Cache II

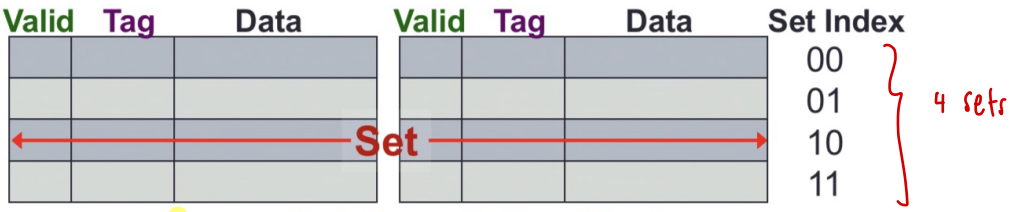
Block Size Trade-off

- Average Access Time = Hit rate × Hit Time + (1 − Hit rate) × Miss penalty.
- Larger block size:
  - \+ takes advantage of spatial locality
  - − larger miss penalty: takes longer time to fill up the block
  - − If block size is too big relative to cache size
    → Too few cache blocks → miss rate will go up.

Set Associative (SA) Cache

- Conflict misses (several blocks mapped to same block/set)
  - Solution via SA cache.
- N-way Set Associative Cache.
  - N: number of blocks in one set.
- Key Idea:
  - Cache consists of a number of sets:
    - Each set contains N cache blocks.
  - Each memory block maps to a unique cache set.
  - Within the set, a memory block can be placed in any of the N cache blocks in the set.

Structure of SA Cache



2-way Set Associative Cache

- each set has 2 cache blocks
- A memory block maps to a unique set.
  - In the set, the memory block can be placed in either of the cache blocks.
    - need to search both to look for the memory block.

Mapping:

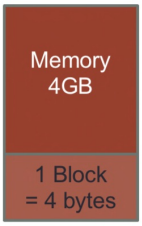
Cache Set Index
= (BlockNumber) modulo (NumberOfCacheSets)

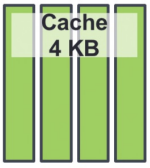
Cache Block size = 2^N bytes
Number of cache sets = 2^M
Offset = **N bits**
Set Index = **M bits**
Tag = **32 – (N + M) bits**

Observation:
It is essentially unchanged from the direct-mapping formula

"Set" index instead of index.

Example:





Memory Address

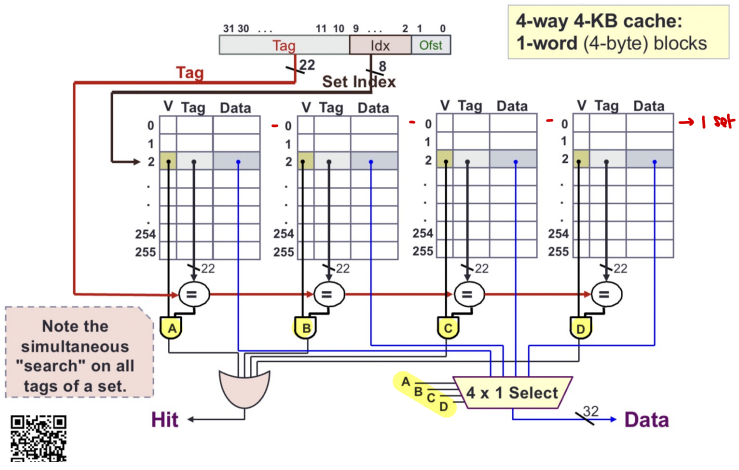
Offset, N = 2 bits
Block Number = 32 – 2 = **30 bits**
Check: Number of Blocks = 2^{30}

