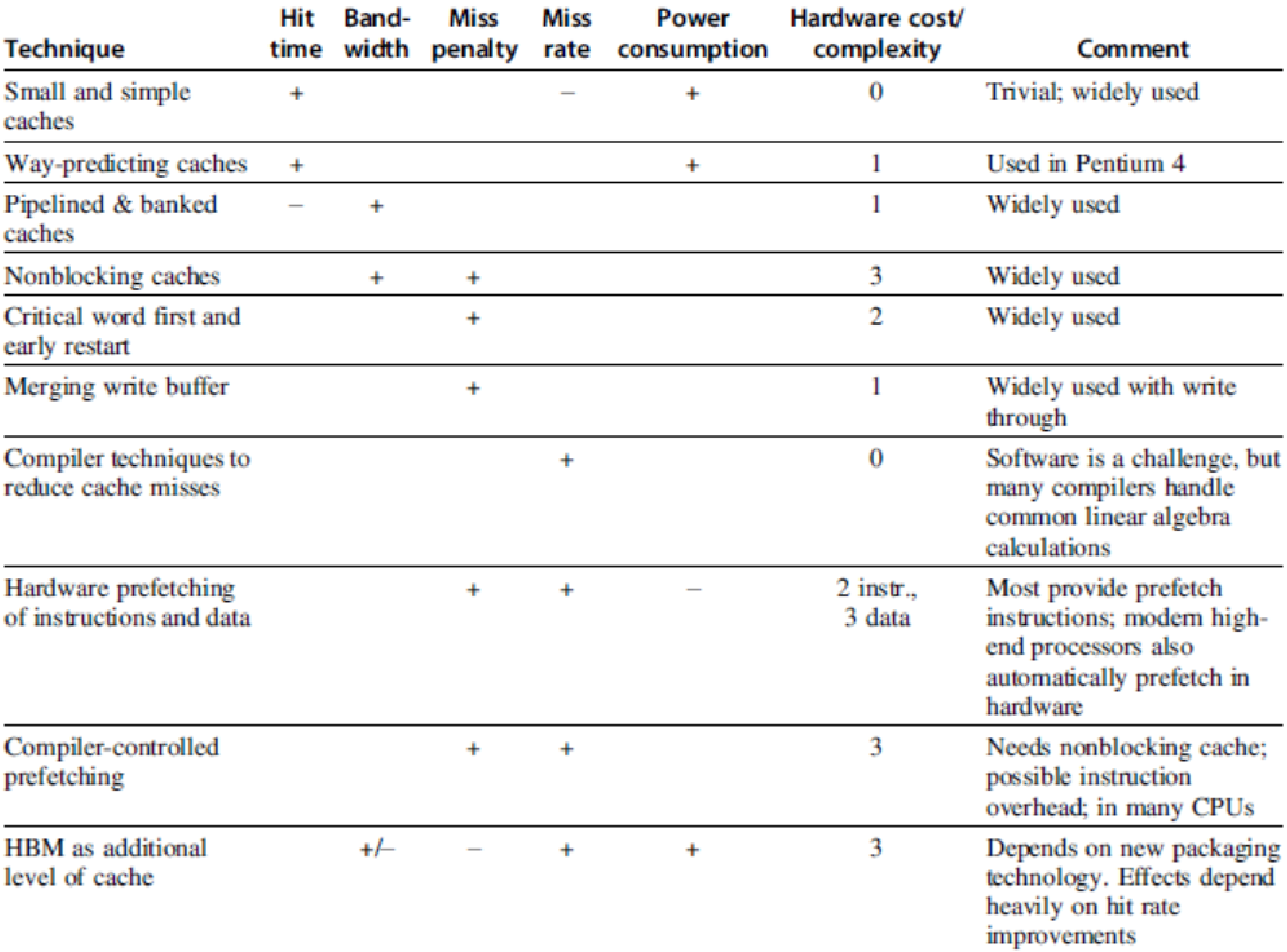
- Another approach (Alloy cache):
  - Mold tag and data together
  - Use direct mapped

- Both schemes require two DRAM accesses for misses
  - Two solutions:
    - Use map to keep track of blocks
    - Predict likely misses

Use HBM to Extend Hierarchy



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 & banked caches	–	+				1	Widely used
Nonblocking caches		+	+			3	Widely used
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
HBM as additional level of cache		+/–	–	+	+	3	Depends on new packaging technology. Effects depend heavily on hit rate improvements

Contents

- Introduction
- Memory Technology and Optimizations
- Ten Advanced Optimizations of Cache Performance
- Virtual Memory and Virtual Machines
- ARM Cortex-A53 and Intel Core i7 6700
- Fallacies and Pitfalls

