

JOHN L. HENNESSY DAVID A. PATTERSON

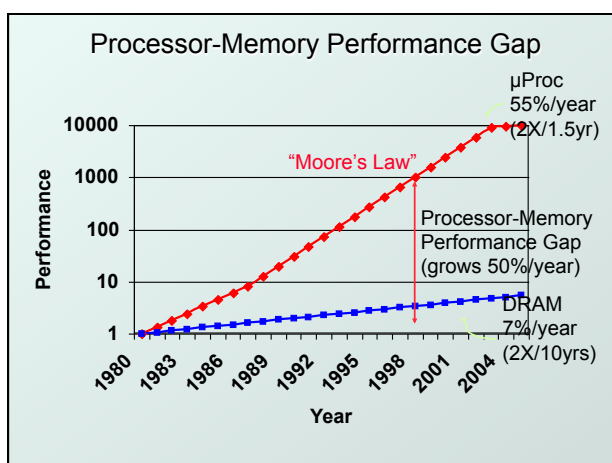
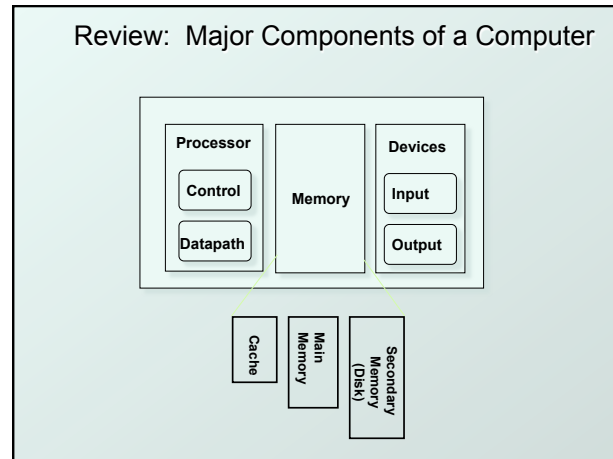
COMPUTER ARCHITECTURE
A Quantitative Approach

FIFTH EDITION

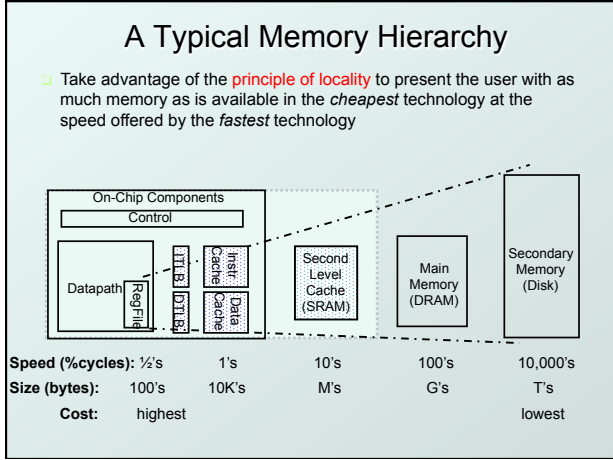
Memory Hierarchy

Original slides from:
Computer Architecture
 A Quantitative Approach
 Hennessy, Patterson

Modified slides by Yashwant Malaiya
 Colorado State University



- ### The Memory Hierarchy Goal
- ◆ Fact: Large memories are slow and fast memories are small
 - ◆ How do we create a memory that gives the illusion of being large, cheap and fast (most of the time)?
 - With hierarchy
 - With parallelism



Memory Technology

- ◆ Static RAM (SRAM)
 - 0.5-2.5ns, 2010: \$2000-\$5000 per GB (2015: same?)
- ◆ Dynamic RAM (DRAM)
 - 50-70ns, 2010: \$20-\$75 per GB (2015: <\$10 per GB)
- ◆ Flash Memory
 - 70-150ns, 2010: \$4-\$12 per GB (2015: \$.14 per GB)
- ◆ Magnetic disk
 - 5ms-20ms, \$0.2-\$2.0 per GB (2015: \$.7 per GB)
- ◆ Ideal memory
 - Access time of SRAM
 - Capacity and cost/GB of disk

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 6

Principle of Locality

- ◆ Programs access a small proportion of their address space at any time
- ◆ Temporal locality
 - Items accessed recently are likely to be accessed again soon
 - e.g., instructions in a loop, induction variables
- ◆ Spatial locality
 - Items near those accessed recently are likely to be accessed soon
 - E.g., sequential instruction access, array data