Number of Cache Blocks
= 4KB / 4bytes = 1024 = 2^{10}

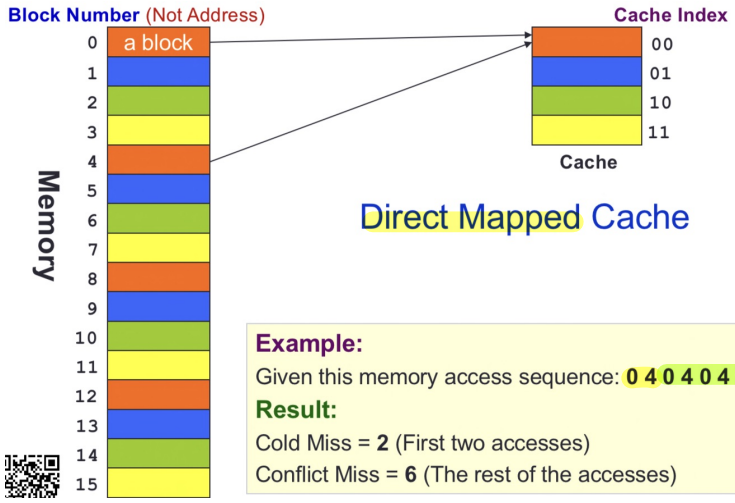
4-way associative, number of sets
= 1024 / 4 = 256 = 2^8
Set Index, M = 8 bits

Cache Tag = 32 – 8 – 2 = **22 bits**

every set has 4 blocks.



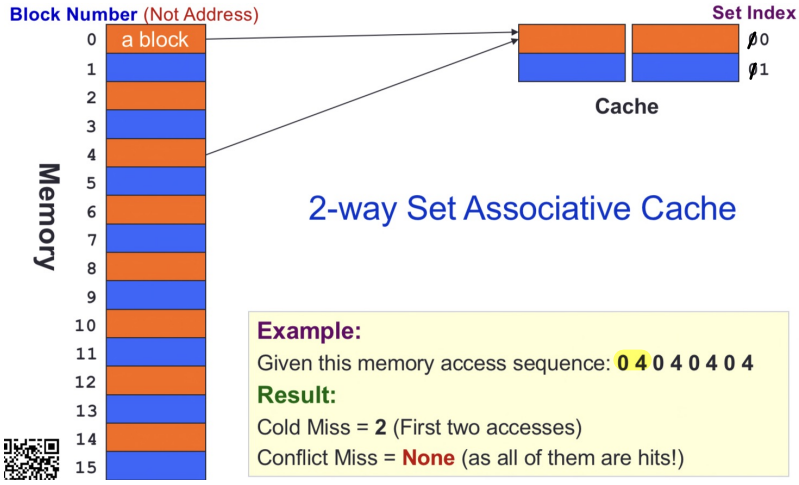
Advantage of Associativity



Direct Mapped Cache

Example:
Given this memory access sequence: **0 4 0 4 0 4 0 4**
Result:
Cold Miss = **2** (First two accesses)
Conflict Miss = **6** (The rest of the accesses)

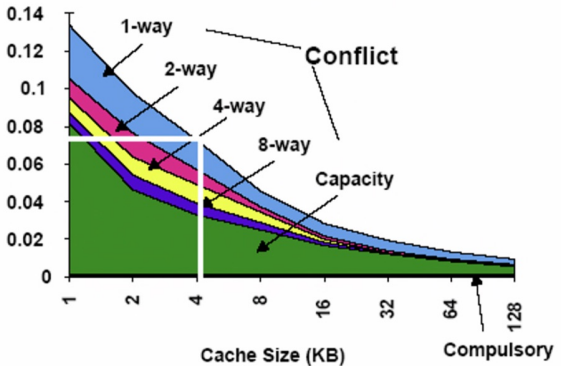
worst performance, no hit.



2-way Set Associative Cache

Example:
Given this memory access sequence: **0 4 0 4 0 4 0 4**
Result:
Cold Miss = **2** (First two accesses)
Conflict Miss = **None** (as all of them are hits!)