Virtual Memory Review

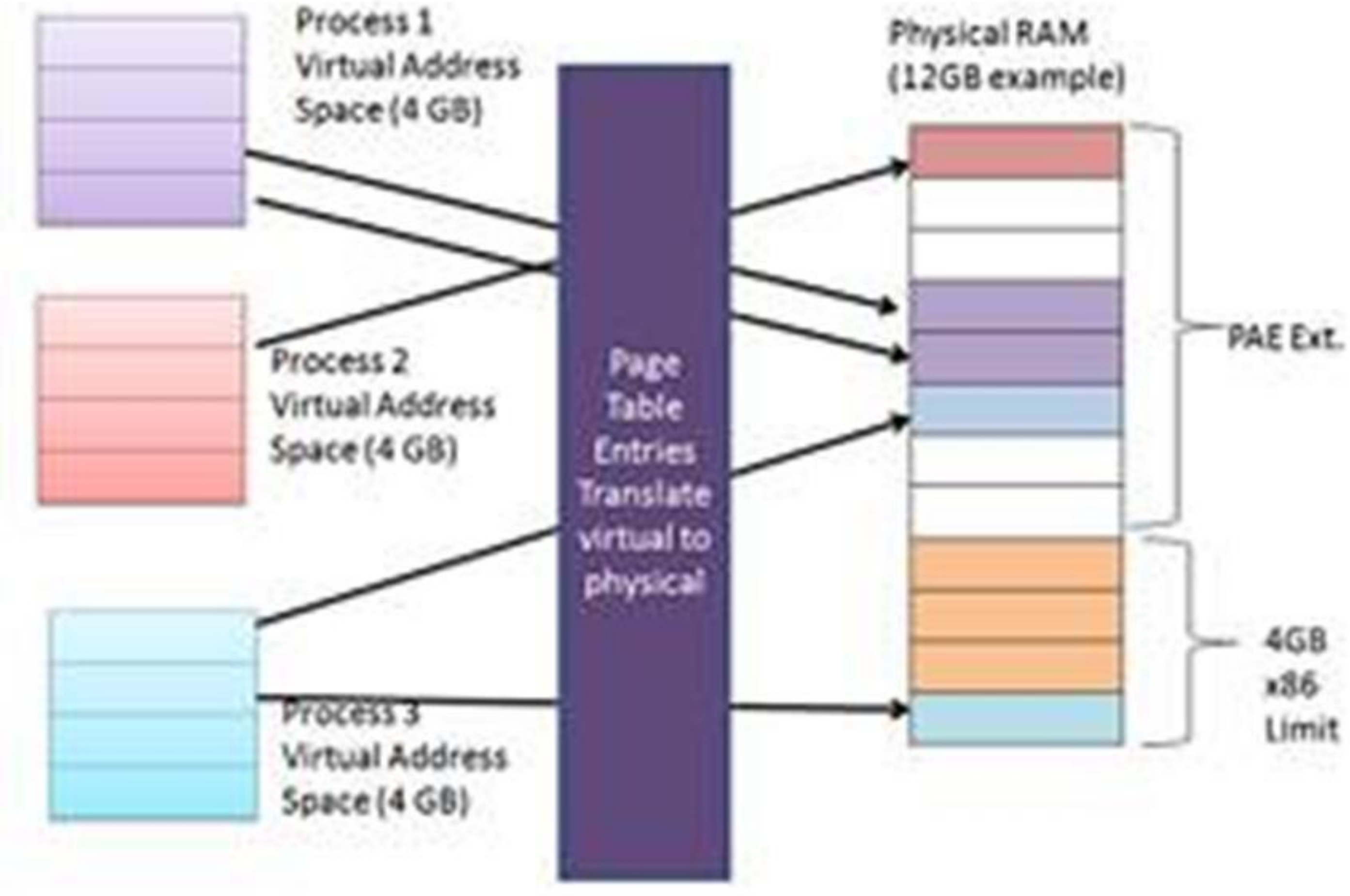
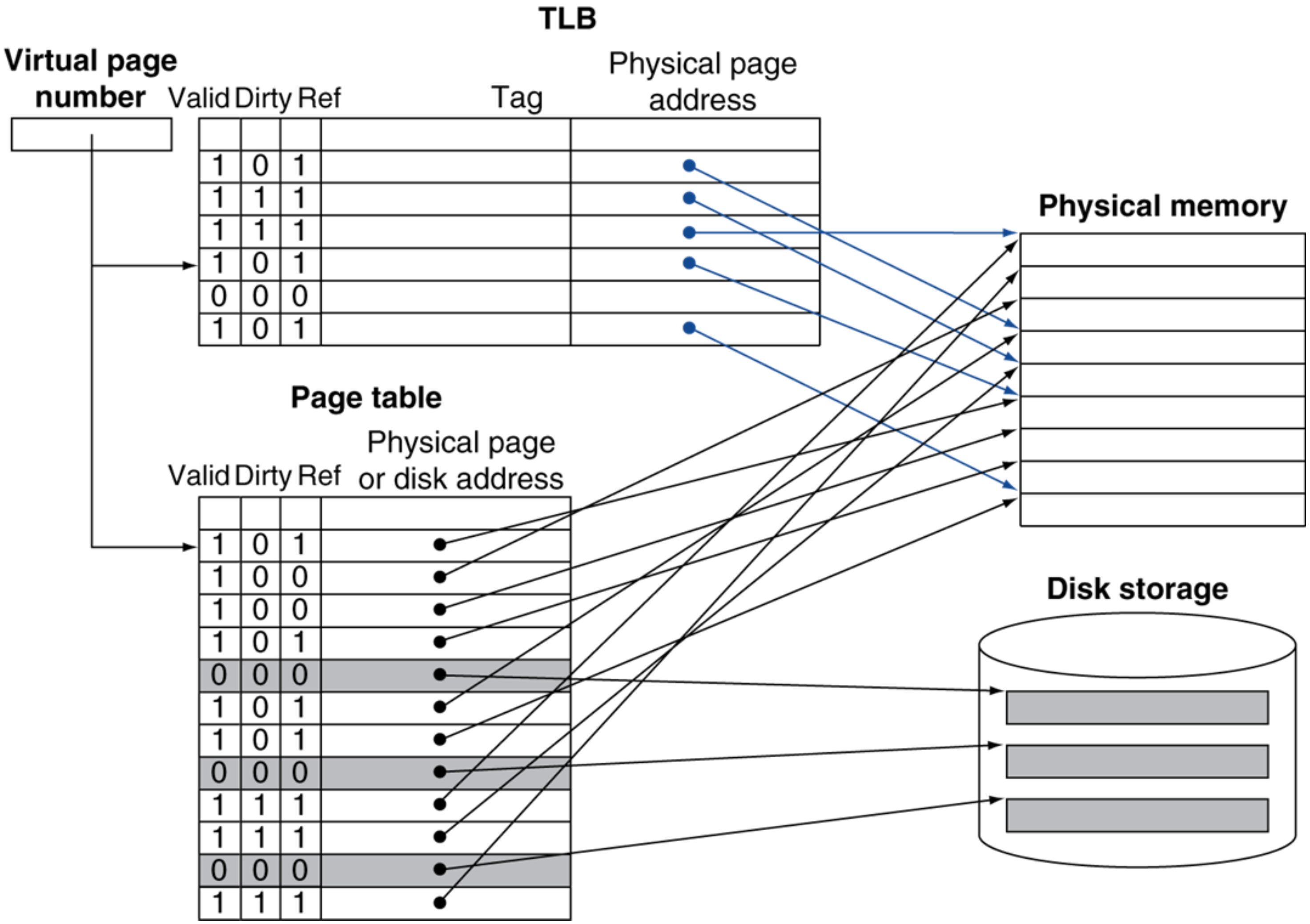


Figure 3

Virtual Memory Review



TLB

Virtual page number

Valid	Dirty	Ref	Tag	Physical page address
1	0	1		●
1	1	1		●
1	1	1		●
1	0	1		●
0	0	0		
1	0	1		●

Page table

Valid	Dirty	Ref	Physical page or disk address
1	0	1	●
1	0	0	●
1	0	0	●
1	0	1	●
0	0	0	●
1	0	1	●
1	0	1	●
0	0	0	●
1	1	1	●
1	1	1	●
0	0	0	●
1	1	1	●