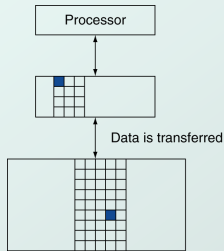
Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 7

Taking Advantage of Locality

- ◆ Memory hierarchy
- ◆ Store everything on disk
- ◆ Copy recently accessed (and nearby) items from disk to smaller DRAM memory
 - Main memory
- ◆ Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
 - Cache memory attached to CPU

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 8

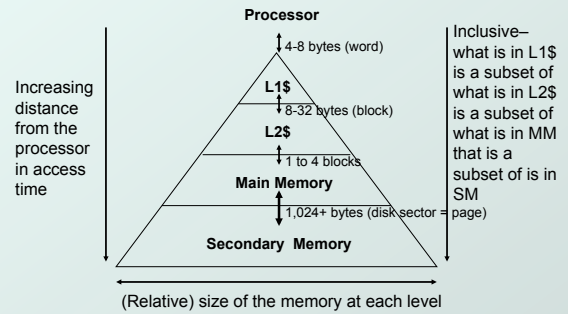
Memory Hierarchy Levels



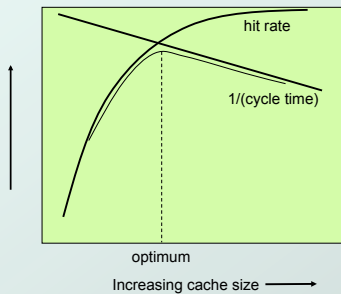
- ◆ **Block (aka line): unit of copying**
 - May be multiple words
- ◆ **If accessed data is present in upper level**
 - Hit: access satisfied by upper level
 - ◆ Hit ratio: hits/accesses
- ◆ **If accessed data is absent**
 - Miss: block copied from lower level
 - ◆ Time taken: miss penalty
 - ◆ Miss ratio: misses/accesses = 1 - hit ratio
 - Then accessed data supplied from upper level

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 9

Characteristics of the Memory Hierarchy



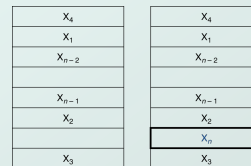
Cache Size



11

Cache Memory

- ◆ **Cache memory**
 - The level of the memory hierarchy closest to the CPU
- ◆ **Given accesses X_1, \dots, X_{n-1}, X_n**



a. Before the reference to X_n b. After the reference to X_n

- How do we know if the data is present?
- Where do we look?

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 12

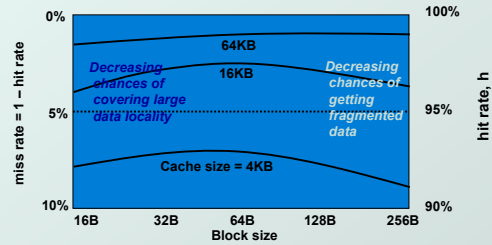
Block Size Considerations

- ◆ Larger blocks should reduce miss rate
 - Due to spatial locality
- ◆ But in a fixed-sized cache
 - Larger blocks \Rightarrow fewer of them
 - ◆ More competition \Rightarrow increased miss rate
 - Larger blocks \Rightarrow *pollution*
- ◆ Larger miss penalty
 - Can override benefit of reduced miss rate
 - *Early restart* and *critical-word-first* can help

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 13

Increasing Hit Rate

- ◆ Hit rate increases with cache size.
- ◆ Hit rate mildly depends on block size.



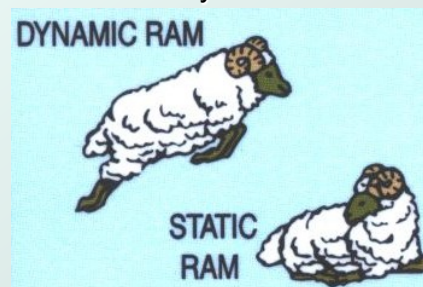
14

Cache Misses

- ◆ On cache hit, CPU proceeds normally
- ◆ On cache miss
 - Stall the CPU pipeline
 - Fetch block from next level of hierarchy
 - Instruction cache miss
 - ◆ Restart instruction fetch
 - Data cache miss
 - ◆ Complete data access

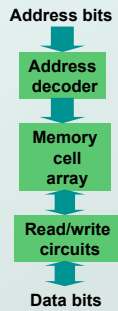
Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 15

Static vs Dynamic RAMs



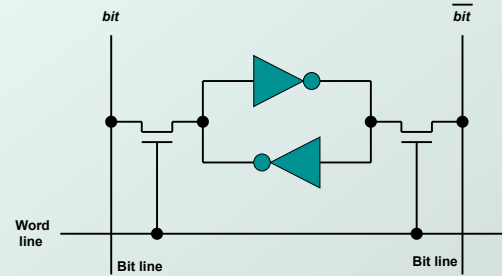
Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 16