Rule of Thumb:
A direct-mapped cache of size **N** has about the same miss rate as a 2-way set associative cache of size **N/2**



Given:

- Memory access sequence: **4, 0, 8, 36, 0**
- 2-way set-associative cache with a total of four 8-byte blocks ➔ **total of 2 sets**
- Indicate hit/miss for each access

Offset, N = 3 bits
Block Number = 32 – 3 = 29 bits

2-way associative, number of sets= 2 **= 2^1**
Set Index, M = 1 bits

Cache Tag = 32 – 3 – 1 = **28 bits**

3. SA Cache Example: **Load #1**

Miss
4, 0 , 8, 36, 0

- Load from **4** ➔ Tag: 00000000000000000000000000 | Index: 0 | Offset: 100

Check: Both blocks in **Set 0** are invalid **[Cold Miss]**

Result: Load from memory and place in **Set 0 - Block 0**

Set Index	Block 0				Block 1			
	Valid	Tag	W0	W1	Valid	Tag	W0	W1
0	1	0	M[0]	M[4]	0			
1	0				0			

3. SA Cache Example: **Load #2**

Miss Hit
4, 0, 8, 36, 0

- Load from **0** ➔ Tag: 00000000000000000000000000 | Index: 0 | Offset: 000

Result:
[Valid and Tags match] in **Set 0-Block 0 [Spatial Locality]**

Set Index	Block 0				Block 1			
	Valid	Tag	W0	W1	Valid	Tag	W0	W1
0	1	0	M[0]	M[4]	0			

3. SA Cache Example: **Load #3**

Miss Hit Miss
4, 0 , 8, 36, 0

- Load from **8** ➔ Tag: 00000000000000000000000000 | Index: 1 | Offset: 000

Check: Both blocks in **Set 1** are invalid **[Cold Miss]**

Result: Load from memory and place in **Set 1 - Block 0**

Set Index	Block 0				Block 1			
	Valid	Tag	W0	W1	Valid	Tag	W0	W1
0	1	0	M[0]	M[4]	0			
1	1	0	M[8]	M[12]	0			

3. SA Cache Example: **Load #4**

Miss Hit Miss Miss
4, 0 , 8, 36, 0

- Load from **36** ➔ Tag: 00000000000000000000000010 | Index: 0 | Offset: 100

Check: [Valid but tag mismatch] **Set 0 - Block 0**
[Invalid] **Set 0 - Block1 [Cold Miss]**
Result: Load from memory and place in **Set 0 - Block 1**

Set Index	Block 0				Block 1			
	Valid	Tag	W0	W1	Valid	Tag	W0	W1
0	1	0	M[0]	M[4]	1	2	M[32]	M[36]
1	1	0	M[8]	M[12]	0			

3. SA Cache Example: **Load #5**

Miss Hit Miss Miss Hit
4, 0 , 8, 36, 0

- Load from **0** ➔ Tag: 00000000000000000000000000 | Index: 0 | Offset: 000

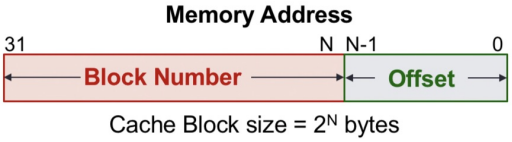
Check: [Valid and tags match] **Set 0-Block 0**
[Valid but tags mismatch] **Set 0-Block1**
[Temporal Locality]

Set Idx	Block 0				Block 1			
	Valid	Tag	W0	W1	Valid	Tag	W0	W1
0	1	0	M[0]	M[4]	1	2	M[32]	M[36]
1	1	0	M[8]	M[12]	0			

Fully Associative Cache

- A memory block can be placed in any location in the cache
- Key Idea:
  - Memory block placement is no longer restricted by cache index or cache set index.
  - Can be placed in any location
  - But need to search all cache blocks for memory access.
  - Incurs capacity misses.

Mapping:

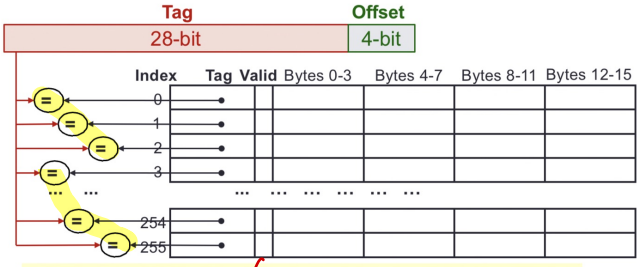


Cache Block size = 2^N bytes

Cache Block size = 2^N bytes
Number of cache blocks = 2^M
Offset = **N bits**
Tag = **32 – N bits**

Observation:
The block number serves as the tag in FA cache.