Physical memory

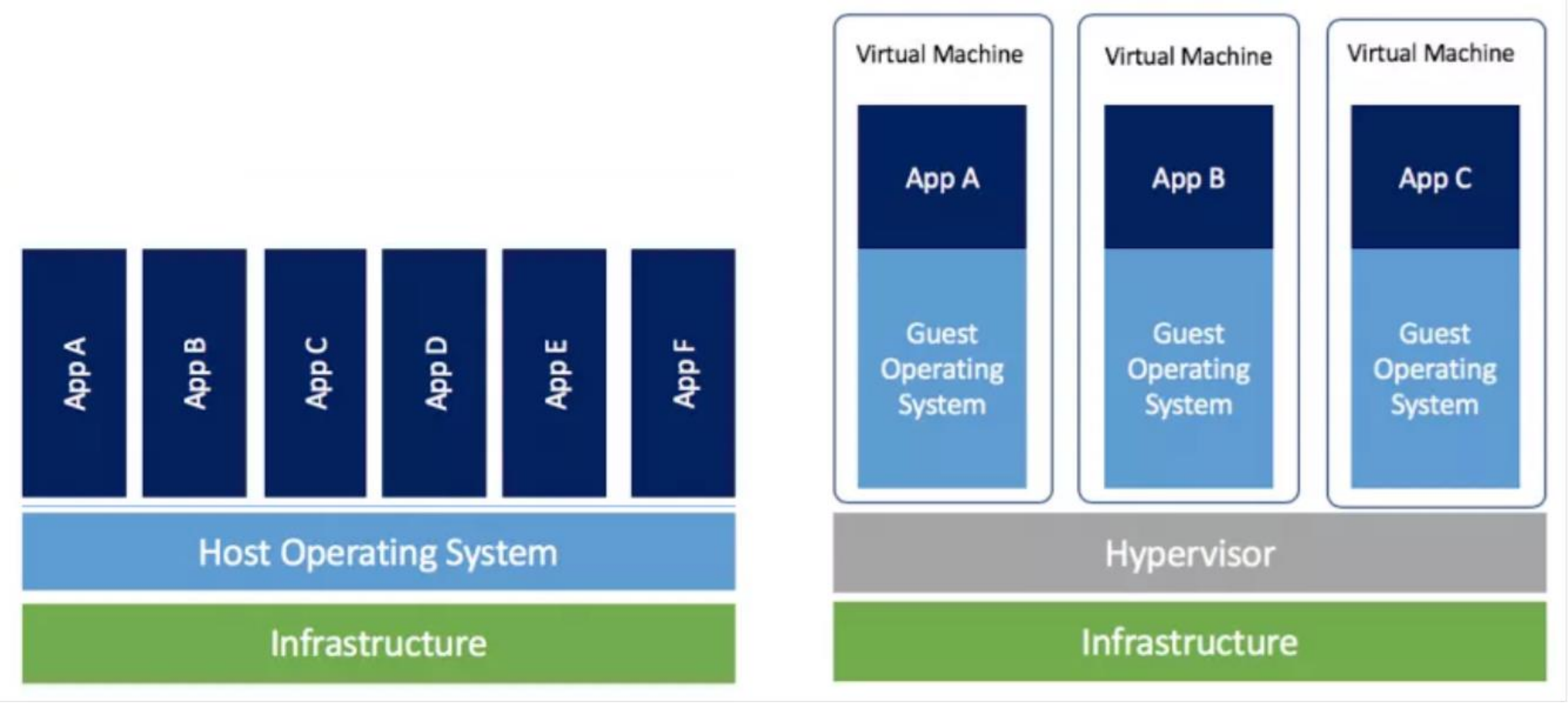
Disk storage

Virtual Memory

- Protection via virtual memory
  - Keeps processes in their own memory space

- Role of architecture
  - Provide user mode and supervisor mode
  - Protect certain aspects of CPU state
  - Provide mechanisms for switching between user mode and supervisor mode
  - Provide mechanisms to limit memory accesses
  - Provide TLB to translate addresses

Virtual Machines Review



Virtual Machines

- Supports isolation and security
- Sharing a computer among many unrelated users
- Enabled by raw speed of processors, making the overhead more acceptable

- Allows different ISAs and operating systems to be presented to user programs
  - “System Virtual Machines”
  - SVM software is called “virtual machine monitor” or “hypervisor”
  - Individual virtual machines run under the monitor are called “guest VMs”

Requirements of VMM

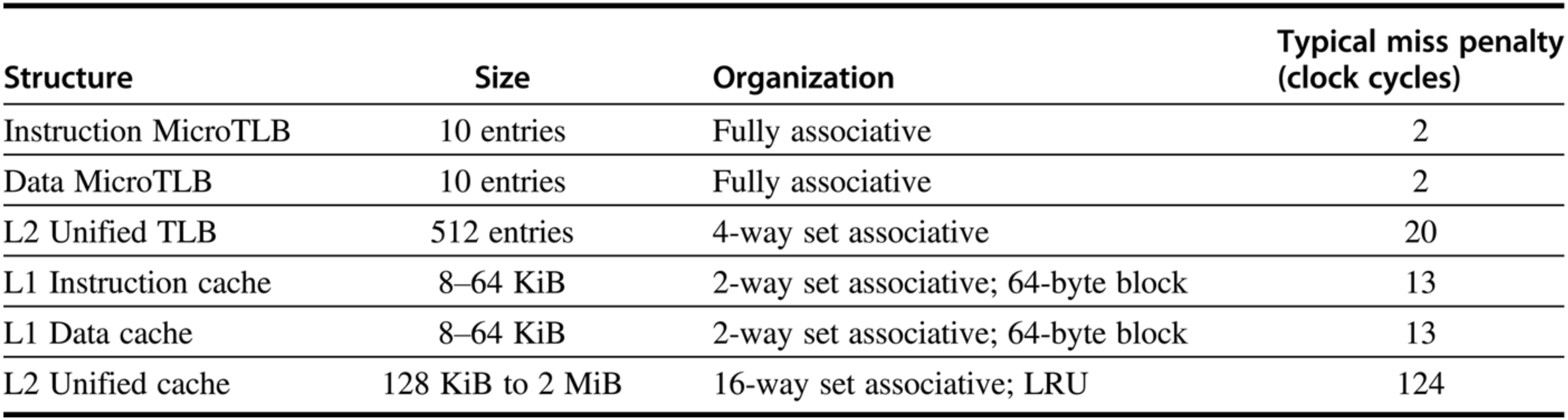
- Guest software should:
  - Behave on as if running on native hardware
  - Not be able to change allocation of real system resources
- VMM should be able to "context switch" guests
- Hardware must allow:
  - System and user processor modes
  - Privileged subset of instructions for allocating system resources

Contents

- Introduction
- Memory Technology and Optimizations
- Ten Advanced Optimizations of Cache Performance
- Virtual Memory and Virtual Machines
- ARM Cortex-A53 and Intel Core i7 6700
- Fallacies and Pitfalls

ARM Cortex-A53

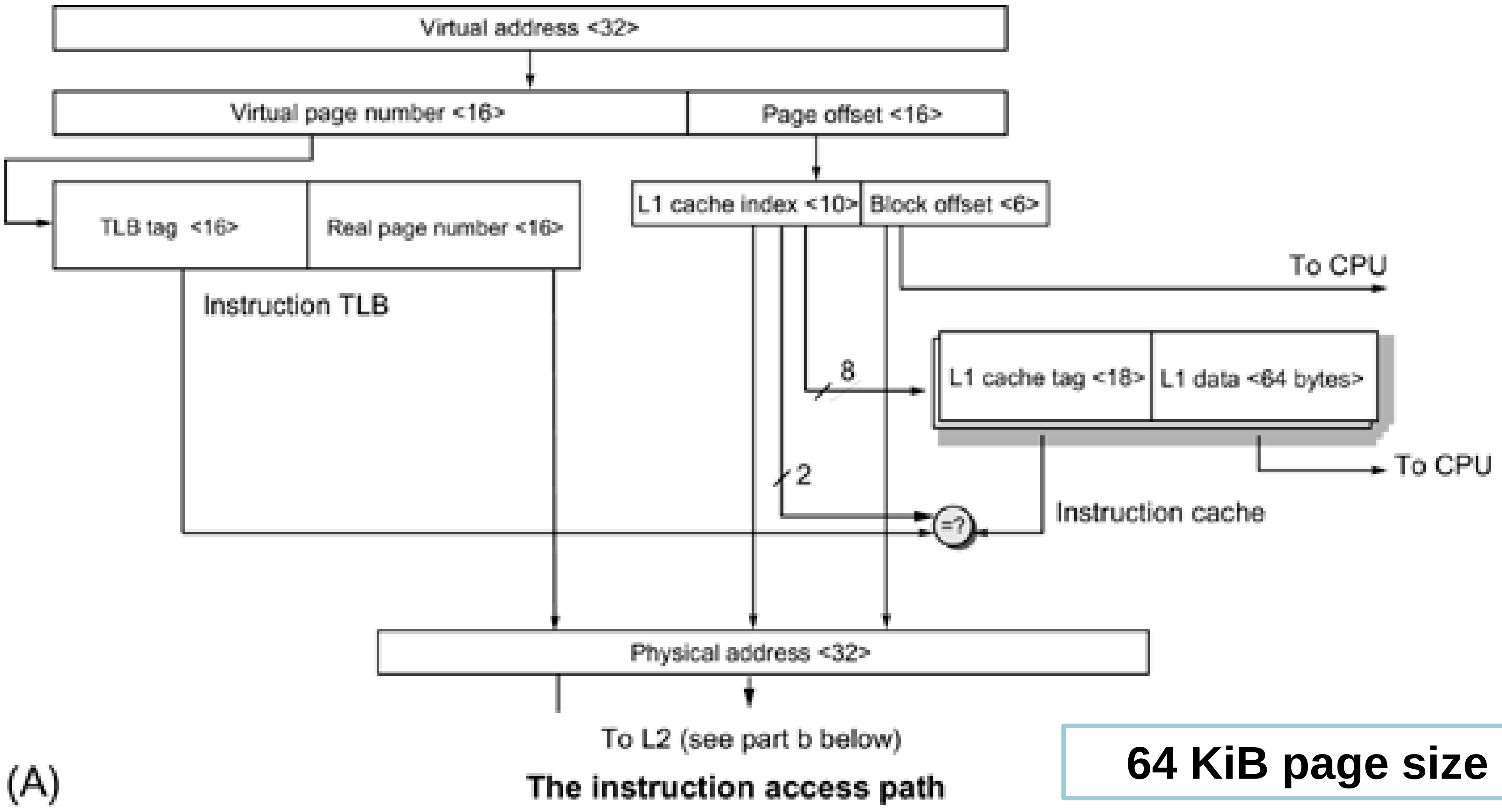
The memory hierarchy of the Cortex A53 includes multilevel TLBs and caches



