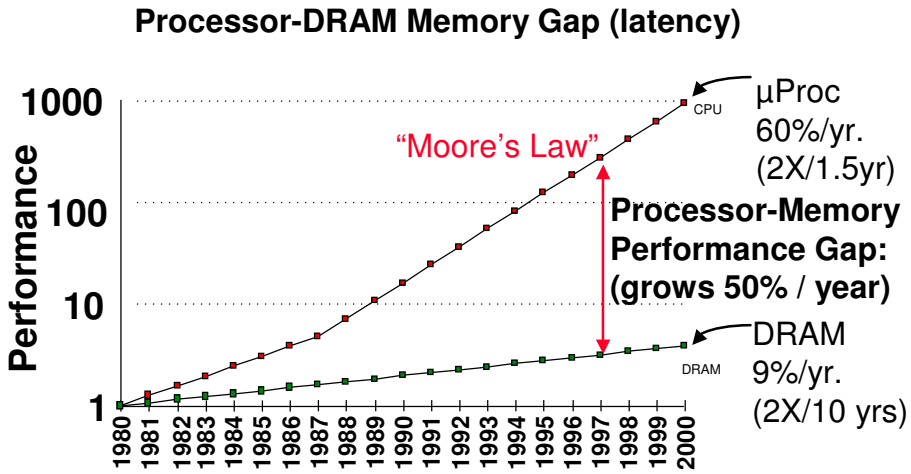
Memory Hierarchy

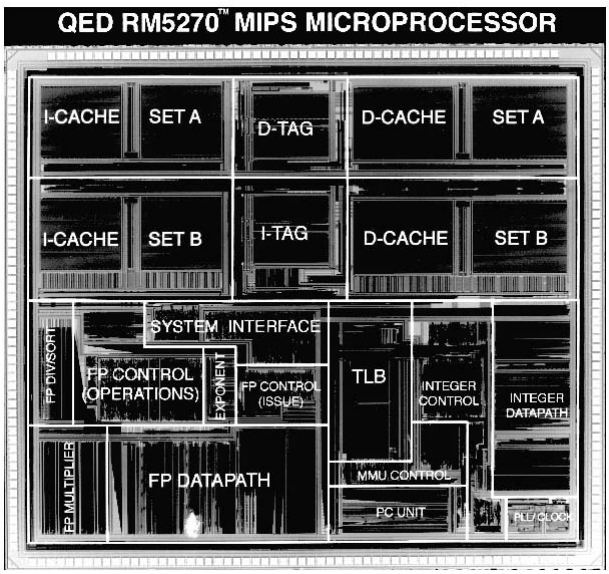
Maurizio Palesi

References

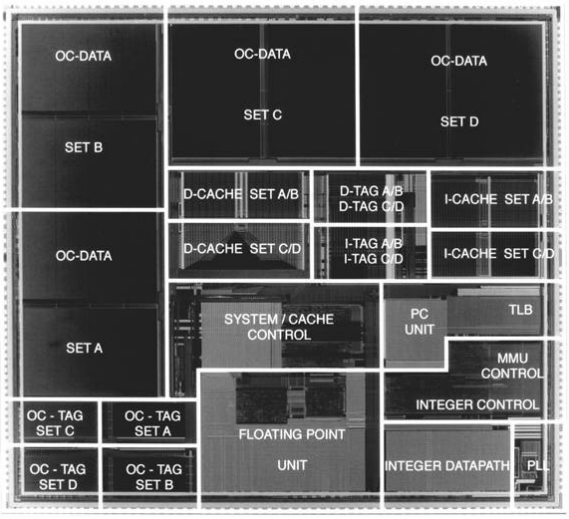
- John L. Hennessy and David A. Patterson, *Computer Architecture a Quantitative Approach*, second edition, Morgan Kaufmann
  - ➔Chapter 5