- Example: 2^{12} 2^4
  - 4KB cache size and 16-Byte block size → 2^8 = 256 blocks.
  - Compare tags and valid bit in parallel



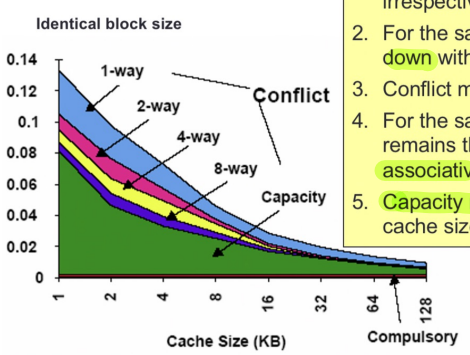
No Conflict Miss (since data can go anywhere)

have to compare every Valid bit (no index)

a lot of comparators!

⇒ Useful for small cache.

4. Cache Performance



Observations:
1. Cold/compulsory miss remains the same irrespective of cache size/associativity.
2. For the same cache size, conflict miss goes down with increasing associativity.
3. Conflict miss is 0 for FA caches.
4. For the same cache size, capacity miss remains the same irrespective of associativity.
5. Capacity miss decreases with increasing cache size . (FA)

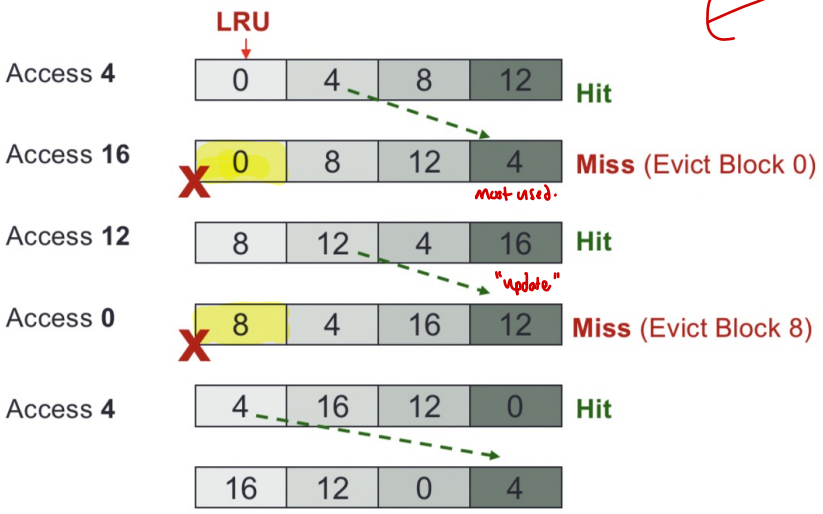
Total Miss = Cold miss + Conflict miss + Capacity miss
Capacity miss (FA) = Total miss (FA) – Cold miss (FA), when Conflict Miss→0

5. Block Replacement Policy (1/3)

- Set Associative or Fully Associative Cache: (not Direct Mapped Cache)
  - Can choose where to place a memory block
  - Potentially replacing another cache block if full
  - Need **block replacement policy**
- **Least Recently Used (LRU)**
  - **How:** For cache hit, record the cache block that was accessed
    - When replacing a block, choose one which has not been accessed for the longest time
  - **Why:** Temporal locality – least used less likely to be used again.

Least Recently Used policy in action:

- 4-way SA cache
- Memory accesses: **0 4 8 12 4 16 12 0 4**



not cache, data structure that supports LRU.