Random Access Memory (RAM)



17

Six-Transistor SRAM Cell

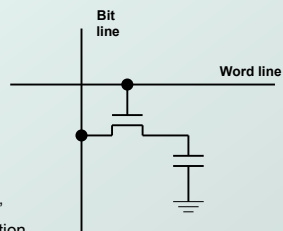


18

Dynamic RAM (DRAM) Cell



"Single-transistor DRAM cell"
Robert Dennard's 1967 invention



19

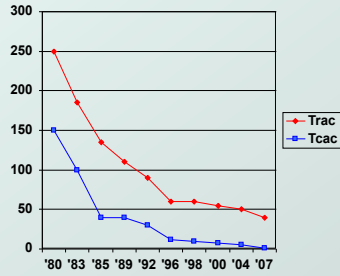
Advanced DRAM Organization

- ◆ 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

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 20

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



Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 21

Average Access Time

- ◆ Hit time is also important for performance
- ◆ Average memory access time (AMAT)
 - $AMAT = Hit\ time + Miss\ rate \times Miss\ penalty$
- ◆ Example
 - CPU with 1ns clock, hit time = 1 cycle, miss penalty = 20 cycles, l-cache miss rate = 5%
 - $AMAT = 1 + 0.05 \times 20 = 2ns$
 - ◆ 2 cycles per instruction

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 22

Performance Summary

- ◆ When CPU performance increased
 - Miss penalty becomes more significant
- ◆ Can't neglect cache behavior when evaluating system performance

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 23

Multilevel Caches

- ◆ Primary cache attached to CPU
 - Small, but fast
- ◆ Level-2 cache services misses from primary cache
 - Larger, slower, but still faster than main memory
- ◆ Main memory services L-2 cache misses
- ◆ Some high-end systems include L-3 cache

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 24

Interactions with Advanced CPUs

- ◆ Out-of-order CPUs can execute instructions during cache miss
 - Pending store stays in load/store unit
 - Dependent instructions wait in reservation stations
 - ◆ Independent instructions continue
- ◆ Effect of miss depends on program data flow
 - Much harder to analyse
 - Use system simulation

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 25

Virtual Memory

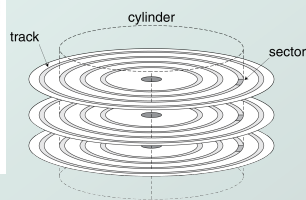
- ◆ Use main memory as a “cache” for secondary (disk) storage
 - Managed jointly by CPU hardware and the operating system (OS)
- ◆ Programs share main memory
 - Each gets a private virtual address space holding its frequently used code and data
 - Protected from other programs
- ◆ CPU and OS translate virtual addresses to physical addresses
 - VM “block” is called a page
 - VM translation “miss” is called a page fault

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 25

§5.4 Virtual Memory

Disk Storage

- ◆ Nonvolatile, rotating magnetic storage



Chapter 6 — Storage and Other I/O Topics — 27

§6.3 Disk Storage

Disk Sectors and Access

- ◆ Each sector records
 - Sector ID
 - Data (512 bytes, 4096 bytes proposed)
 - Error correcting code (ECC)
 - ◆ Used to hide defects and recording errors
 - Synchronization fields and gaps
- ◆ Access to a sector involves
 - Queuing delay if other accesses are pending
 - Seek: move the heads
 - Rotational latency
 - Data transfer
 - Controller overhead

Chapter 6 — Storage and Other I/O Topics — 28

Disk Access Example

- ◆ Given
 - 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk
- ◆ Average read time
 - 4ms seek time
 - $+ \frac{1}{2} / (15,000/60) = 2\text{ms}$ rotational latency
 - $+ 512 / 100\text{MB/s} = 0.005\text{ms}$ transfer time
 - + 0.2ms controller delay
 - = 6.2ms
- ◆ If actual average seek time is 1ms
 - Average read time = 3.2ms

Chapter 6 — Storage and Other I/O Topics — 29

Disk Performance Issues

- ◆ Manufacturers quote average seek time
 - Based on all possible seeks
 - Locality and OS scheduling lead to smaller actual average seek times
- ◆ Smart disk controller allocate physical sectors on disk
 - Present logical sector interface to host
 - SCSI, ATA, SATA
- ◆ Disk drives include caches
 - Prefetch sectors in anticipation of access
 - Avoid seek and rotational delay