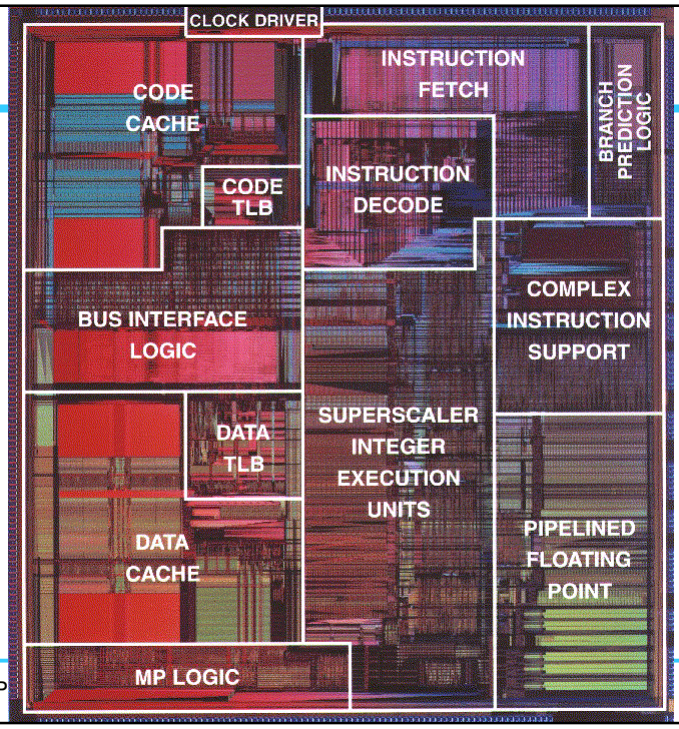
Who Cares About the Memory Hierarchy?



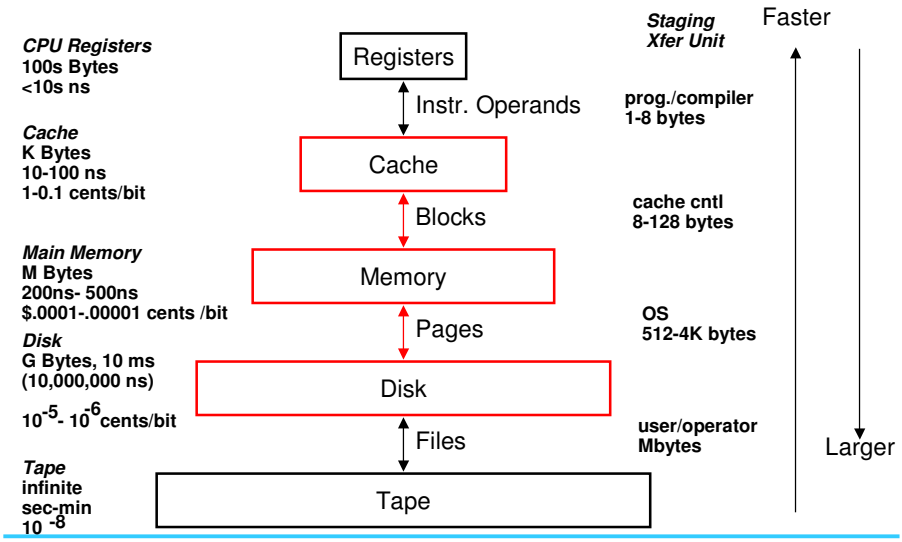


QED RM7000™ MIPS MICROPROCESSOR





Levels of the Memory Hierarchy



What is a Cache?

- Small, fast storage used to improve average access time to slow memory
- Exploits spacial and temporal locality
- In computer architecture, almost everything is a cache!
  - Registers a cache on variables
  - First-level cache a cache on second-level cache
  - Second-level cache a cache on memory
  - Memory a cache on disk (virtual memory)
  - TLB a cache on page table
  - Branch-prediction a cache on prediction information?