Structure	Size	Organization	Typical miss penalty (clock cycles)
Instruction MicroTLB	10 entries	Fully associative	2
Data MicroTLB	10 entries	Fully associative	2
L2 Unified TLB	512 entries	4-way set associative	20
L1 Instruction cache	8–64 KiB	2-way set associative; 64-byte block	13
L1 Data cache	8–64 KiB	2-way set associative; 64-byte block	13
L2 Unified cache	128 KiB to 2 MiB	16-way set associative; LRU	124

L1 D$ 2- or 4-way set associative

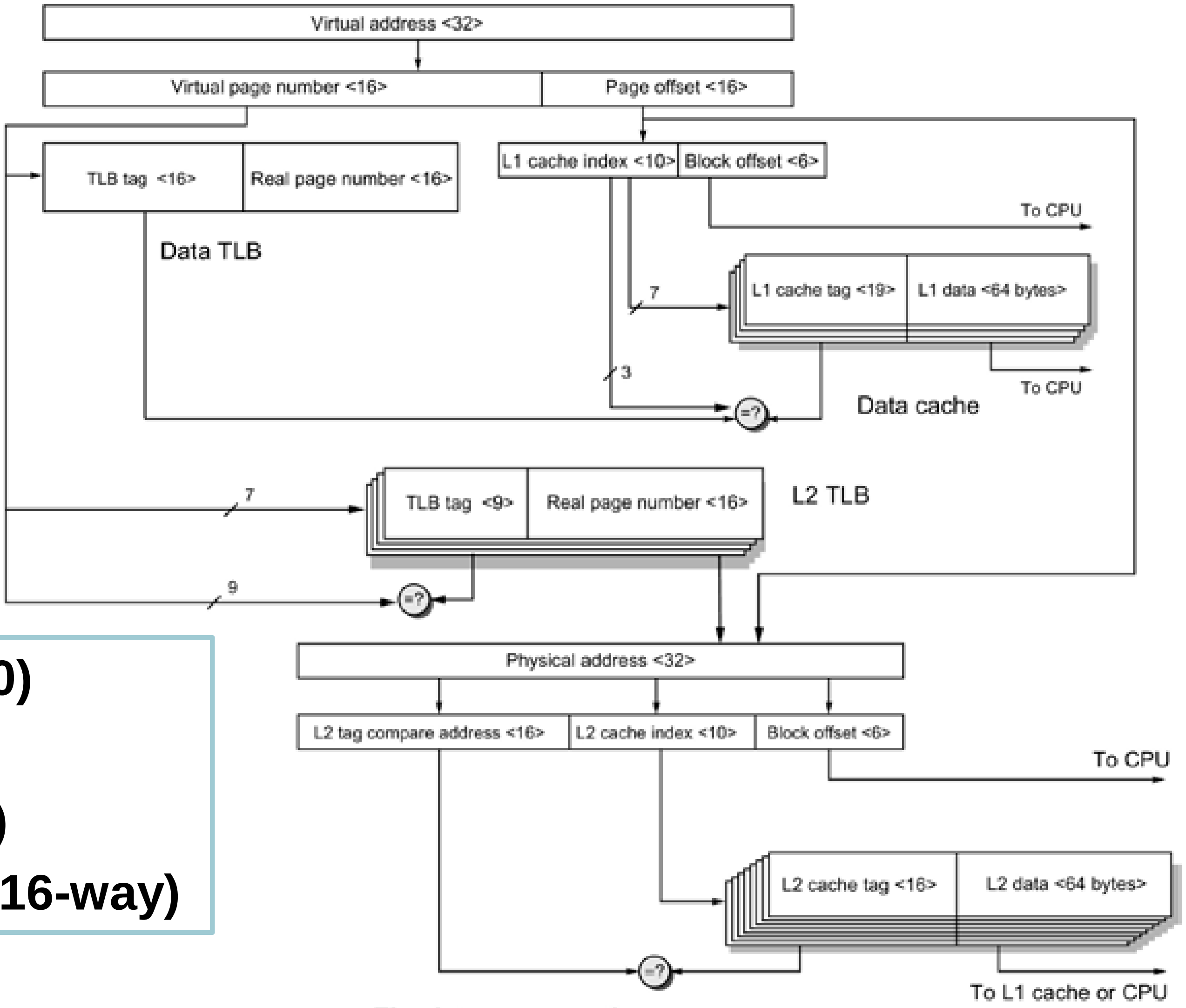
A53 L1 I-TLB (10) & L1 I$ (32 KiB)



The instruction access path

64 KiB page size

A53 L1 D-TLB & D$, L2 TLB & $



L1 D-TLB (10)

D$ (32 KiB)

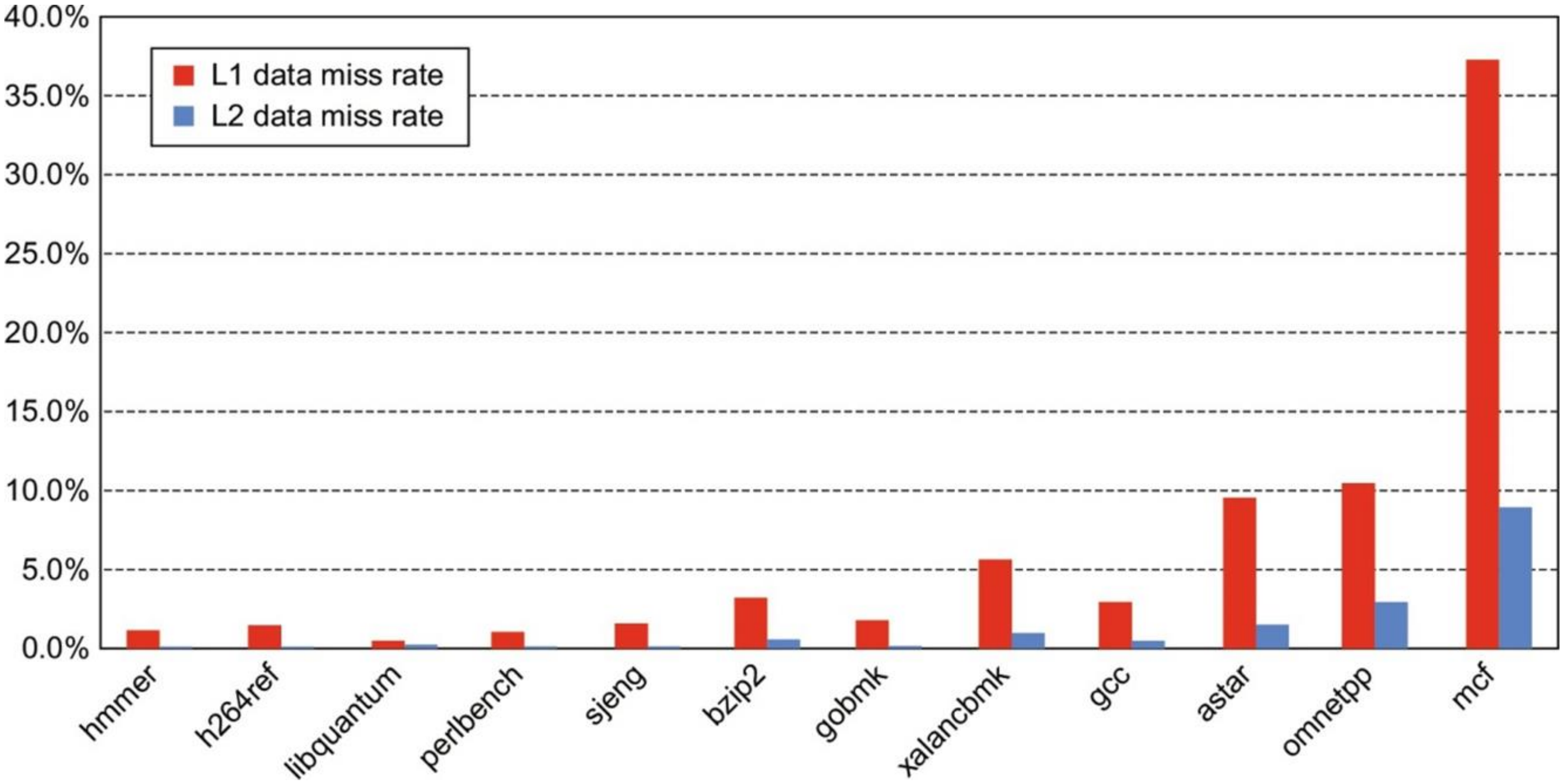
L2 TLB (512)

