

EEC 485 High Performance Arch. (Fall 2007)

Chapter 7 Memory Hierarchy

Chansu Yu

Cleveland State University

Table of Contents

- Ch.1, 4 Introduction & Performance
 - Ch. 2 Instruction: Machine Language
 - 3.1 Introduction
 - 3.3 Operands
 - 3.5 Control flow operations
 - 3.7 Beyond numbers
 - 3.9 Starting a program
 - 3.2 Operations (arithmetic, memory operations)
 - 3.4 Representing instructions
 - 3.6 Supporting procedures
 - 3.8 MIPS addressing
 - 3.10 Example
 - Ch. 3 CPU Implementation: Arithmetic
 - 4.1 Introduction
 - 4.3 Addition and subtraction
 - 4.5 Constructing an Arithmetic Logic Unit (ALU)
 - 4.6 Multiplication
 - 4.8 Floating point
 - 4.2 Signed and unsigned numbers
 - 4.4 Logical operations
 - 4.7 Division
 - 4.9-4.14 Etc.
 - Ch. 5 CPU Implementation: All others
 - 5.1 Introduction
 - 5.3 Implementation Scheme
 - 5.5 Microprogramming: Simplifying Control Design
 - 5.6 Exceptions
 - 5.2 Building a Datapath
 - 5.4 A Multicycle Implementation
 - 5.7-5.12 Etc.
 - Ch. 6-9 Advanced topics
- Software interface*
- Hardware interface*
- Key parts
of EEC 483**
- Ch.6 Pipelining
Ch.7 Cache
=> Issue is
"High Performance"*

What's Ahead?

- ❑ Ch. 6: Pipeline <- CPU
 - ❑ Ch. 7: Cache <- Memory
 - ❑ Ch. 8: Input, Output <- I/O
 - ❑ Ch. 9: Multiprocessor
- + Network
= Modern Computer
- * Network is part of I/O and briefly discussed in Ch. 8 but the issues are more extensive than CA itself. That's why we have a separate course on computer networks.
- Multiple CPUs + Memory + I/O or Multiple (CPU + Memory + I/O)s
 - The question is how to interconnect them.
 - Multi-CPU: cache becomes a BIG problem
 - Multi-computer: networking becomes a BIG problem

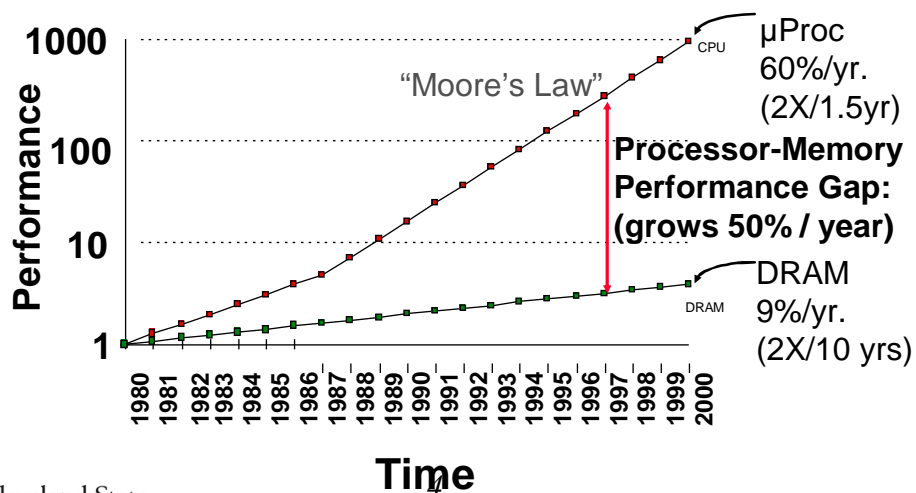
Cleveland State University

3

c.yu91@csuohio.edu

Who Cares About the Memory Hierarchy?

