DISCUSSION 10: Cache

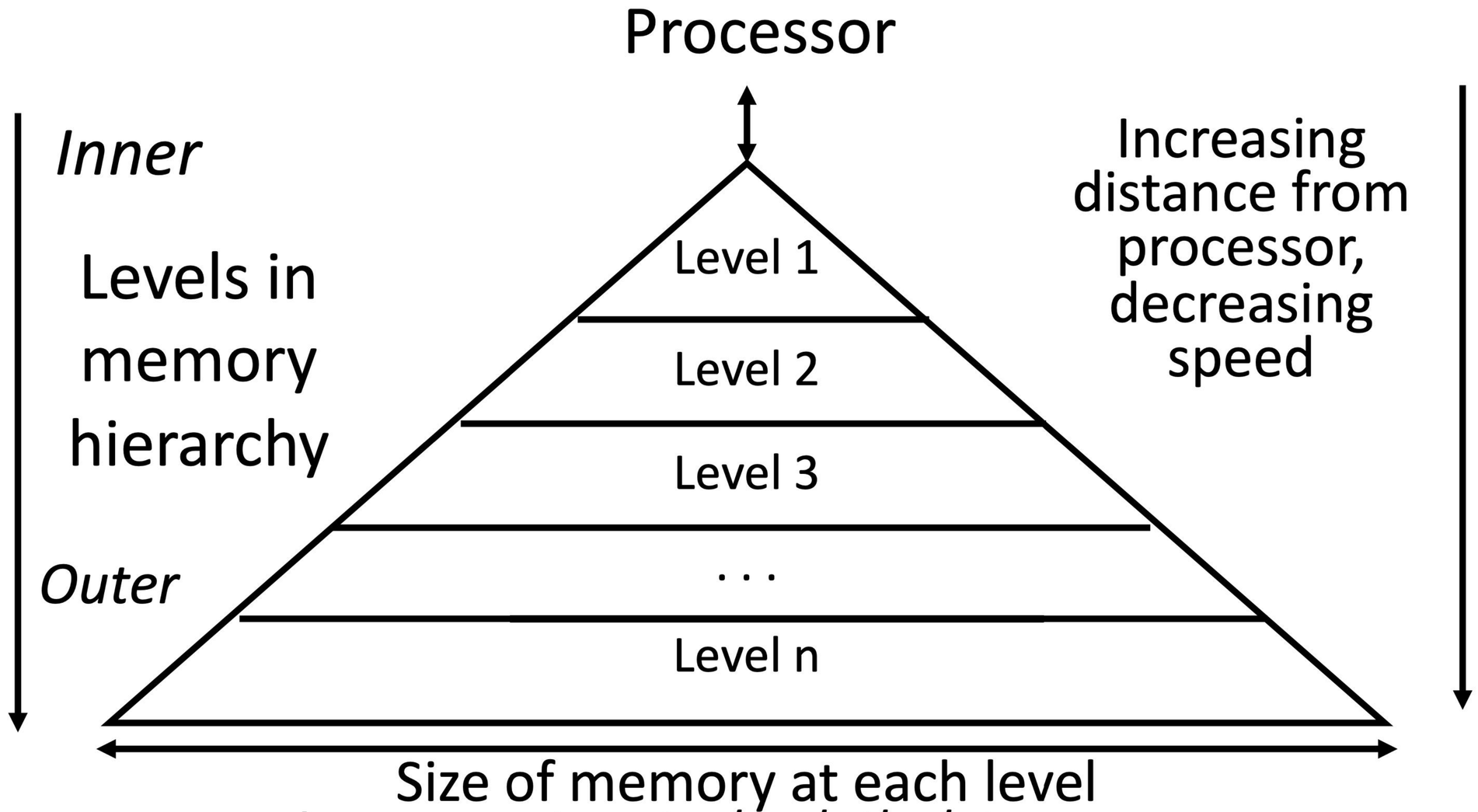
Yibo Zhao

Memory Hierarchy

- **Why memory hierarchy?**
  - Huge and increasing processor-DRAM gap.
  - Hundreds of clock cycles per memory access.
  - Slow DRAM access has a disastrous impact on CPU performance!
- ***Solution:***
  - Memory hierarchy!

Where Cached	Latency (cycles)
CPU registers	0
On-Chip TLB	0
On-Chip L1	1
Off-Chip L2	10
Main memory	100
Main memory	100
Local disk	10,000,000
Local disk	10,000,000
Remote server disks	1,000,000,000

Memory Hierarchy



As we move to outer levels the latency goes up and price per bit goes down.