L2 $ (1 MiB, 16-way)

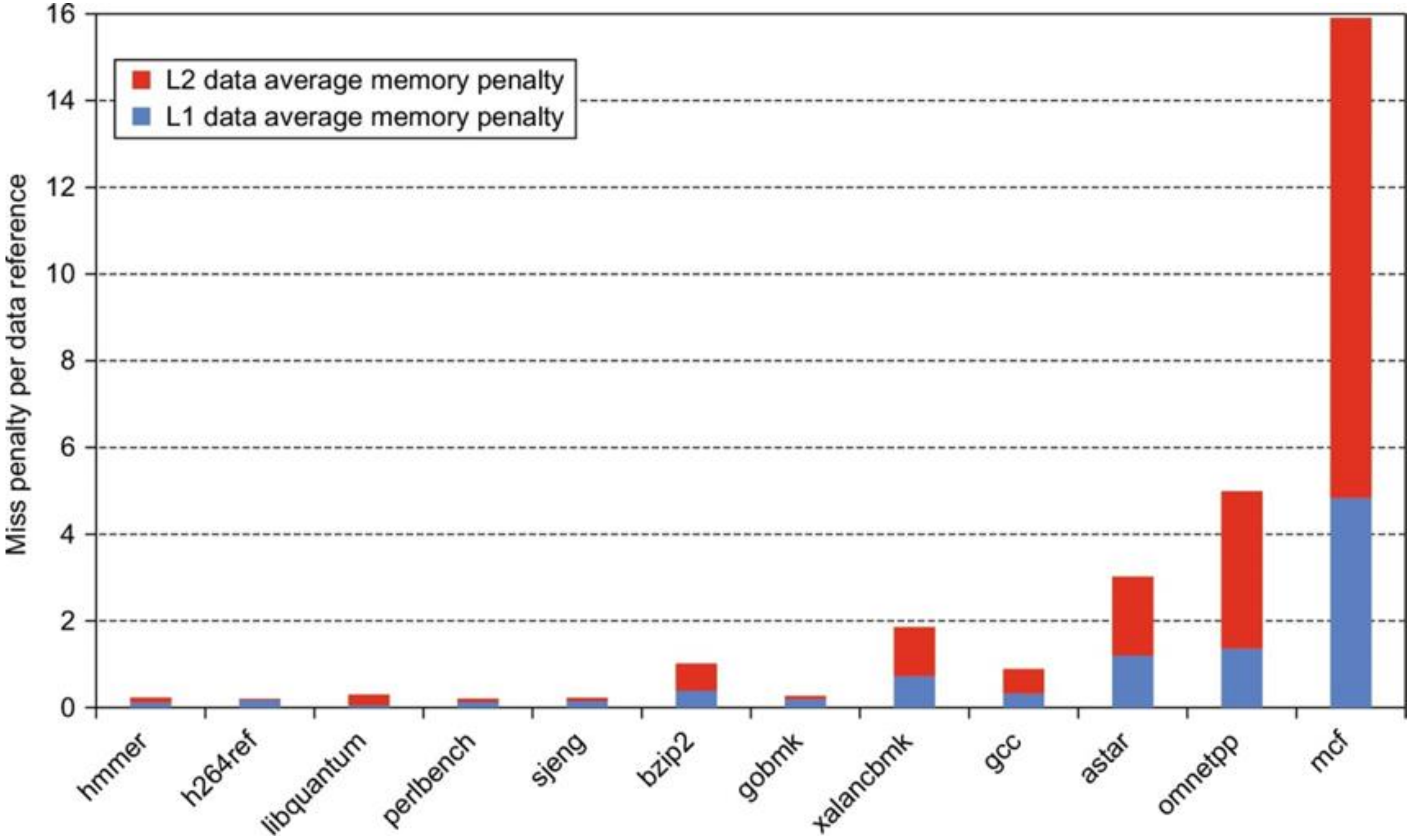
(B)