Processor-DRAM Memory Gap (latency)



Cleveland State University

c.yu91@csuohio.edu

Motivation

- Processor-memory performance gap
 - Memory speeds improving, but more slowly than processor speeds
 - Memory speeds have become a primary bottleneck
 - Fast memory is expensive

- Main techniques
 - Parallelism
 - hardware-level access to multiple data at once
 - Memory hierarchy
 - use a small fast memory (cache) and a large slow memory
 - techniques to hide layering

Technology Trends (from 1st lecture)

	Capacity	Speed (latency)
Logic:	2x in 3 years	2x in 3 years
DRAM:	4x in 3 years	2x in 10 years
Disk:	4x in 3 years	2x in 10 years

DRAM		
Year	Size	Cycle Time
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1995	64 Mb	120 ns

1000:1! (between 1980 and 1995 Size)
2:1! (between 1980 and 1995 Cycle Time)

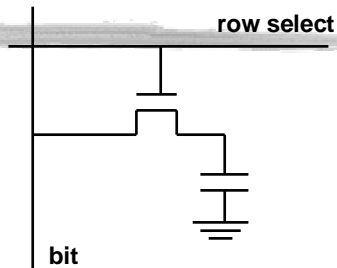
As of 2004 (page 469),
 SRAM 0.5~5 ns \$4~10K/GB
 DRAM 50~70 ns \$100~200
 HDD 5M~20M ns \$0.5~2

Main Memory Background

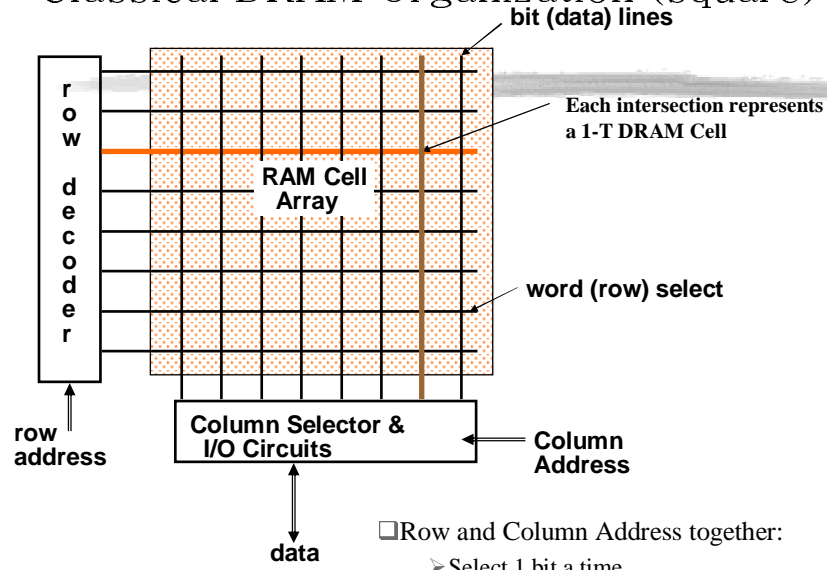
- Performance of Main Memory:
 - Latency: Cache Miss Penalty
 - *Access Time*: time between request and word arrives
 - *Cycle Time*: time between requests
 - Bandwidth: I/O & Large Block Miss Penalty (L2)
- Main Memory is *DRAM*: Dynamic Random Access Memory
 - Dynamic since needs to be refreshed periodically (8 ms)
 - Addresses divided into 2 halves (Memory as a 2D matrix):
 - *RAS* or *Row Access Strobe*
 - *CAS* or *Column Access Strobe*
- Cache uses *SRAM*: Static Random Access Memory
 - No refresh (6 transistors/bit vs. 1 transistor /bit)
 - Address not divided
- Size: DRAM/SRAM - 4-8, Cost/Cycle time: SRAM/DRAM - 8-16

1-Transistor Memory Cell (DRAM)

- Write:
 - 1. Drive bit line
 - 2. Select row
- Read:
 - 1. Precharge bit line to V_{dd}
 - 2. Select row
 - 3. Cell and bit line share charges
 - Very small voltage changes on the bit line
 - 4. Sense (fancy sense amp)
 - Can detect changes of ~1 million electrons
 - 5. Write: restore the value
- Refresh
 - 1. Just do a dummy read to every cell.