The Principle of Locality

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time
- Two Different Types of Locality:
  - **Temporal Locality** (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - **Spatial Locality** (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straightline code, array access)
- Last 15 years, HW relied on locality for speed

Exploit Locality

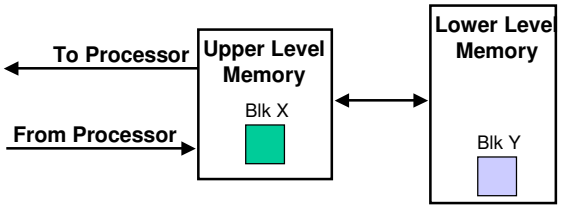
- By taking advantage of the principle of locality
  - Present the user with as much memory as is available in the cheapest technology
  - Provide access at the speed offered by the fastest technology
- DRAM is slow but cheap and dense
  - Good choice for presenting the user with a BIG memory system
- SRAM is fast but expensive and not very dense
  - Good choice for providing the user FAST access time

General Principes

- Locality
  - *Temporal Locality*: referenced again soon
  - *Spatial Locality*: nearby items referenced soon
- Locality + smaller HW is faster = memory hierarchy
  - *Levels*: each smaller, faster, more expensive/byte than level below
  - *Inclusive*: data found in top also found in the bottom
- Definitions
  - *Upper* is closer to processor
  - *Block*: minimum unit that present or not in upper level
  - Address = *Block frame address* + *block offset address*

Memory Hierarchy: Terminology

- **Hit**: data appears in some block in the upper level (example: Block X)
  - **Hit Rate**: the fraction of memory access found in the upper level
  - **Hit Time**: Time to access the upper level which consists of
    RAM access time + Time to determine hit/miss
- **Miss**: data needs to be retrieve from a block in the lower level (Block Y)
  - **Miss Rate** = 1 - (Hit Rate)
  - **Miss Penalty**: Time to replace a block in the upper level +
    Time to deliver the block the processor
- Hit Time << Miss Penalty (500 instructions on 21264!)

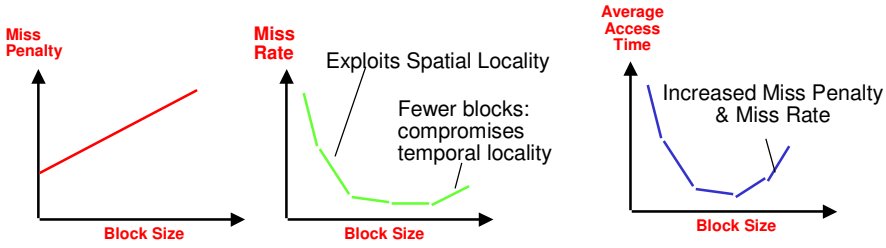


Cache Measures

- ***Average memory-access time***
  = Hit time + Miss rate x Miss penalty [ns or clocks]
- ***Miss penalty***: time to replace a block from lower level, including time to replace in CPU
  - ➔ ***access time***: time to lower level
    = f(latency to lower level)
  - ➔ ***transfer time***: time to transfer block
    =f(BW between upper & lower levels)

Block Size Tradeoff

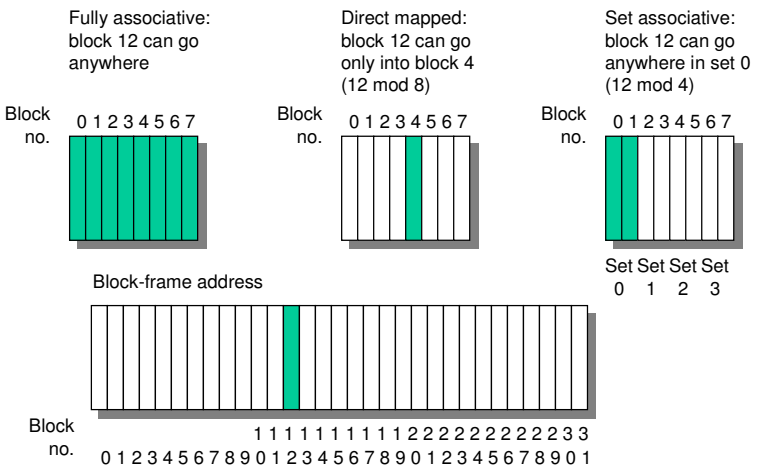
- In general, larger block size take advantage of spatial locality BUT
  - ➔ Larger block size means larger miss penalty
    - ✓ Takes longer time to fill up the block
  - ➔ If block size is too big relative to cache size, miss rate will go up
    - ✓ Too few cache blocks
- In general, Average Access Time
  = Hit Time + Miss Penalty x Miss Rate