Memory Reference Pattern

- **Locality**
  - **Temporal locality:** Recently referenced items are likely to be referenced in the near future.
  - **Spatial locality:** Items with nearby addresses tend to be referenced close together in time.
- **Locality Example**

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

  - Reference array elements in succession reference pattern): Spatial locality
  - Reference sum each iteration: Temporal locality

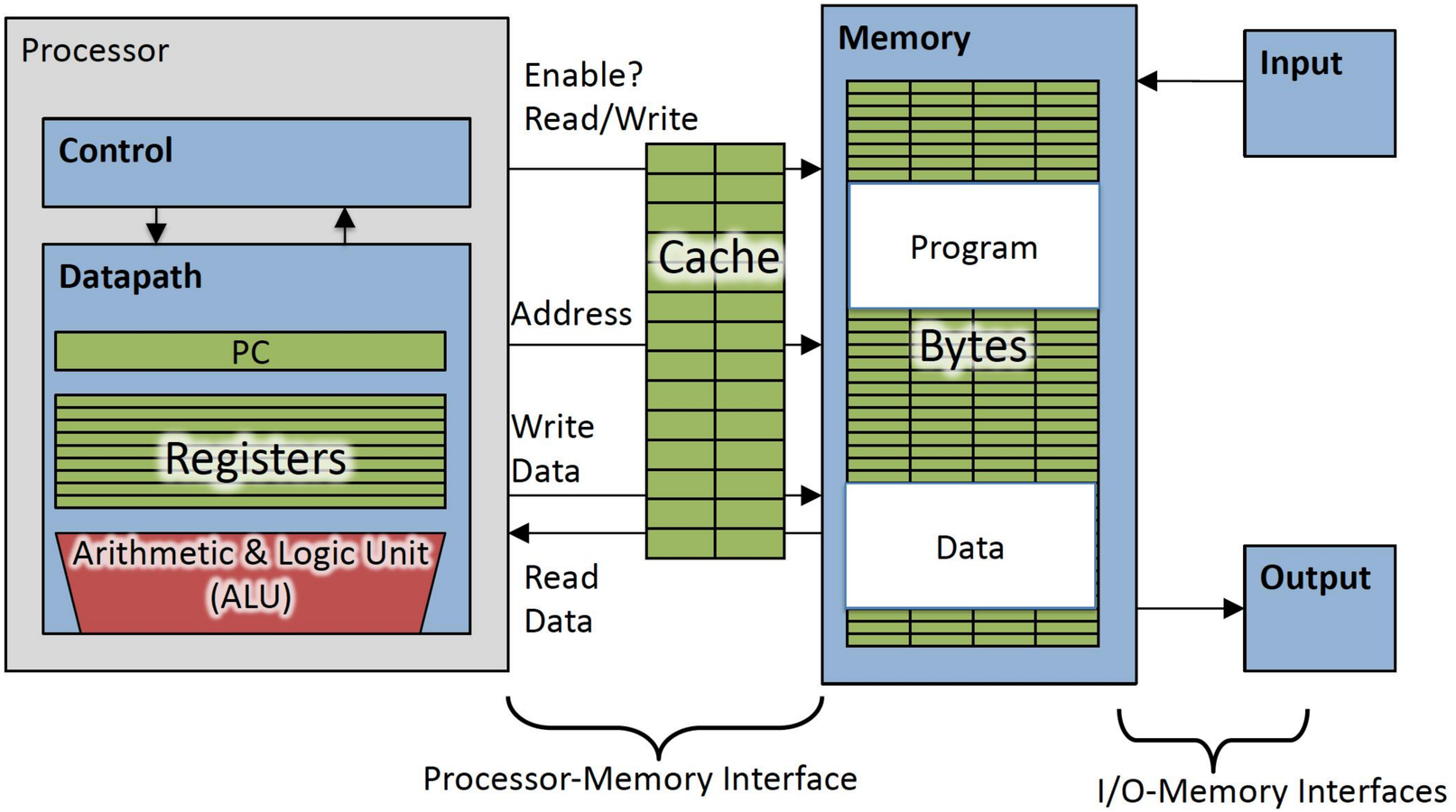
Locality Example

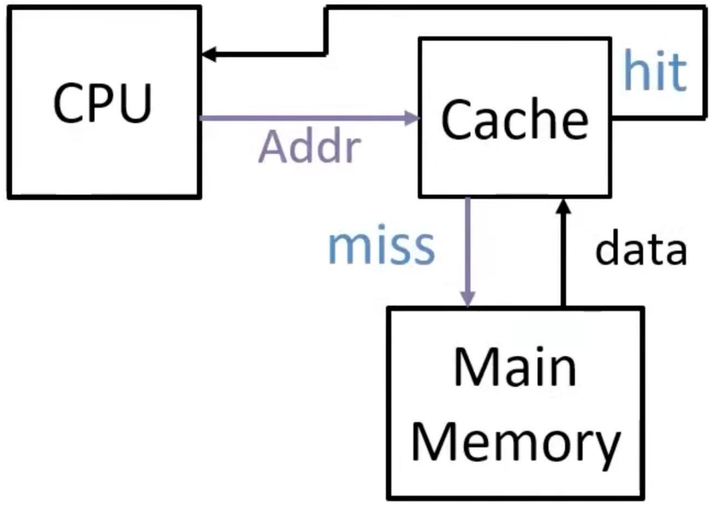