Chapter 6 — Storage and Other I/O Topics — 30

Flash Storage

- ◆ Nonvolatile semiconductor storage
 - 100× – 1000× faster than disk
 - Smaller, lower power, more robust
 - But more \$/GB (between disk and DRAM)



Chapter 6 — Storage and Other I/O Topics — 31

§6.4 Flash Storage

Flash Types

- ◆ NOR flash: bit cell like a NOR gate
 - Random read/write access
 - Used for instruction memory in embedded systems
- ◆ NAND flash: bit cell like a NAND gate
 - Denser (bits/area), but block-at-a-time access
 - Cheaper per GB
 - Used for USB keys, media storage, ...
- ◆ Flash bits wears out after 1000's of accesses
 - **Not suitable for direct RAM or disk replacement**
 - Wear leveling: remap data to less used blocks

Chapter 6 — Storage and Other I/O Topics — 32

Virtual vs. Physical Address

- ◆ Processor assumes a certain memory addressing scheme:
 - A block of data is called a virtual page
 - An address is called virtual (or logical) address
- ◆ Main memory may have a different addressing scheme:
 - Real memory address is called a physical address, MMU translates virtual address to physical address
 - Complete address translation table is large and must therefore reside in main memory
 - MMU contains TLB (translation lookaside buffer), which is a small cache of the address translation table

33

Page Fault Penalty

- ◆ On page fault, the page must be fetched from disk
 - Takes millions of clock cycles
 - Handled by OS code
- ◆ Try to minimize page fault rate
 - Smart replacement algorithms

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 34

Memory Protection

- ◆ Different tasks can share parts of their virtual address spaces
 - But need to protect against errant access
 - Requires OS assistance
- ◆ Hardware support for OS protection
 - Privileged supervisor mode (aka kernel mode)
 - Privileged instructions
 - Page tables and other state information only accessible in supervisor mode
 - System call exception (e.g., syscall in MIPS)

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 35

The Memory Hierarchy

The BIG Picture

- ◆ Common principles apply at all levels of the memory hierarchy
 - Based on notions of caching
- ◆ At each level in the hierarchy
 - Block placement
 - Finding a block
 - Replacement on a miss
 - Write policy

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 36

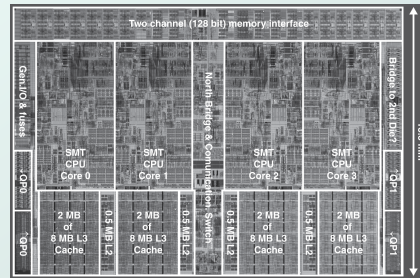
§5.5 A Common Framework for Memory Hierarchies

Virtual Machines

- Host computer emulates guest operating system and machine resources
 - Improved isolation of multiple guests
 - Avoids security and reliability problems
 - Aids sharing of resources
- Virtualization has some performance impact
 - Feasible with modern high-performance computers
- Examples
 - IBM VM/370 (1970s technology!)
 - VMWare
 - Microsoft Virtual PC

Multilevel On-Chip Caches

Intel Nehalem 4-core processor



Per core: 32KB L1 I-cache, 32KB L1 D-cache, 512KB L2 cache

3-Level Cache Organization

	Intel Nehalem	AMD Opteron X4
L1 caches (per core)	L1 I-cache: 32KB, 64-byte blocks, 4-way, approx LRU replacement, hit time n/a L1 D-cache: 32KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a	L1 I-cache: 32KB, 64-byte blocks, 2-way, LRU replacement, hit time 3 cycles L1 D-cache: 32KB, 64-byte blocks, 2-way, LRU replacement, write-back/allocate, hit time 9 cycles
L2 unified cache (per core)	256KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a	512KB, 64-byte blocks, 16-way, approx LRU replacement, write-back/allocate, hit time n/a
L3 unified cache (shared)	8MB, 64-byte blocks, 16-way, replacement n/a, write-back/allocate, hit time n/a	2MB, 64-byte blocks, 32-way, replace block shared by fewest cores, write-back/allocate, hit time 32 cycles

n/a: data not available

Concluding Remarks

- Fast memories are small, large memories are slow
 - We really want fast, large memories ☹️
 - Caching gives this illusion 😊
- Principle of locality
 - Programs use a small part of their memory space frequently
- Memory hierarchy
 - L1 cache ↔ L2 cache ↔ ... ↔ DRAM memory ↔ disk
- Memory system design is critical for multiprocessors