4 Questions for Memory Hierarchy

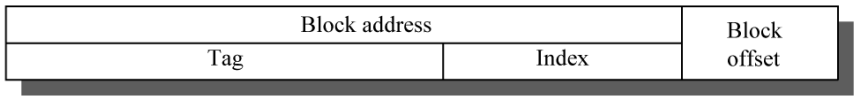
- Q1: Where can a block be placed in the upper level? ***(Block placement)***
  - ➔Fully Associative, Set Associative, Direct Mapped
- Q2: How is a block found if it is in the upper level? ***(Block identification)***
  - ➔Tag/Block
- Q3: Which block should be replaced on a miss? ***(Block replacement)***
  - ➔Random, LRU
- Q4: What happens on a write? ***(Write strategy)***
  - ➔Write Back or Write Through (with Write Buffer)

Q1: Where can a block be placed in the upper level?



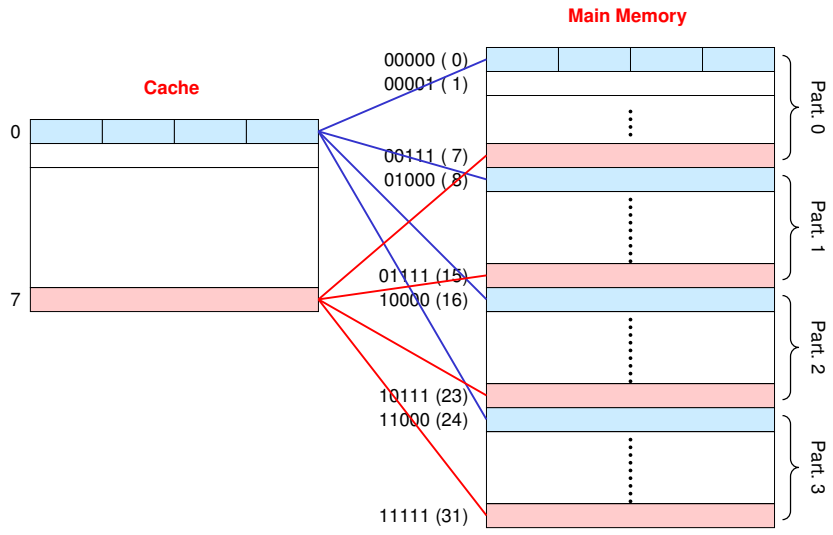
Q2: How Is a Block Found If It Is in the Upper Level?

- Tag on each block
  - ➔No need to check index or block offset
- Increasing associativity shrinks index, expands tag

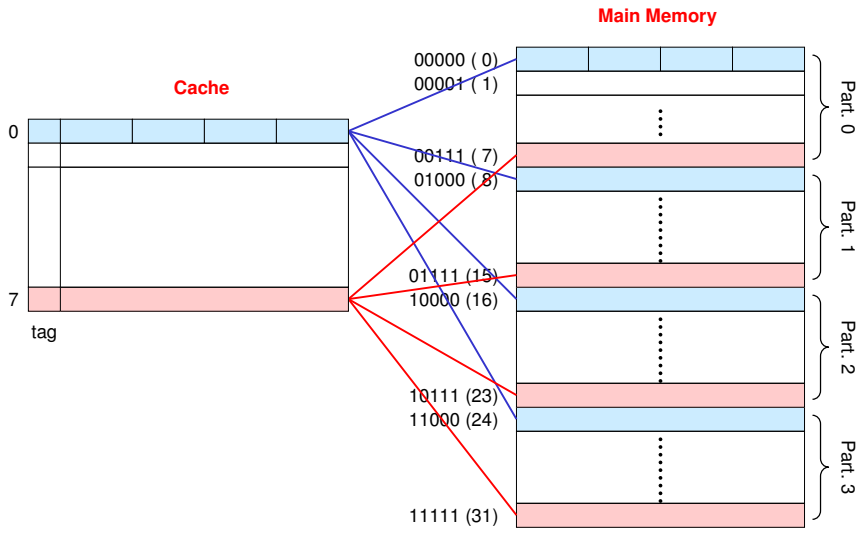


Full Associative: No index
Direct Mapped : Large index

Cache Direct Mapped



Cache Direct Mapped



Q3: Which Block Should be Replaced on a Miss?

