Computer Architecture

Discussion 10

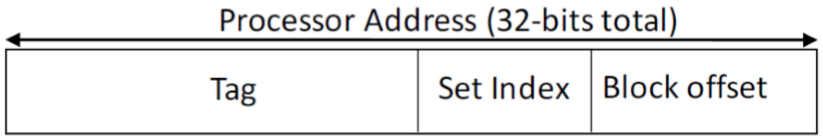
CB

Relationships between 3 mappings

Direct Mapped

Set Associative

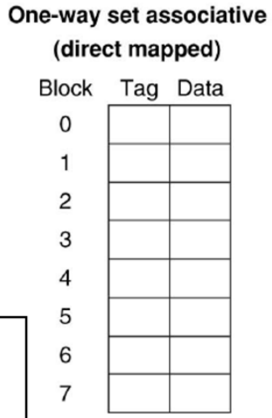
Fully Associative: remove set index

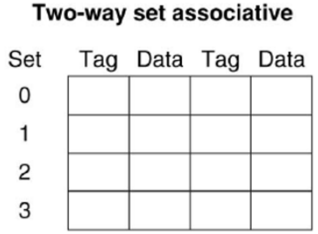


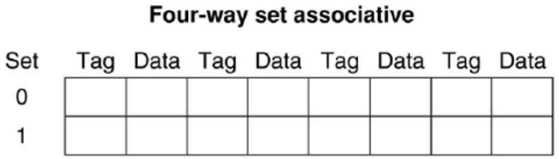
Same format of address:
If each set maps to N numbers, then:
Direct Mapped: $a+\log(N)+c$
Set Associative: $a+n_w+(\log(N)-n_w)+c$
Fully Associative: remove set index

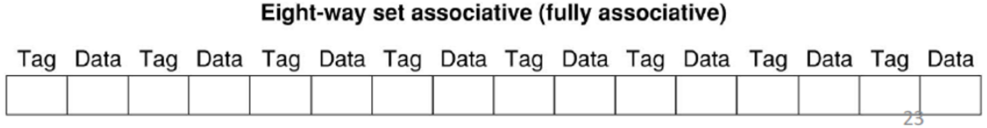
Different Organizations of an Eight-Block Cache

Total size of \$ in blocks is equal to *number of sets* × *associativity*. For fixed \$ size and fixed block size, increasing associativity decreases number of sets while increasing number of elements per set. With eight blocks, an 8-way set-associative \$ is same as a fully associative \$.





