07 Memory

07.03 Memory hierarchy

- Reference locality
- Caching
- Virtual memory

Conflicting requirements

- Requirements:
  - Size: Computer systems require an ever increasing amount of main memory
  - Speed: CPUs are designed assuming that memory accesses take 1 CPU clock cycle
  - Cost: The memory system should have a marginal impact on the cost of a computer system
- Issues:
  - For a given technology, the access time of a memory device grows with its size
  - SRAMs are much faster than DRAMs, but they are more expansive and less dense
- An ideal memory should be large and cheap as a DRAM and fast as a small SRAM

Memory hierarchy

- Locality of memory references:
  - Spatial locality: the likelihood of referencing a memory address is higher if an address near it was just referenced.
  - Temporal locality: a memory address that is referenced at one point in time is likely to be referenced again sometime in the near future.
- Access time of memory devices:
  - Most DRAM devices can work in page mode to provide a burst of data read from the same row at a speed much higher than the typical access time of a random access

Cache

- A cache is a small and fast memory that duplicates some of the entries of main memory
- Accesses to the cache are much faster than accesses to the main memory
- As long as the processor finds in cache the memory entries it needs, the perceived performance is the performance of the cache
- The principle of locality is used to decide what to place in cache

Direct-mapped cache

RAM

	00	01	10	11
0000				
0001				
0010				
0011				
0100				A
0101				
0110				
0111				
1000				
1001				
1010				
1011				
1100				
1101				
1110			B	
1111				

block address					
tag		index		offset	
0	1	0	0	1	1

cache

	00	01	10	11	TAG
00				A	01
01					
10			B		11
11					

Fully associative cache

RAM

	00	01	10	11
0000				
0001				
0010				
0011				
0100				A
0101				
0110				
0111				
1000				
1001				
1010				
1011				
1100				
1101				
1110			B	
1111				

block address					
tag				offset	
0	1	0	0	1	1

cache way1

00	01	10	11	TAG
			A	0100

cache way2

00	01	10	11	TAG
		B		1110

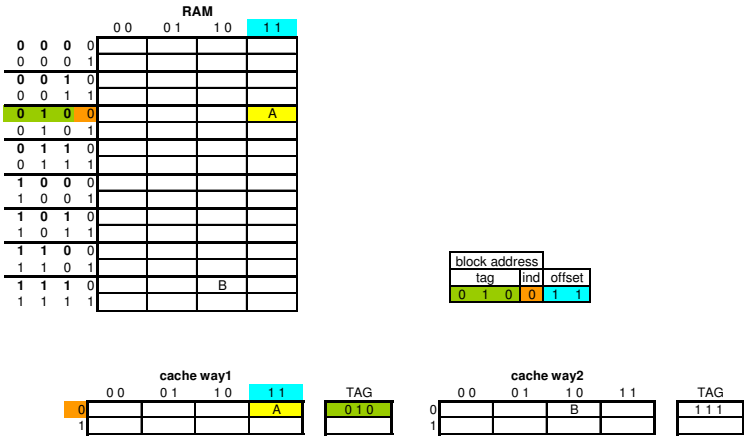
cache way3

00	01	10	11	TAG

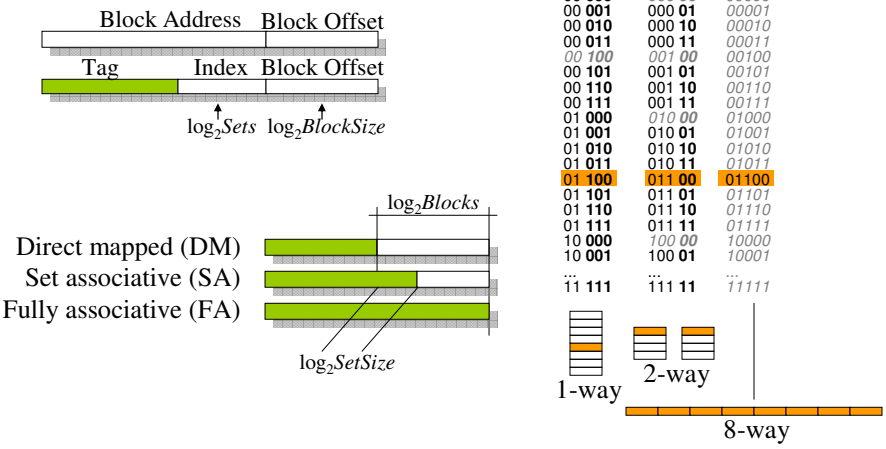
cache way4

00	01	10	11

Set-associative cache



Cache: block identification



Cache: block replacement

- Which block should be replaced on a cache miss?
  1. *Random (RND)*
  2. *Least-Recently Used (LRU)*
  3. *First In First Out (FIFO)*

  RND and FIFO policies are easier to implement, while LRU is more coherent with locality

  Caches provide a *used* flag associated with each block to support an approximation of LRU:
  - the flag is periodically reset
  - candidates for replacement on a miss are blocks with flag 0 (i.e., unused since last reset)

Cache performance

$$AvgAccessTime = HitTime + MissRate \cdot MissPenalty$$

$$AvgAccessTime = HitRate \cdot HitTime + MissRate \cdot MissTime$$

$$MissPenalty = MissTime - HitTime$$

Level-1 and Level-2 cache:

$$AvgAccessTime = HitTime + MissRate(HitTime2 + MissRate2 \cdot MissPenalty2)$$