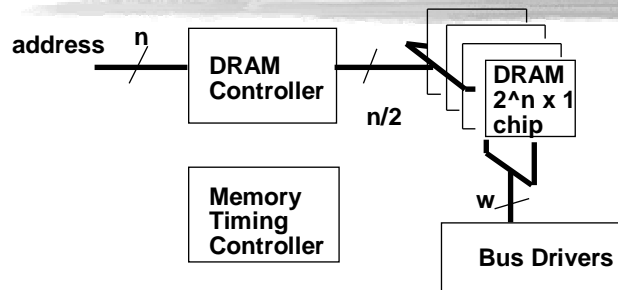
Classical DRAM Organization (square)



Cleveland State University

c.yu91@csuohio.edu

Memory Systems



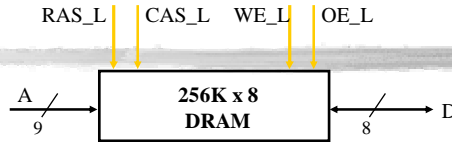
$$T_c = T_{\text{cycle}} + T_{\text{controller}} + T_{\text{driver}}$$

Cleveland State University

10

c.yu91@csuohio.edu

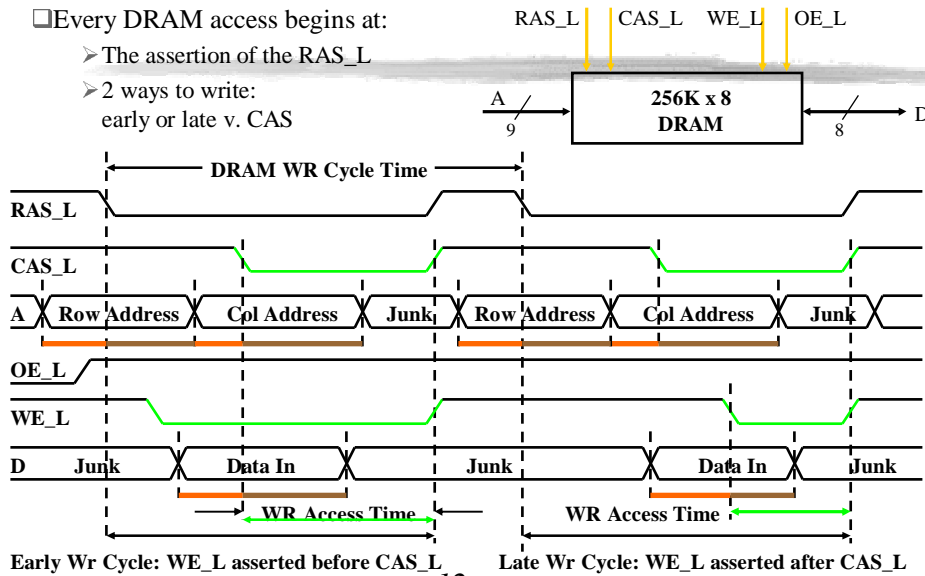
Logic Diagram of a Typical DRAM



- Control Signals (RAS_L, CAS_L, WE_L, OE_L) are all active low
- Din and Dout are combined (D):
 - WE_L is asserted (Low), OE_L is disasserted (High)
 - D serves as the data input pin
 - WE_L is disasserted (High), OE_L is asserted (Low)
 - D is the data output pin
- Row and column addresses share the same pins (A)
 - RAS_L goes low: Pins A are latched in as row address
 - CAS_L goes low: Pins A are latched in as column address
 - RAS/CAS edge-sensitive

DRAM Write Timing

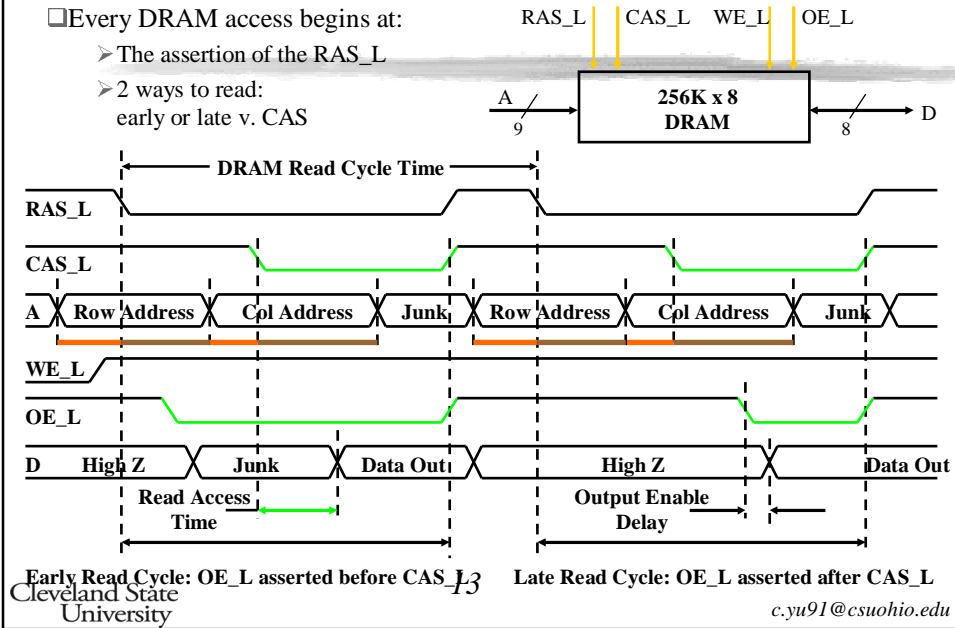
- Every DRAM access begins at:
 - The assertion of the RAS_L
 - 2 ways to write: early or late v. CAS