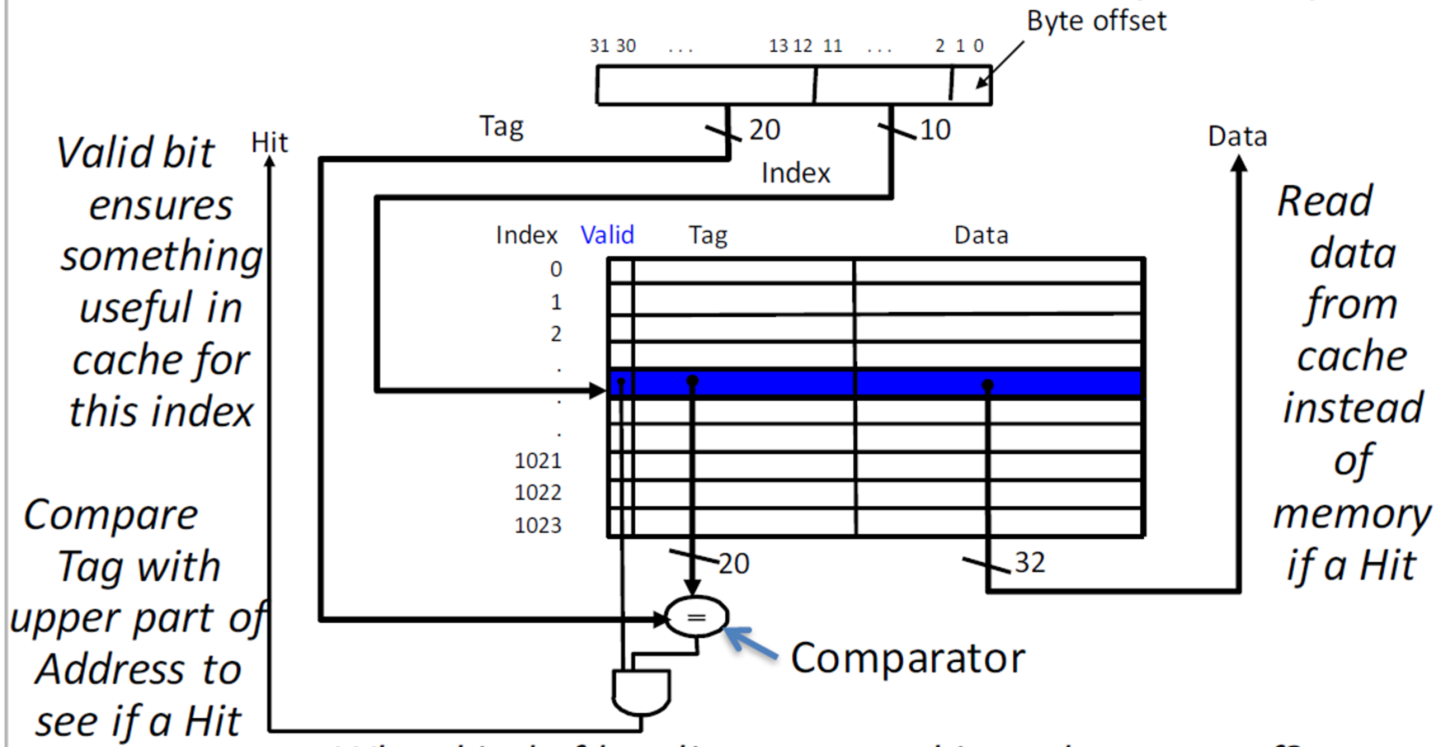




Direct Mapped Cache

- Only one comparator is enough – each memory block is mapped to only 1 index in cache
- Number of index bits determined