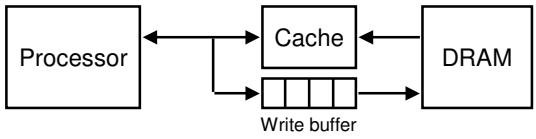
- Easy for Direct Mapped
- S.A. or F.A.:
  - ➔Random (large associativities)
  - ➔LRU (smaller associativities)

	Associativity					
	2-way		4-way		8-way	
Size	LRU	RND	LRU	RND	LRU	RND
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

Q4: What Happens on a Write?

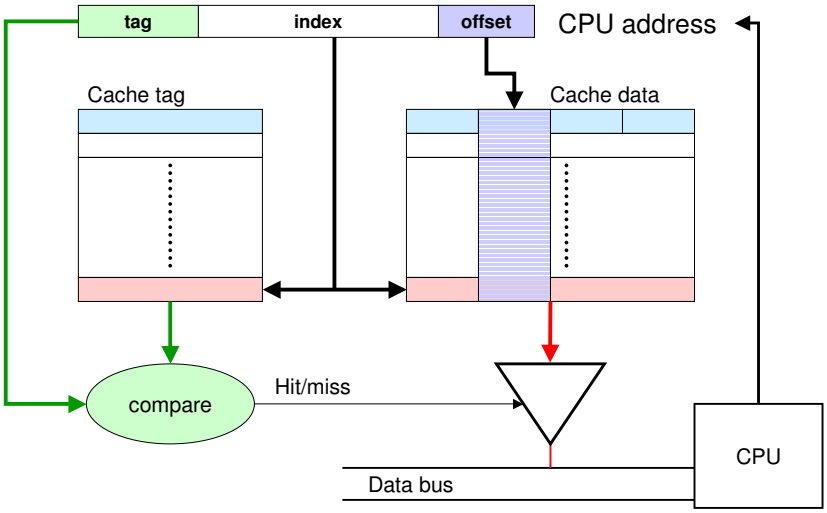
- Write through: The information is written to both the block in the cache and to the block in the lower-level memory
- Write back: The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced
  - ➔ Is block clean or dirty?
- Pros and Cons of each
  - ➔ WT: read misses cannot result in writes (because of replacements)
  - ➔ WB: no writes of repeated writes
- WT always combined with write buffers so that don't wait for lower level memory

Write Buffer for Write Through



- A Write Buffer is needed between the Cache and Memory
  - ➔ Processor: writes data into the cache and the write buffer
  - ➔ Memory controller: write contents of the buffer to memory
- Write buffer is just a FIFO
  - ➔ Typical number of entries: 4
  - ➔ Works fine if: Store frequency (w.r.t. time) << 1 / DRAM write cycle
- Memory system designer's nightmare
  - ➔ Store frequency (w.r.t. time) > 1 / DRAM write cycle
  - ➔ Write buffer saturation

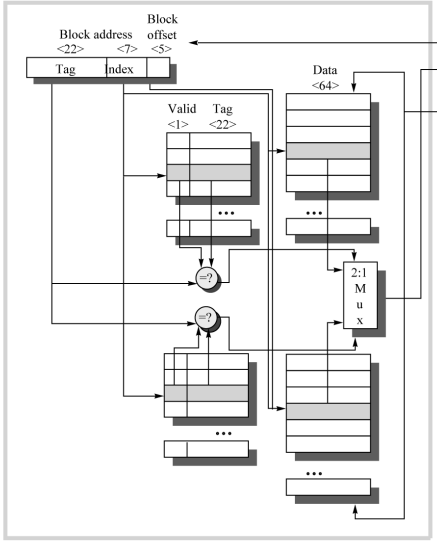
How a Block is Found in Cache



How a Block is Found in Cache

Two sets of Address tags and data RAM

Use address bits to select correct DRAM



2:1 Mux for the way