DRAM Read Timing

□ Every DRAM access begins at:

- The assertion of the RAS_L
- 2 ways to read: early or late v. CAS



Main Memory Background (Again)

□ Performance of Main Memory:

- Latency: Cache Miss Penalty
 - Access Time: time between request and word arrives
 - Cycle Time: time between requests
- Bandwidth: I/O & Large Block Miss Penalty (L2)

□ Main Memory is *DRAM*: Dynamic Random Access Memory

- Dynamic since needs to be refreshed periodically (8 ms)
- Addresses divided into 2 halves (Memory as a 2D matrix):
 - RAS or Row Access Strobe
 - CAS or Column Access Strobe

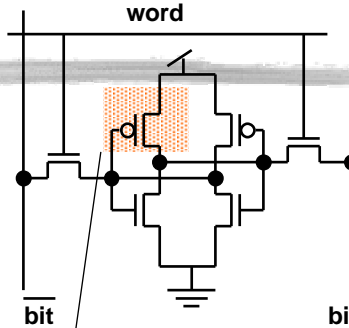
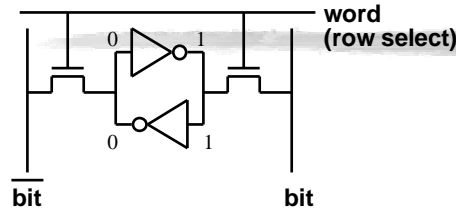
□ Cache uses *SRAM*: Static Random Access Memory

- No refresh (6 transistors/bit vs. 1 transistor /bit)
- Address not divided

□ Size: DRAM/SRAM - 4-8, Cost/Cycle time: SRAM/DRAM - 8-16

Static RAM Cell

6-Transistor SRAM Cell



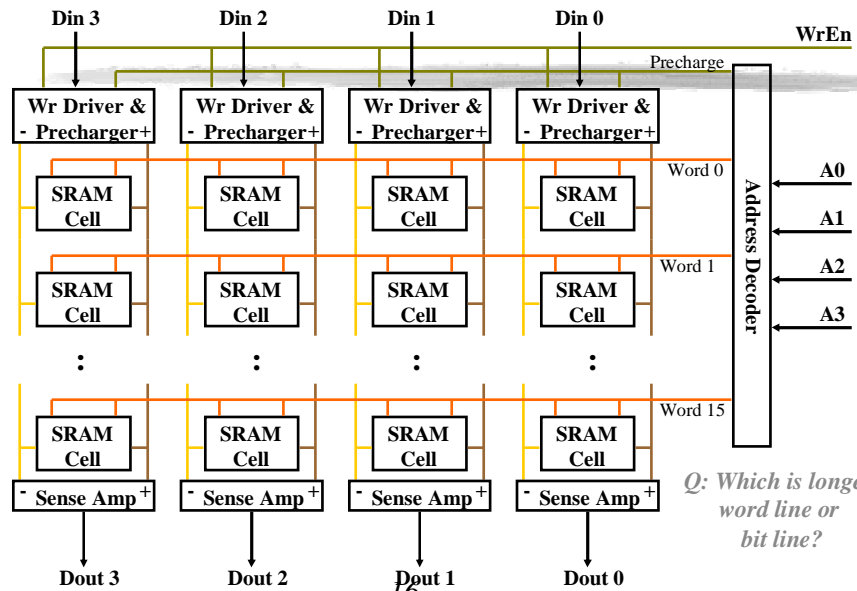
Write:

1. Drive bit lines (bit=1, $\overline{\text{bit}}=0$)
- 2.. Select row

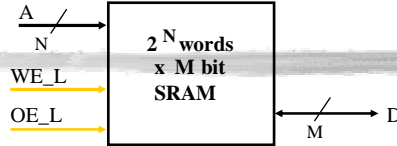
Read:

1. Precharge bit and $\overline{\text{bit}}$ to Vdd
- 2.. Select row
3. Cell pulls one line low
4. Sense amp on column detects difference between bit and $\overline{\text{bit}}$

Typical SRAM Organization: 16-word x 4-bit

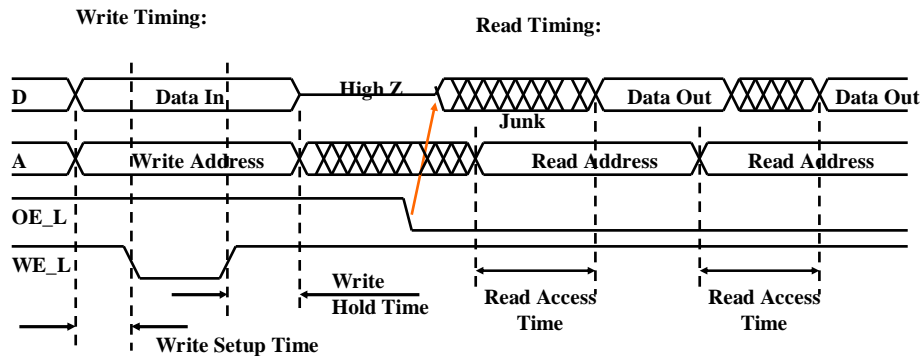
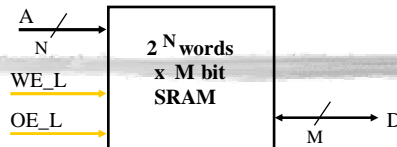


Logic Diagram of a Typical SRAM



- Write Enable is usually active low (WE_L)
- Din and Dout are combined to save pins:
 - A new control signal, output enable (OE_L) is needed
 - WE_L is asserted (Low), OE_L is disasserted (High)
 - D serves as the data input pin
 - WE_L is disasserted (High), OE_L is asserted (Low)
 - D is the data output pin
 - Both WE_L and OE_L are asserted:
 - Result is unknown. Don't do that!!!
- Although could change VHDL to do what desire, must do the best with what you've got (vs. what you need)

Typical SRAM Timing



Comparison

- ❑ SRAM (Cache) vs. DRAM (Memory)
 - SRAM
 - value is stored on a pair of inverting gates
 - very fast but takes up more space than DRAM (4 to 6 transistors)
 - DRAM
 - value is stored as a charge on capacitor (must be refreshed)
 - very small but slower than SRAM (factor of 5 to 10)
- ❑ Cache vs. CPU Registers
 - cache is transparent to the instruction set
 - cache managed by hardware, registers by software
 - cache stores both instruction and data, register stores data only
 - memory ???

Memory Hierarchy

- ❑ A typical memory hierarchy of today:
 - Registers
 - On-chip SRAM cache
 - Off-chip SRAM cache
 - DRAM memory
 - Local magnetic hard disk
 - Fileserver (possibly RAID), i.e., networked magnetic hard disks
 - Magnetic tape
- ❑ Several levels of memory hierarchy
 - CPU Registers -> Cache -> Memory -> Virtual memory -> Disk
 - Cost -----> cheap
 - Access time -----> slow
 - Size -----> large

Impact on Performance

□ Example

- Suppose a processor executes at clock rate of 1 GHz (cycle time 1 ns)
- Software
 - CPI = 1
 - 30% memory instructions (load or store), 70% etc. (arithmetic, logic, control)
- Hardware
 - SRAM (cache) 1ns (1 cycle)
 - DRAM (memory) 50 ns (50 cycles)

□ Without cache

- CPI for memory instructions = $1 + 50 = 51.1$ cycles, CPI for etc. = 1
- Average CPI = $1 + 0.3 * 50 = 16$ cycles
- 94% (15/16) of the time the processor is stalled waiting for memory!