by cache size and block size
- Index_num = cache_size / 2^(byte_offset) (in Byte)

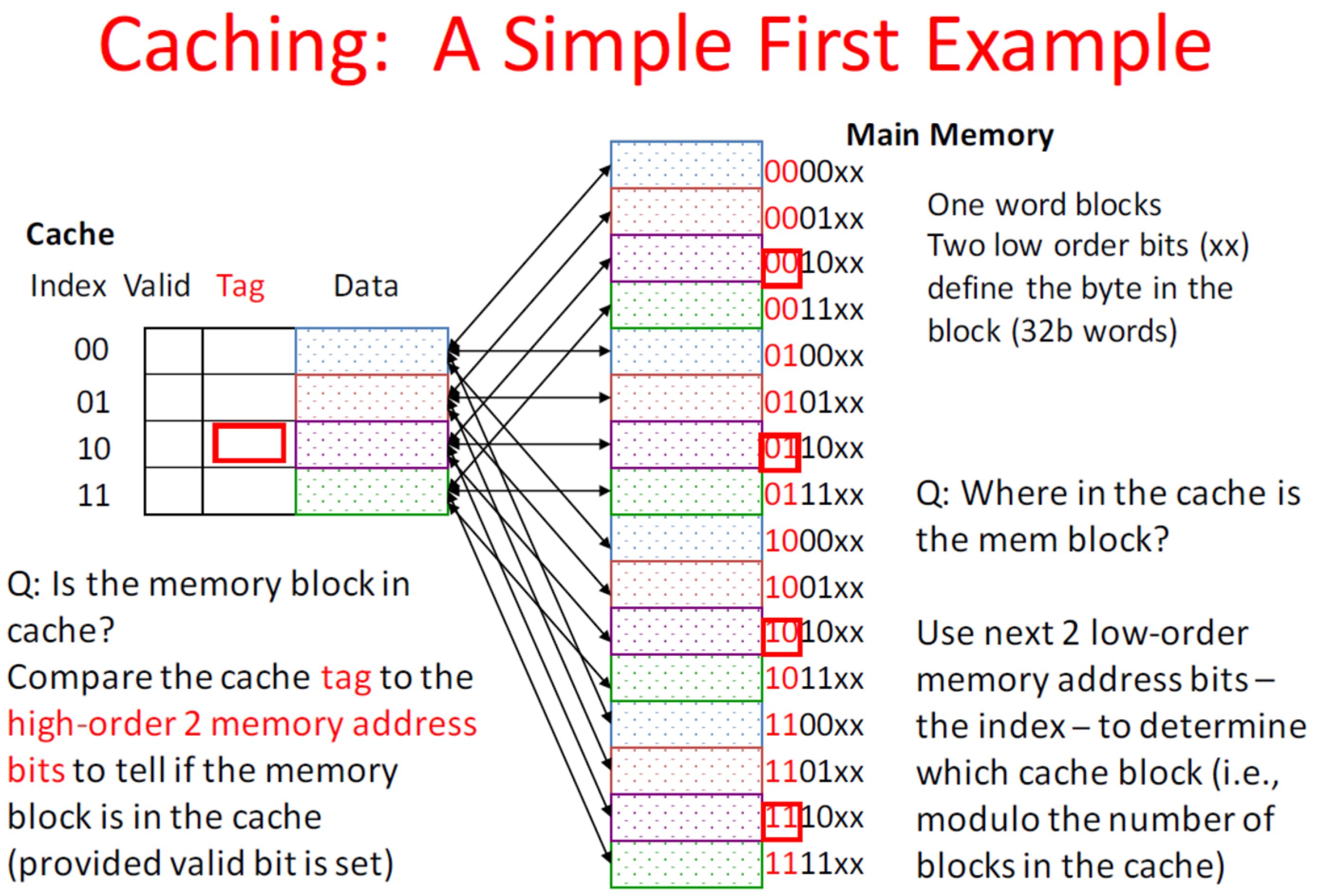
- One word blocks, cache size = 1K words (or 4KB)