The data access path

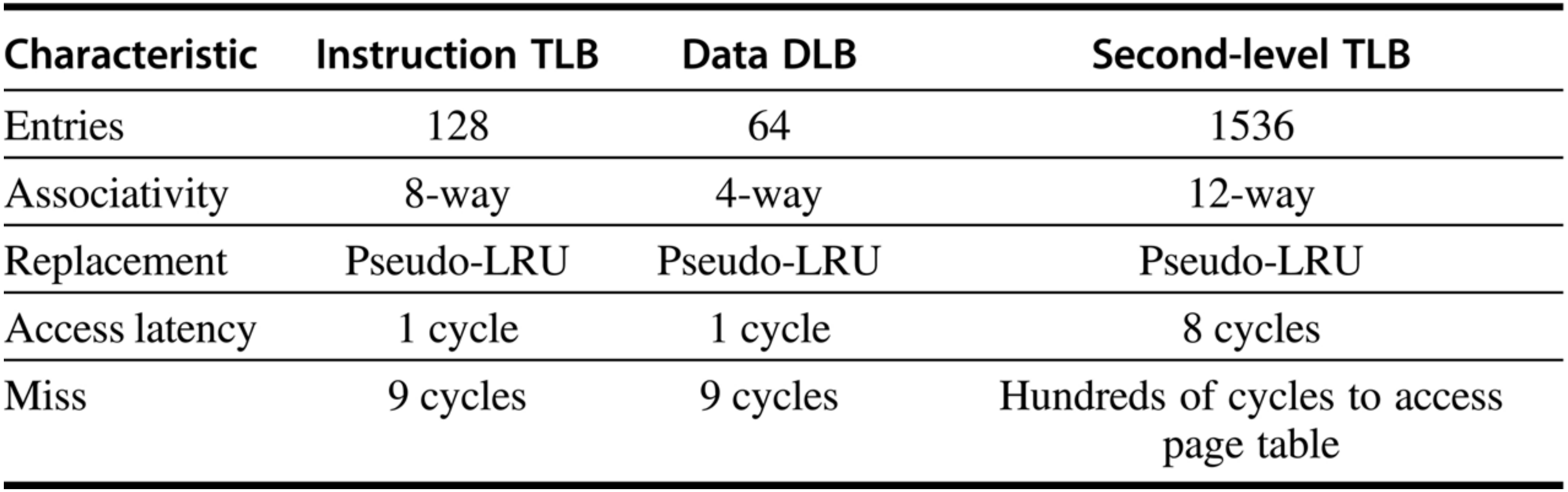
A53 SPECInt2006 Performance



A53 SPECInt2006 Performance

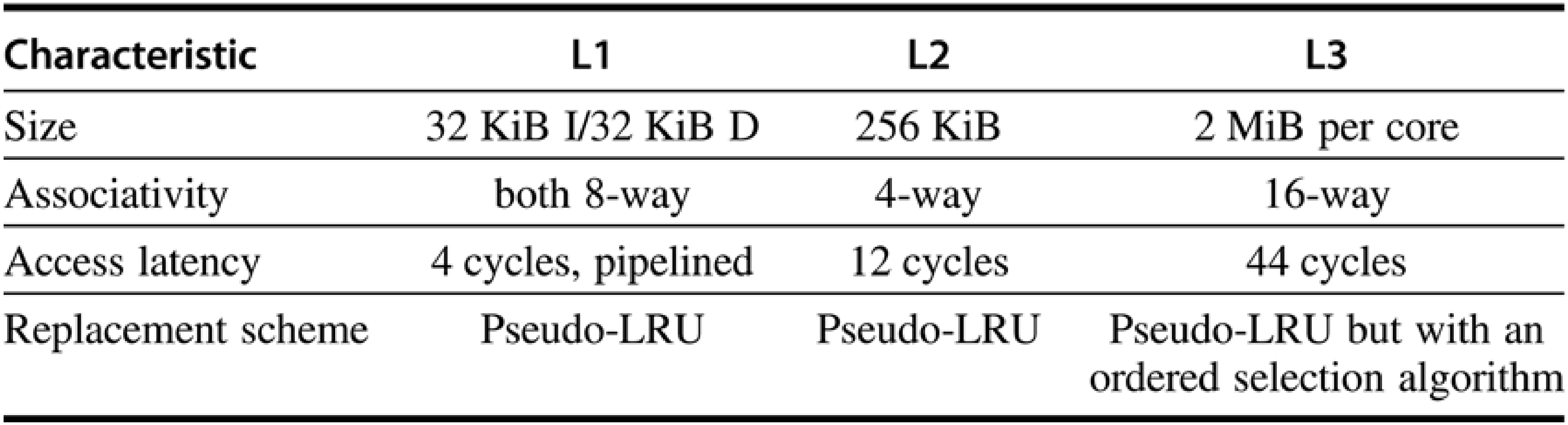


Intel Core i7 6700 – TLBs



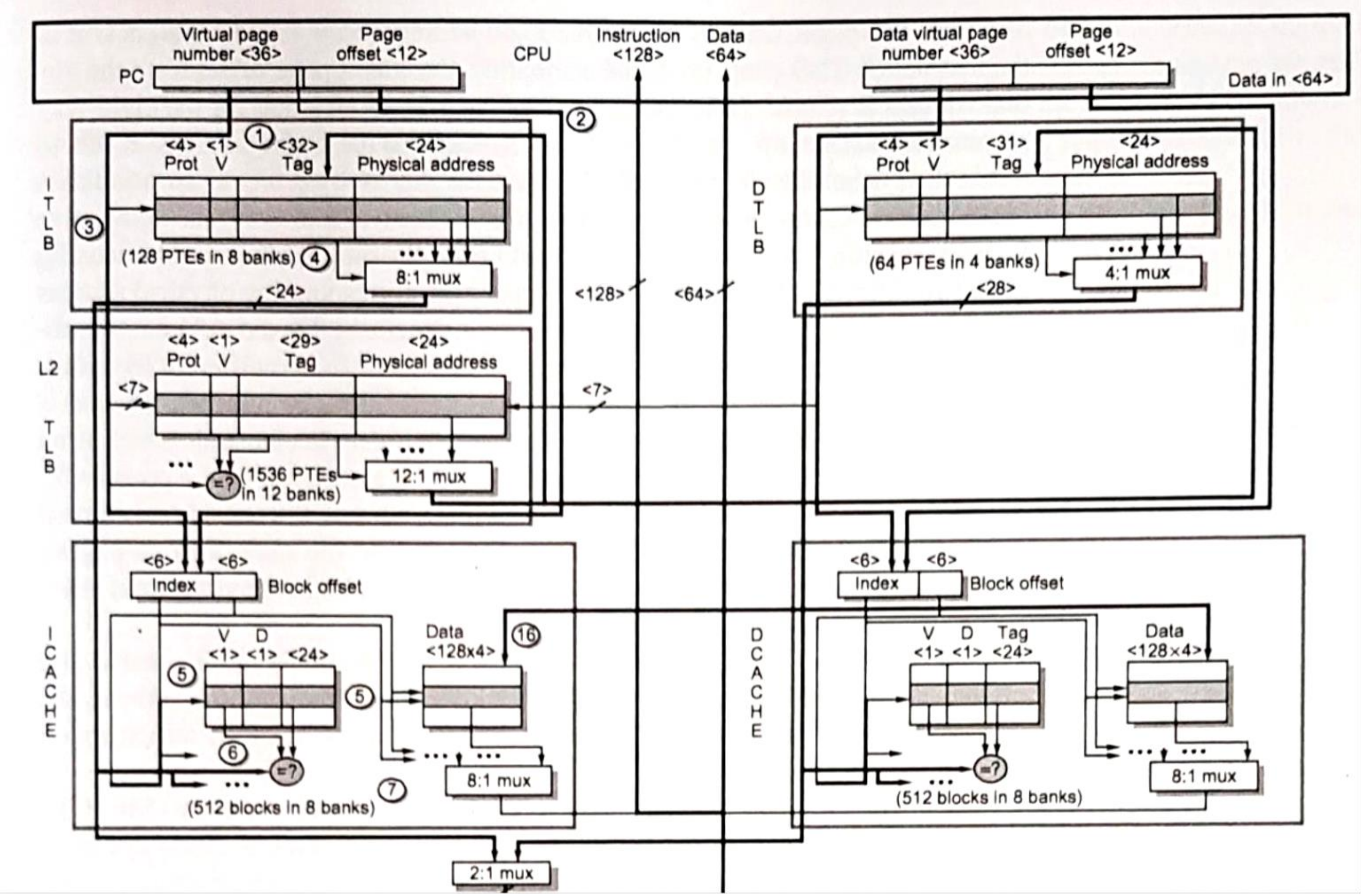
Characteristic	Instruction TLB	Data DLB	Second-level TLB
Entries	128	64	1536
Associativity	8-way	4-way	12-way
Replacement	Pseudo-LRU	Pseudo-LRU	Pseudo-LRU
Access latency	1 cycle	1 cycle	8 cycles
Miss	9 cycles	9 cycles	Hundreds of cycles to access page table