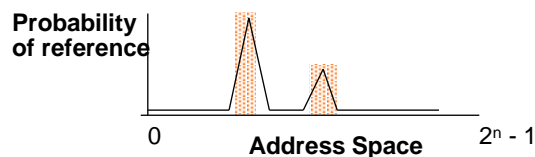
□ With cache (miss rate 10% and 5%)

- CPI for memory instructions = $1 + 50 * \text{cache miss rate}$, CPI for etc. = 1
- Average CPI = $1 + 0.3 * 50 * \text{miss (10\%)} = 2.5$ cycles
- 60% (1.5/2.5) of the time the processor is stalled waiting for memory!
- But, cache speeds up the processor about 6.4 (16/2.5)!
- If Miss rate=5%, Avg CPI = $1 + 0.3 * 50 * 0.05 = 1.75$ cycles => speed up 9.1!

Why Hierarchy works

□ The Principle of Locality:

- Program access a relatively small portion of the address space at any instant of time.



Locality

- ❑ If an item is referenced,

temporal locality: it will tend to be referenced again soon

spatial locality: nearby items will tend to be referenced soon.

Why does code have locality?

Code: loops, recursion, commonly called functions

Data: main data structures

Code: sequential code, small loops

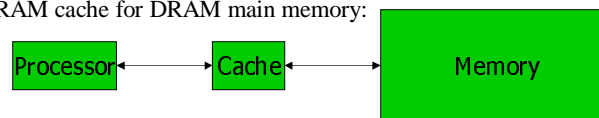
Data: array elements

???

Caches: The Basic Idea

- ❑ A smaller set of storage locations storing a subset of information from a larger set.

➤ Typically, SRAM cache for DRAM main memory:



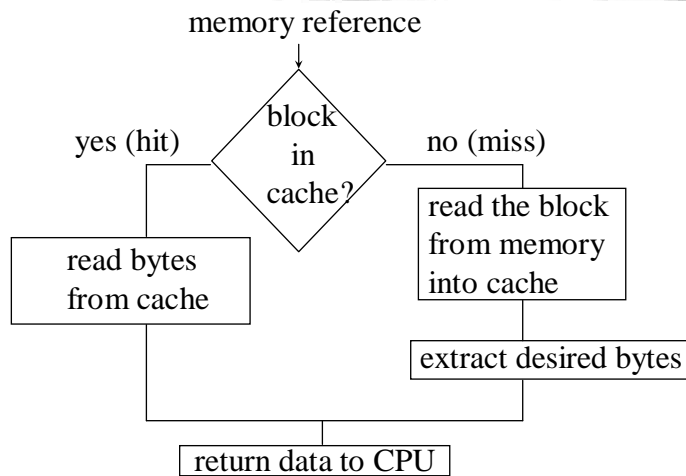
- Goal: Decrease average time for data access.
- Use: Look in cache for data. Look in larger storage only if not found in cache.
- Invisible to programmer.

- ❑ Multiple ways to organize – we will see several.
- ❑ For simplicity, assume all memory accesses are word-sized.

Terminology

- ❑ Cache holds code/data that will “probably” be accessed later
- ❑ Later access is really found in cache: “Hit”
or
it is not found in cache: “Miss” } 100%
- ❑ “Hit time” : cache access time
- ❑ “Miss penalty” : memory access time
 - Bring the data into cache since it will probably be accessed later
 - Bring the following data altogether (“block” or “line” size)
 - Need to throw away the oldest block in the cache (“replacement”)

Caches: Flowchart



Hits vs. Misses

- ❑ Read hits
 - this is what we want!
- ❑ Read misses
 - stall the CPU, fetch block from memory, deliver to cache, restart
- ❑ Write hits:
 - can replace data in cache and memory (write-through)
 - write the data only into the cache (write-back the cache later)
- ❑ Write misses:
 - read the entire block into the cache, then write the word

Cache Performance

- ❑ Describing performance:
 - Hit time = time to access on hit.
 - Usually, read hit time = write hit time.
 - SRAM caches: a few cycles
 - Miss penalty = additional time to access on miss.
 - Usually, read miss penalty ≈ write miss penalty.
 - DRAM main memory: 10s-100s cycles
 - Hit ratio = #hits / #accesses
 - Miss ratio = #misses / #accesses
- ❑ Measuring performance:
 - Lots of benchmarking & simulations.