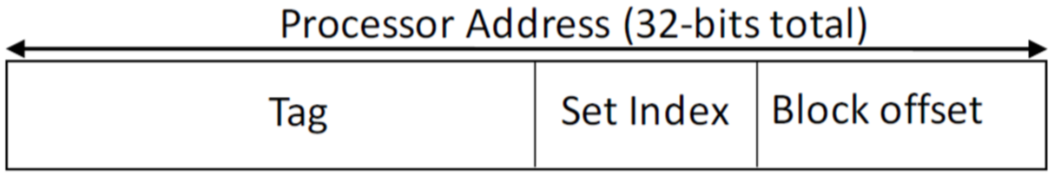
What kind of locality are we taking advantage of?

Direct Mapped Cache

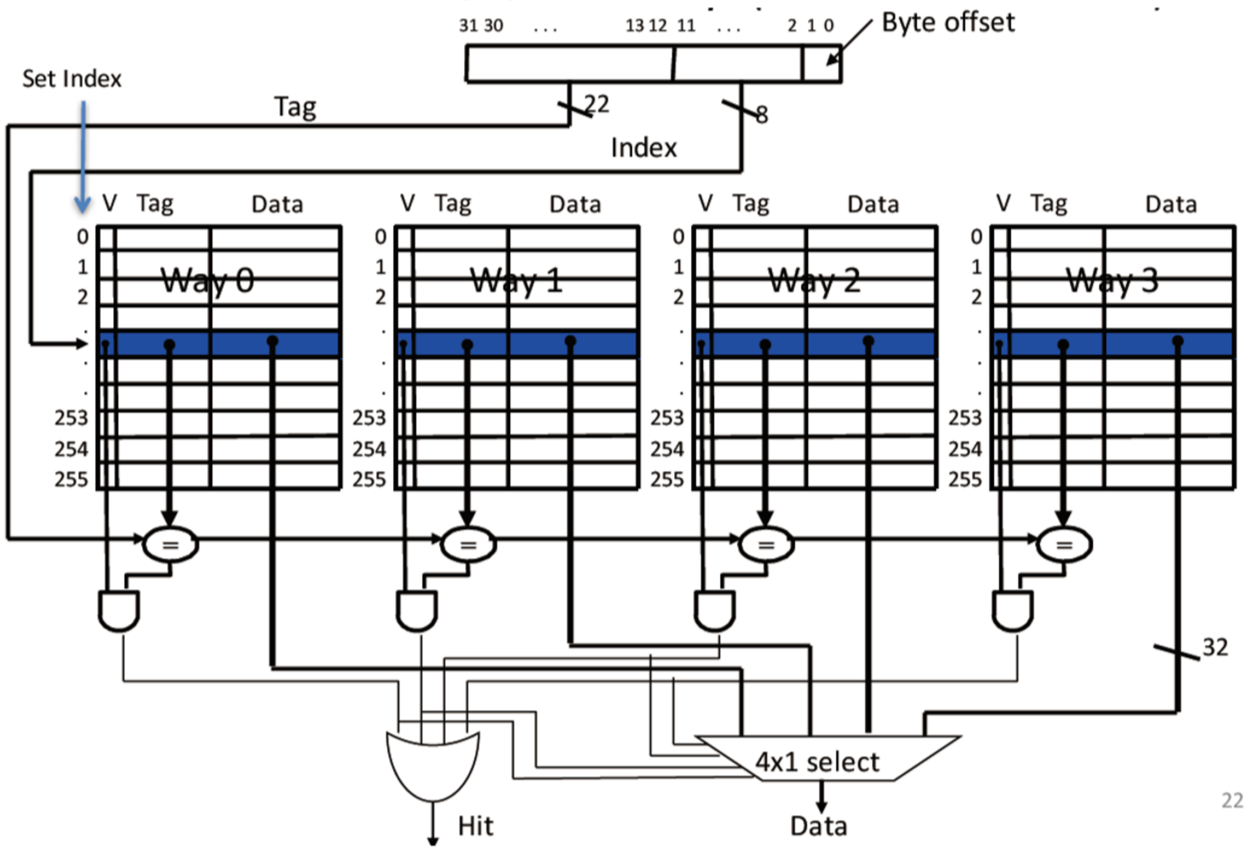
- A 16B cache
- Memory blocks with the save index could be stored in the same data address of a cache
- Compare Tag(the next 2 low-order bits) to judge if the memory block ins in cache
- If in, add byte offset