Intel Core i7 6700 – Caches

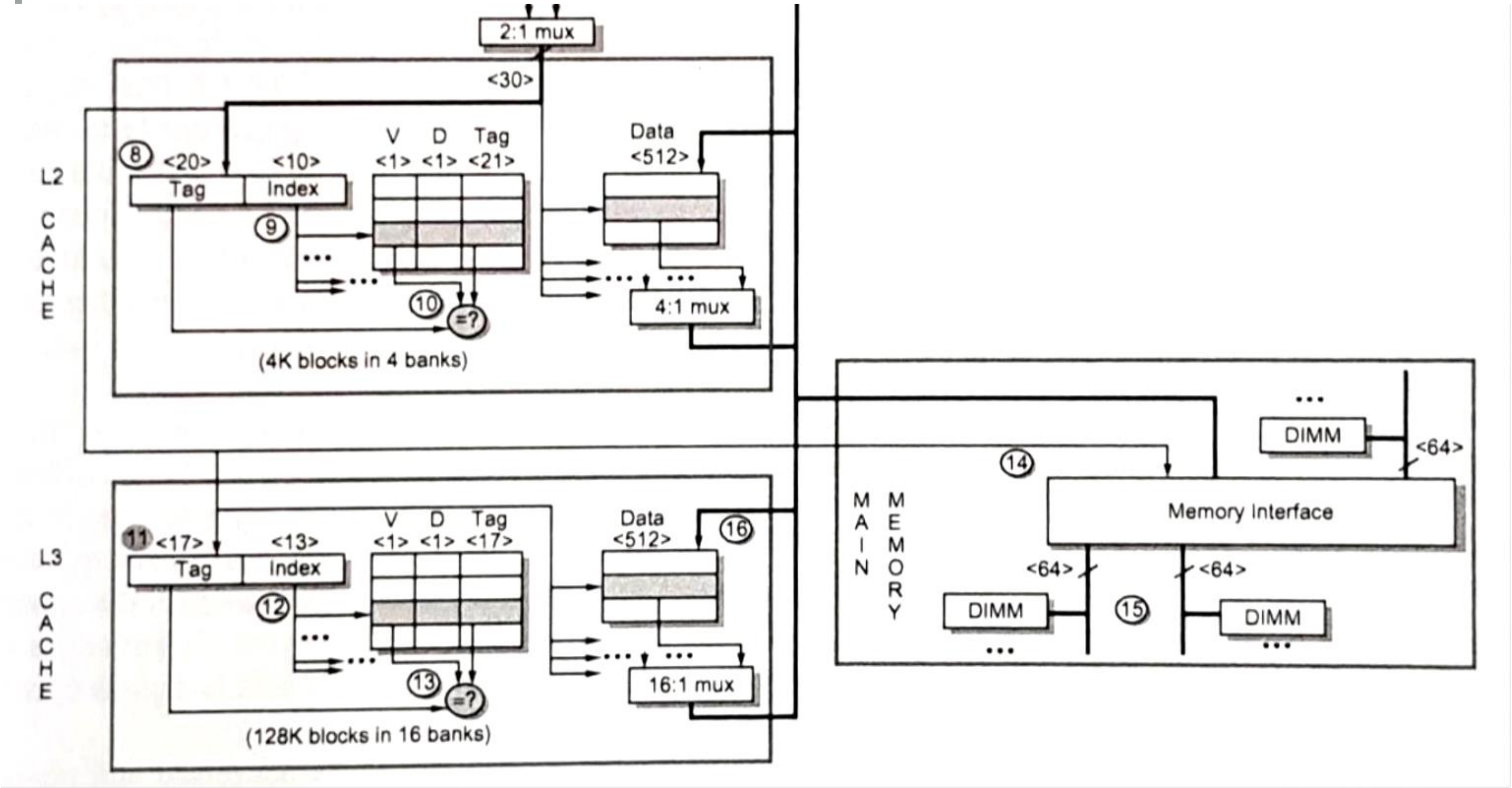


Characteristic	L1	L2	L3
Size	32 KiB I/32 KiB D	256 KiB	2 MiB per core
Associativity	both 8-way	4-way	16-way
Access latency	4 cycles, pipelined	12 cycles	44 cycles
Replacement scheme	Pseudo-LRU	Pseudo-LRU	Pseudo-LRU but with an ordered selection algorithm