- Which program has a good locality?
  - Assume $M = N = 2048$ (anyway, a large number)

```
int sumarrayrows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

```
int sumarraycols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

Adding Cache to Computer





Address --> T/I/O

TAG: Used to distinguish different blocks that use the same index.
#bits=address bits-Index Bits-Offset Bits

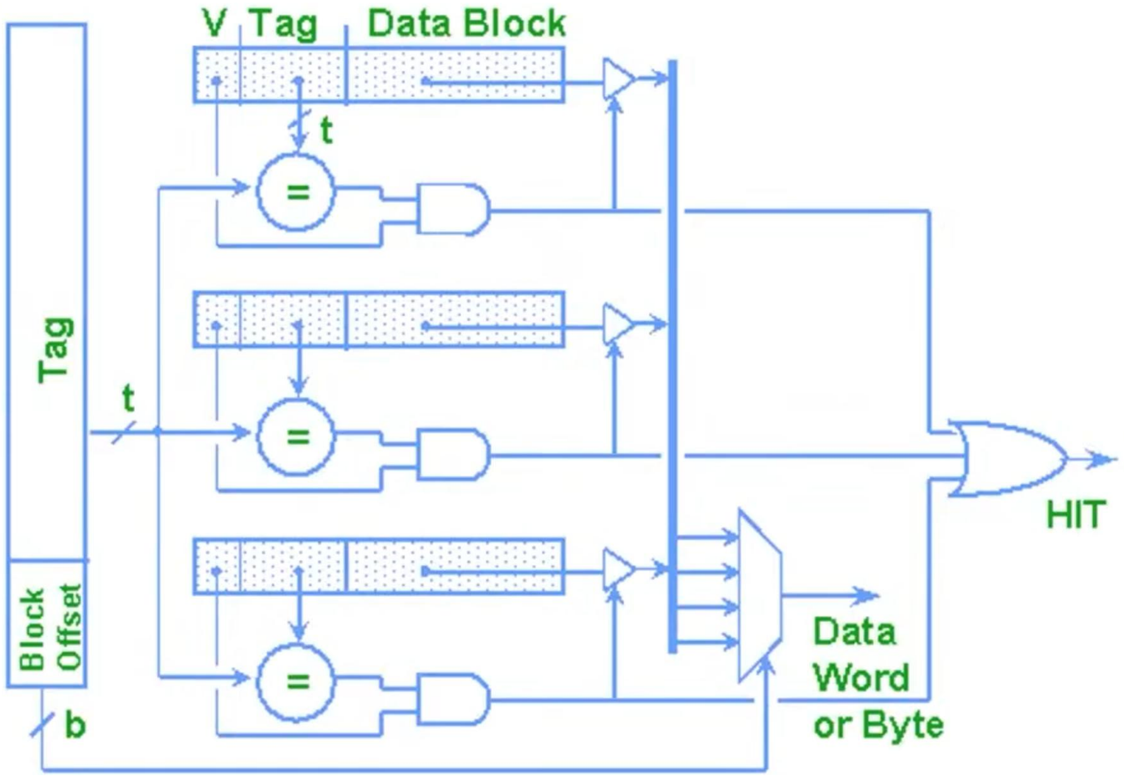
Index: The set that this piece of memory will be placed in.
#bits = $\log_2(\#\text{ of indices})$