Set-Associative Caches

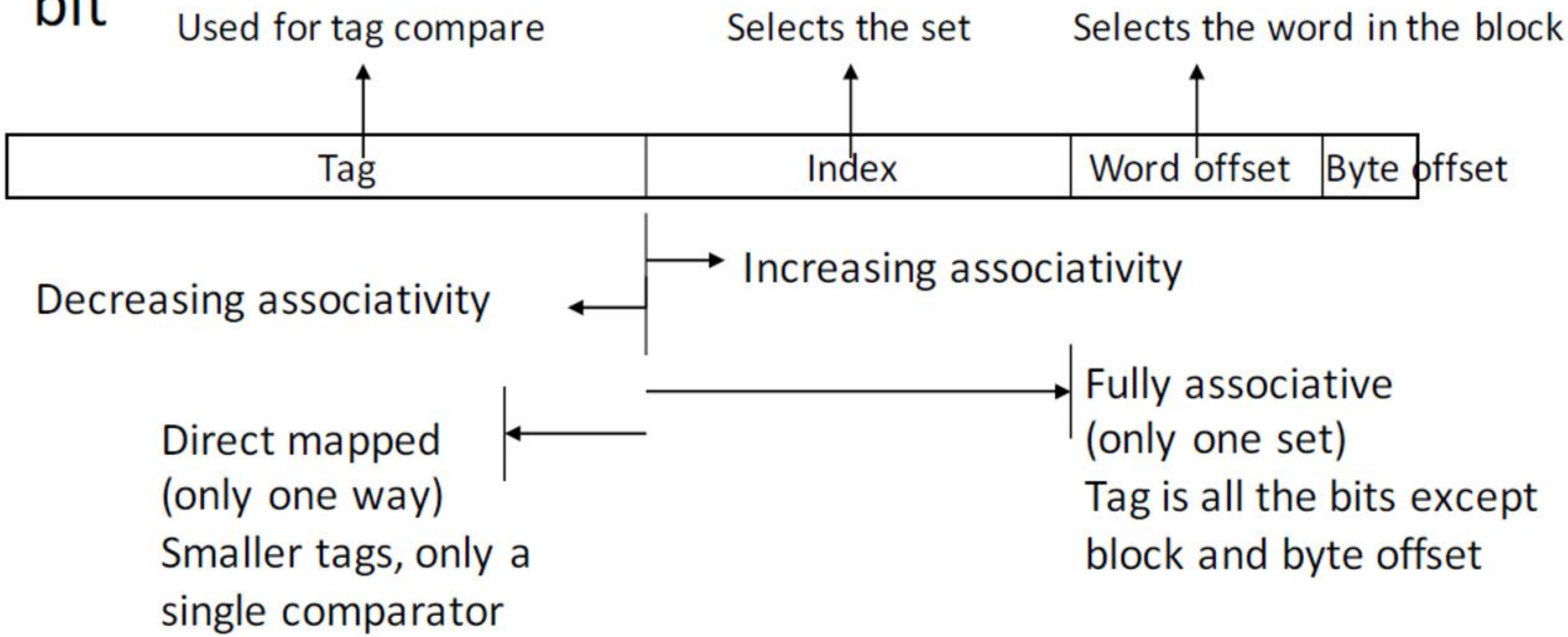


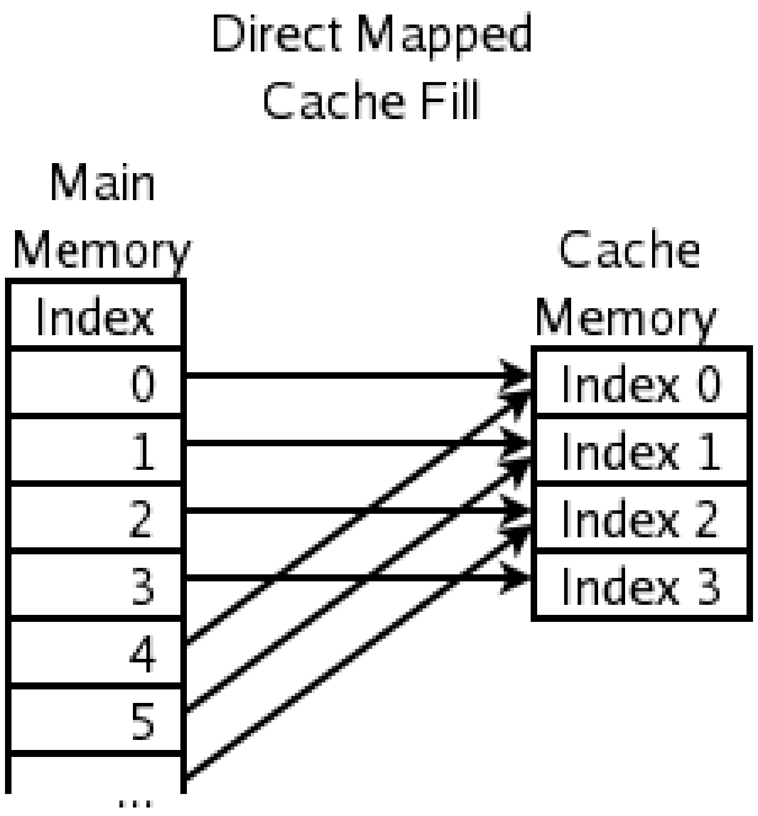
- A mixture of Fully Associative and Direct Mapped
  - FA: looks up every tag
  - DM: compare with only 1 tag
  - SA: looks up N ways
- Tag_width + index_width + offset_width = const
- If one is changed, we can change another to maintain the cache size.