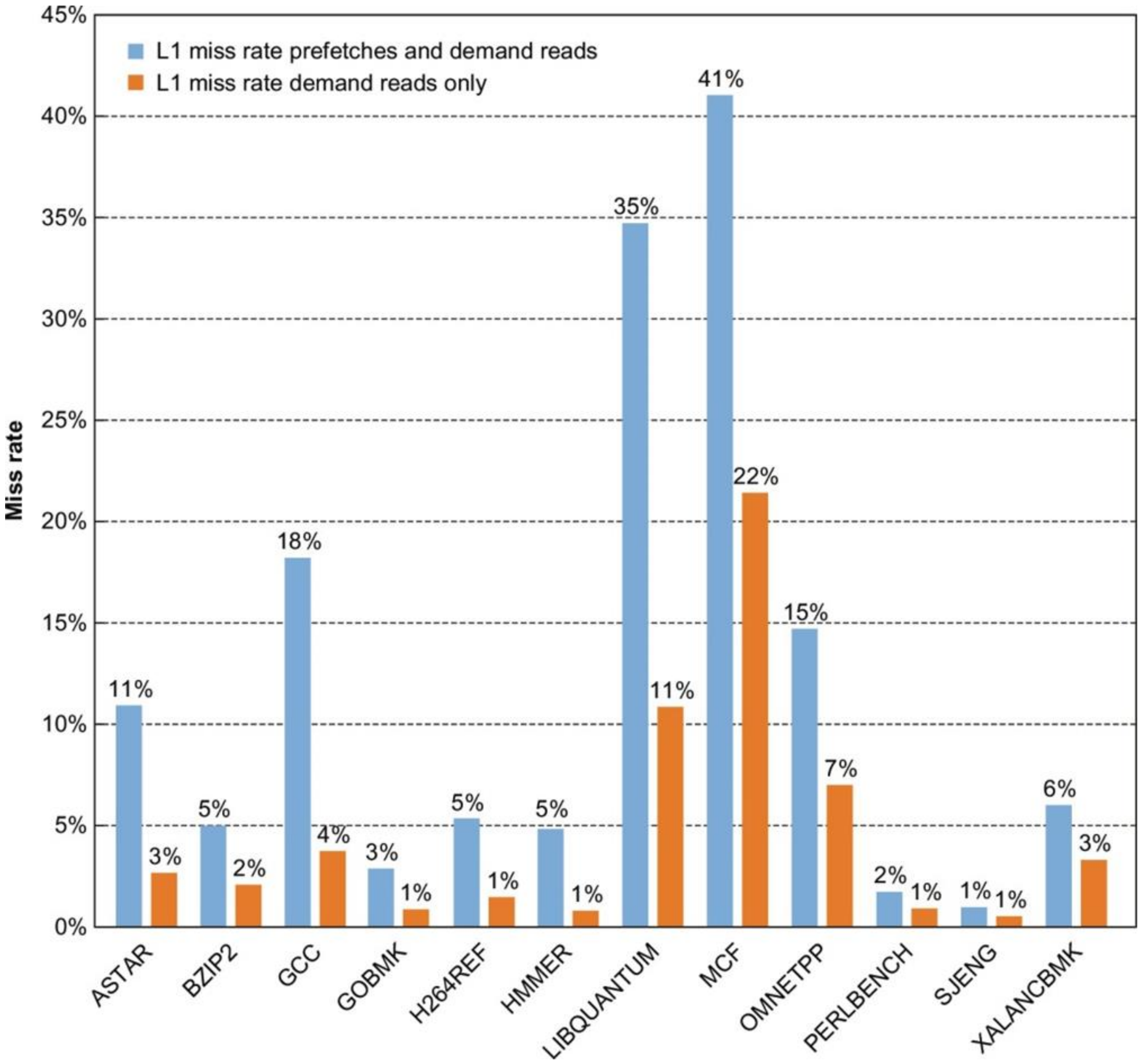
i7 L1 and L2 TLBs and L1 Caches



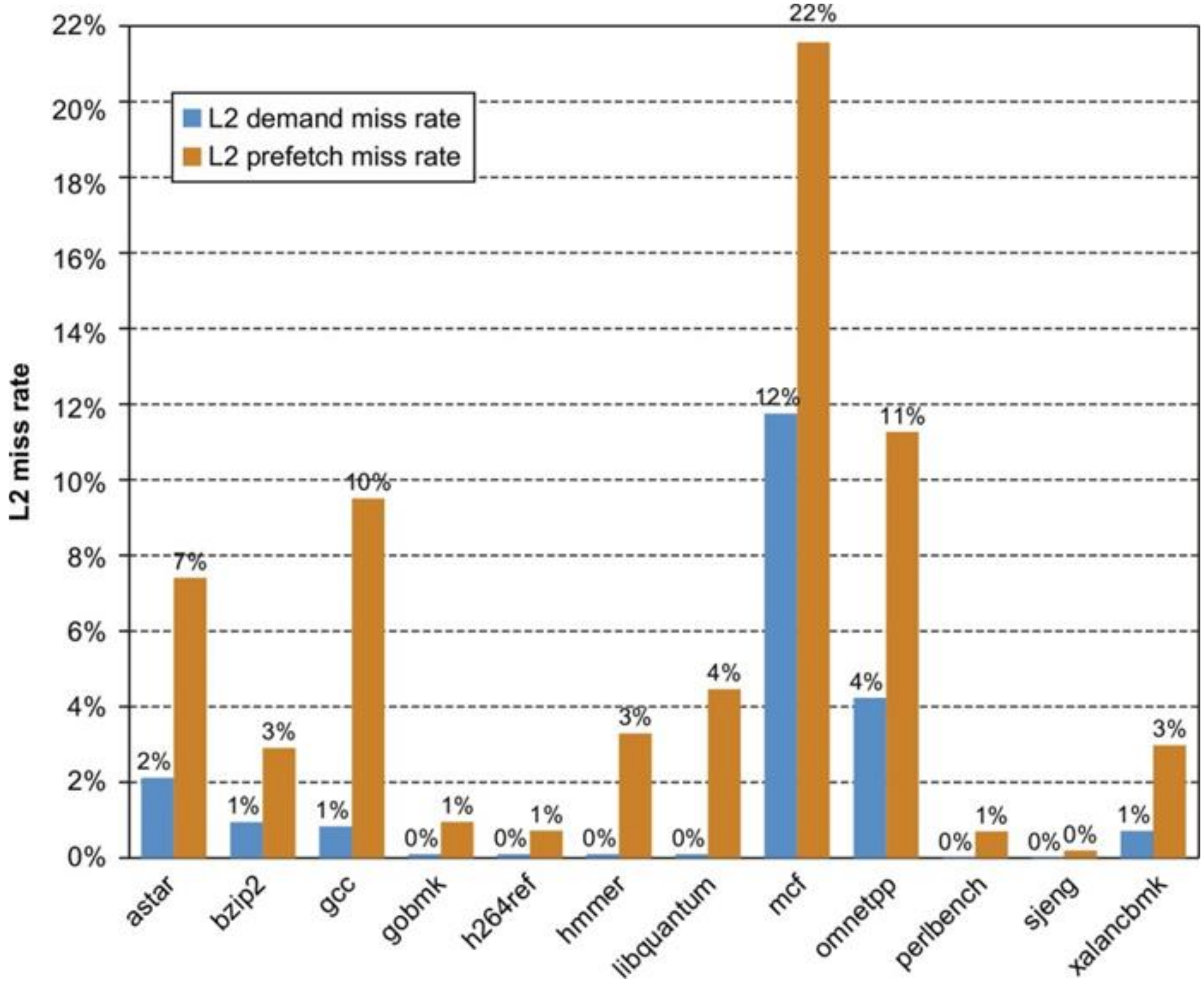
i7 L2 and L3 (8 MiB) Caches



i7 SPECInt2006 L1 Performance



i7 SPECInt2006 L2 Performance

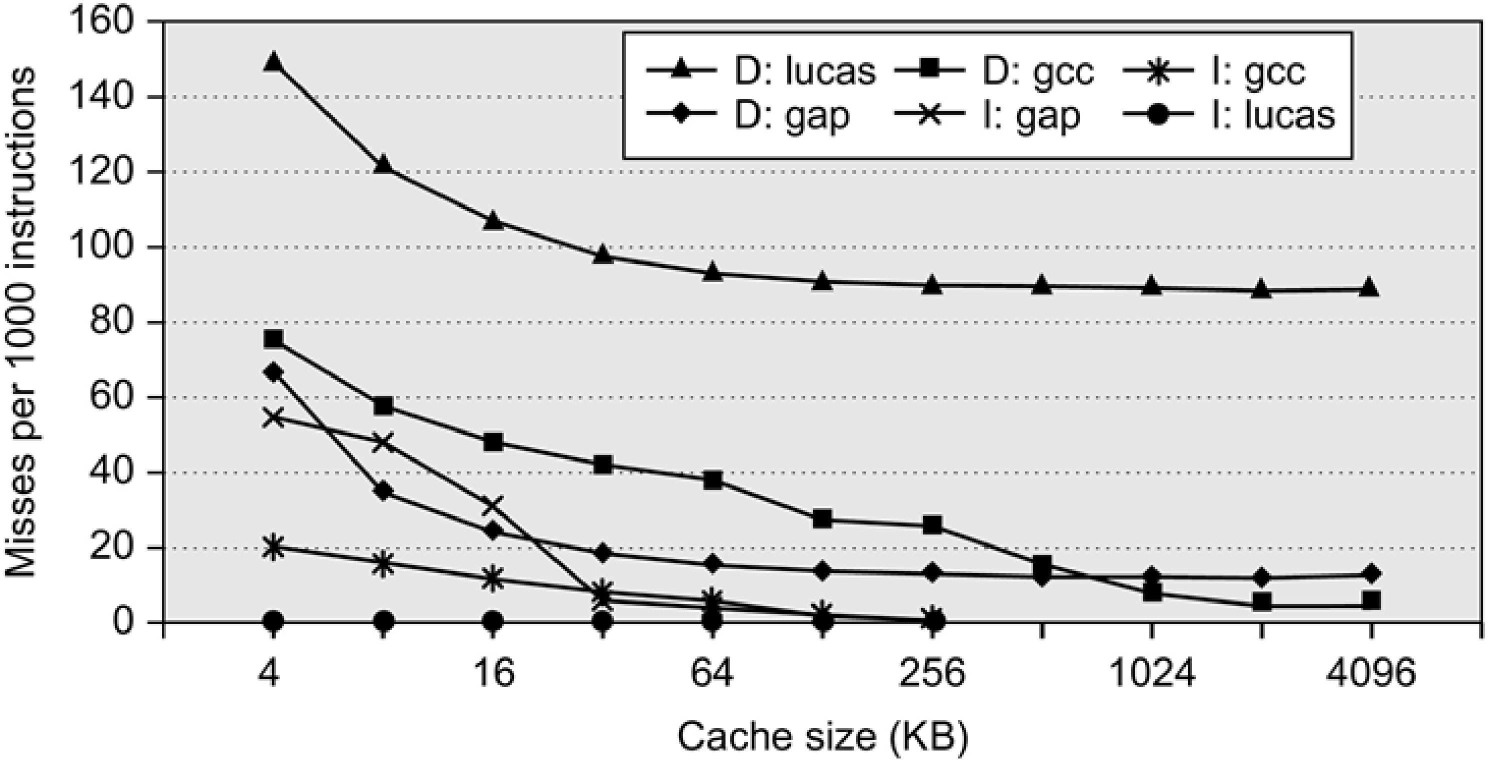


Contents

- Introduction
- Memory Technology and Optimizations
- Ten Advanced Optimizations of Cache Performance
- Virtual Memory and Virtual Machines
- ARM Cortex-A53 and Intel Core i7 6700
- Fallacies and Pitfalls

Fallacies

- F: Predicting cache performance of one program from another



Pitfalls

- P: Simulating enough instructions to get accurate performance measures of the memory hierarchy
- P: Not delivering high memory bandwidth in a cache-based system