- Drawback for LRU
  - Hard to keep track if there are many choices
- Other replacement policies:
  - First in first out (FIFO)
  - Random replacement (RR)
  - Least frequently used (LFU)

Example (for all 3 types):

6. Additional Examples #1

- **Direct-Mapped** Cache:
  - Four 8-byte blocks
- Memory accesses:
  4, 8, 36, 48, 68, 0, 32

Addr:	Tag	Index	Offset
4:	00…000	00	100
8:	00…000	01	000
36:	00…001	00	100
48:	00…001	10	000
68:	00…010	00	100
0:	00…000	00	000
32:	00…001	00	000

Index	Valid	Tag	Word0	Word1
0	~~0~~ 1	~~0~~ 1 ~~1~~ 2 ~~0~~	~~M[0]~~ M[32] ~~M[32]~~ M[64] ~~M[0]~~	~~M[4]~~ M[36] ~~M[36]~~ M[68] ~~M[4]~~
1	~~0~~ 1	0	M[8]	M[12]
2	~~0~~ 1	1	M[48]	M[52]
3	0			

6. Additional Examples #2

- **Fully-Associative** Cache:
  - Four 8-byte blocks
  - LRU Replacement Policy
- Memory accesses:
  4, 8, 36, 48, 68, 0, 32 (Hit)

Addr:	Tag	Offset
4:	00…00000	100
8:	00…00001	000
36:	00…00100	100
48:	00…00110	000
68:	00…01000	100
0:	00…00000	000
32:	00…00100	000

Index	Valid	Tag	Word0	Word1
0	~~0~~ 1	~~0~~ 8	~~M[0]~~ M[64]	~~M[4]~~ M[68]
1	~~0~~ 1	~~1~~ 0	~~M[8]~~ M[0]	~~M[12]~~ M[4]
2	~~0~~ 1	4	M[32]	M[36]
3	~~0~~ 1	6	M[48]	M[52]

replace at current cache for next iteration.

6. Additional Examples #3

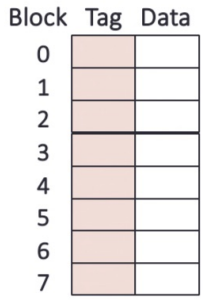
- **2-way Set-Associative** Cache:
  - Four 8-byte blocks
  - LRU Replacement Policy
- Memory accesses:
  4, 8, 36, 48, 68, 0, 32

Addr:	Tag	Index	Offset
4:	00…0000	0	100
8:	00…0000	1	000
36:	00…0010	0	100
48:	00…0011	0	000
68:	00…0100	0	100
0:	00…0000	0	000
32:	00…0010	0	000

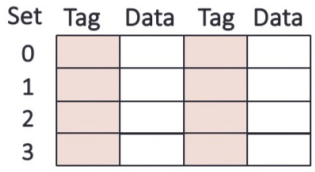
Set Index	Block 0				Block 1			
	Valid	Tag	Word0	Word1	Valid	Tag	Word0	Word1
0	~~0~~ 1	~~0~~ 3 0	~~M[0]~~ M[48] M[0]	~~M[4]~~ M[52] M[4]	~~0~~ 1	~~2~~ 4 2	~~M[32]~~ M[64] M[32]	~~M[36]~~ M[68] M[36]
1	~~0~~ 1	0	M[8]	M[12]	0			

7. Summary: Cache Organizations

One-way set associative (direct mapped)

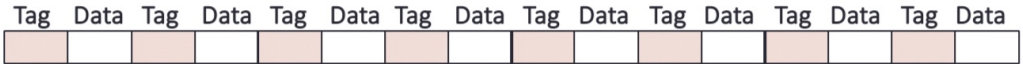


Two-way set associative



Four-way set associative

Eight-way set associative (fully associative)



offset = what words in the block.

Block Placement: Where can a block be placed in cache?

Direct Mapped:	N-way Set-Associative:	Fully Associative:
• Only one block defined by index	• Any one of the **N** blocks within the set defined by index	• Any cache block

Block Identification: How is a block found if it is in the cache?

Direct Mapped:	N-way Set Associative:	Fully Associative:
• Tag match with only one block	• Tag match for all the blocks within the set	• Tag match for all the blocks within the cache

Block Replacement: Which block should be replaced on a cache miss?

Direct Mapped:	N-way Set-Associative:	Fully Associative:
• No Choice	• Based on replacement policy	• Based on replacement policy

Write Strategy: What happens on a write?

Write Policy: Write-through vs write-back

Write Miss Policy: Write allocate vs write no allocate