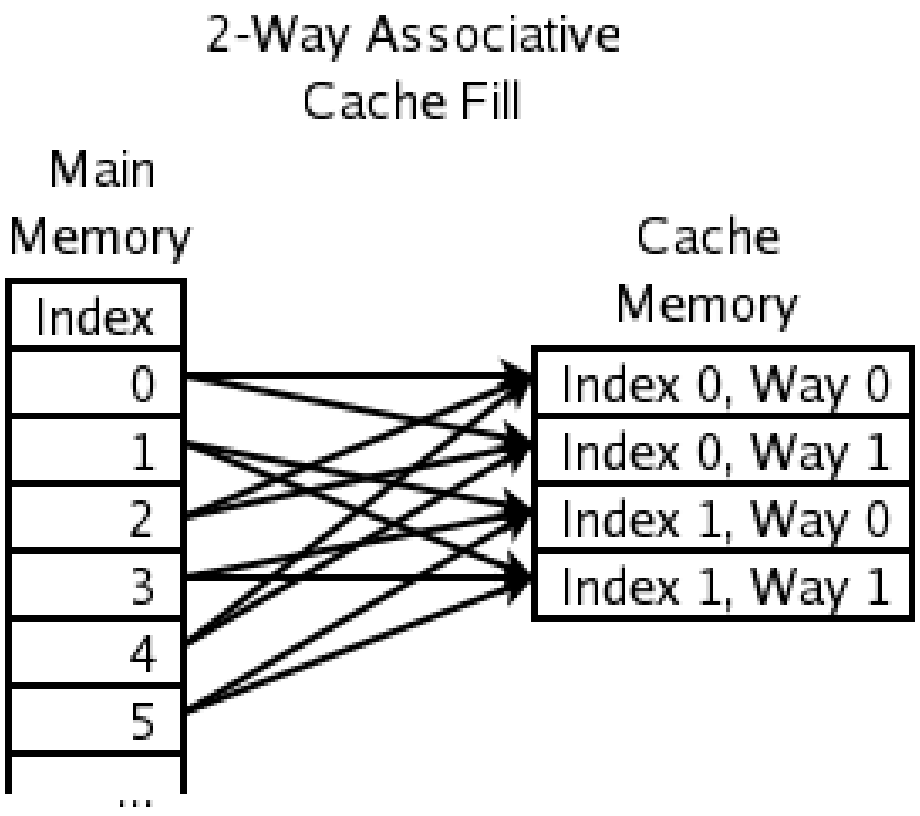
Range of Set-Associative Caches

- For a *fixed-size* cache and fixed block size, each increase by a factor of two in associativity doubles the number of blocks per set (i.e., the number or ways) and halves the number of sets – decreases the size of the index by 1 bit and increases the size of the tag by 1 bit





Each location in main memory can be cached by just one cache location.



Each location in main memory can be cached by one of two cache locations.

Set-Associative Caches

Processor Address (32-bits total)

Tag	Set Index	Block offset

- For a cache with constant total capacity, if we increase the number of ways by a factor of 2, which statement is false:
- A: The number of sets could be doubled
- B: The tag width could decrease
- C: The block size could stay the same
- D: The block size could be halved
- E: Tag width must increase

$2^i 2^b 2^w$ =const→ $i + b + w$ =const
Tag width must increase by 1.

- 1 more index bit
- A: true if we divide block size by 4
- B: False.
- C: byte offset not changed
- D: b_width-1
- E: Correct

Average Memory Access Time (AMAT)

- Average Memory Access Time (AMAT) is the average time to access memory considering both hits and misses in the cache

AMAT = Time for a hit

+ Miss rate × Miss penalty

- Hit rate: fraction of accesses that hit in the cache
- Miss rate: 1 – Hit rate
- Miss penalty: time to replace a block from lower level in memory hierarchy to cache
- Hit time: time to access cache memory (including tag comparison)

Average Memory Access Time(AMAT)

AMAT = Time for a hit + Miss rate x Miss penalty

Given a 200 psec clock, a miss penalty of 50 clock cycles, a miss rate of 0.02 misses per instruction and a cache hit time of 1 clock cycle, what is AMAT?

- ☐ A: ≤200 psec
- ☐ B: 400 psec
- ☐ C: 600 psec
- ☐ D: ≥ 800 psec

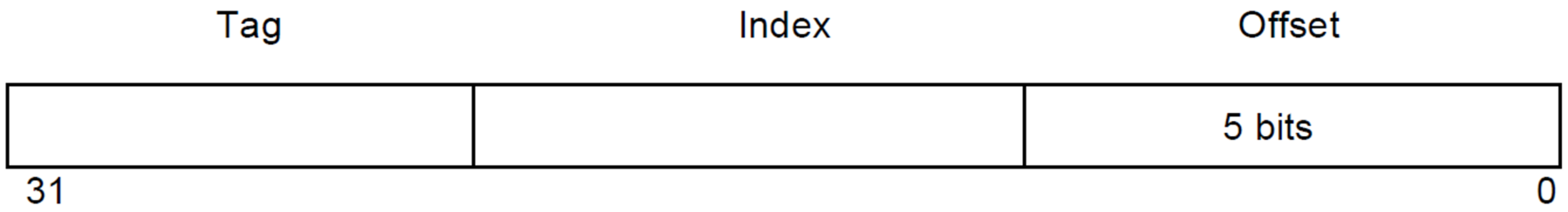
Understanding Cache Misses: The 3Cs

- Compulsory (cold start or process migration, 1^{st} reference):
  - First access to block impossible to avoid; small effect for long running programs
  - Solution: increase block size (increases miss penalty; very large blocks could increase miss rate)
- Capacity:
  - Cache cannot contain all blocks accessed by the program
  - Solution: increase cache size (may increase access time)
- *Conflict (collision):*
  - *Multiple memory locations mapped to the same cache location*
  - *Solution 1: increase cache size*
  - *Solution 2: increase associativity (may increase access time)*

Exercise

- Consider a 32-bit physical memory space and a 32 KiB 2-way associative cache with LRU replacement.

You are told the cache uses 5 bits for the offset field. Write in the number of bits in the tag and index fields in the figure below.



Exercise

Tag	Index	Offset
18 bits	**9 bits**	5 bits
31		0

- For the same cache, after the execution of the following code:

```
int ARRAY_SIZE = 64 * 1024;
int arr[ARRAY_SIZE]; // *arr is aligned to a cache block
/* loop 1 */ for (int i = 0; i < ARRAY_SIZE; i += 8) arr[i] = i;
/* loop 2 */ for (int i = ARRAY_SIZE - 8; i >= 0; i -= 8)
                 arr[i+1] = arr[i];
```

- 1. What is the hit rate of loop 1? What types of misses (of the 3 Cs), if any, occur as a result of loop 1?
- 2. What is the hit rate of loop 2? What types of misses (of the 3 Cs), if any, occur as a result of loop 2?

```
int ARRAY_SIZE = 64 * 1024;
int arr[ARRAY_SIZE]; // *arr is aligned to a cache block
/* loop 1 */ for (int i = 0; i < ARRAY_SIZE; i += 8) arr[i] = i;
/* loop 2 */ for (int i = ARRAY_SIZE - 8; i >= 0; i -= 8)
                       arr[i+1] = arr[i];
```

- 1. What is the hit rate of loop 1? What types of misses (of the 3 Cs), if any, occur as a result of loop 1? 0, Compulsory Misses
- 2. What is the hit rate of loop 2? What types of misses (of the 3 Cs), if any, occur as a result of loop 2? 9/16, Capacity Misses

Floating-Point Representation (1/2)

- Normal format: $+1.xxx...x_{two} * 2^{yyy...y}{}_{two}$
- Multiple of Word Size (32 bits)

31	30 – 23	22 – 0
S	Exponent	Significand
1 bit	8 bits	23 bits

- S represents Sign
  Exponent represents y's
  Significand represents x's

- Represent numbers as small as $2.0_{ten} \times 10^{-38}$ to as large as $2.0_{ten} \times 10^{38}$

- Summary (single precision):

31	30 … 23	22 … 0
S	Exponent	Significand
1 bit	8 bits	23 bits

- $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$

- We still haven't used Exponent = 0, Significand nonzero
- DEnormalized number: no (implied) leading 1, implicit exponent = -126.

Special Numbers Summary

- **Reserve exponents, significands:**

Exponent	Significand	Object
0	0	0
0	nonzero	Denorm
1-254	anything	+/- fl. pt. #
255	0	+/- ∞
255	nonzero	NaN

THX >.<