Offset: The location of the byte in the block.
#bits = $\log_2(\text{size of block})$

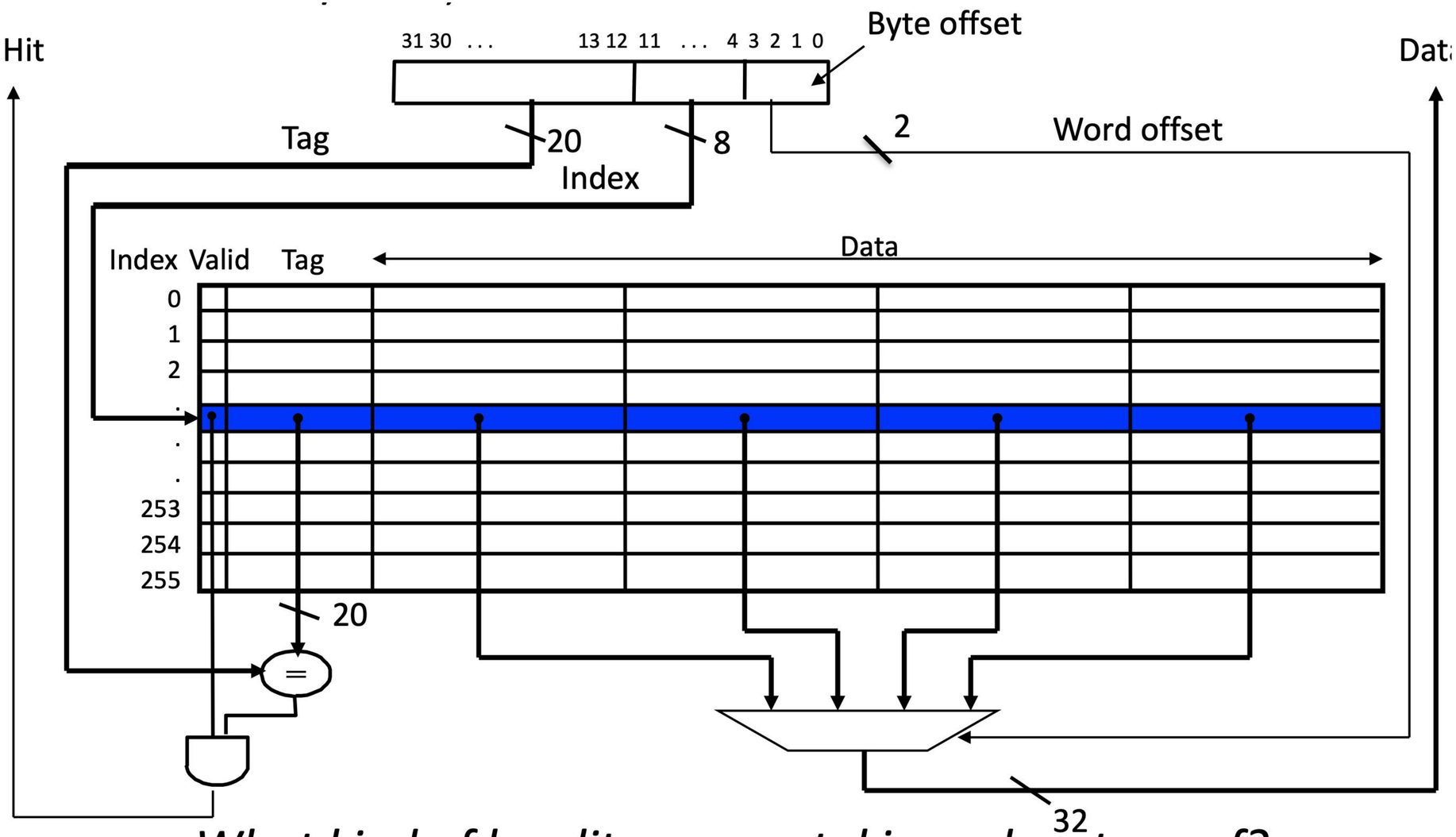
Cache Design

- **Fully Associative:** need 1 comparator/line (block), and have to look through all blocks.
- **Direct Mapped:** use a hash on address to limit line on one place.
  - One comparator
  - # sets = # blocks
- **N-way Set Associative:** N places for a line
  - N comparators
  - # sets = # blocks / N

Fully Associative Cache



Direct Mapped Cache

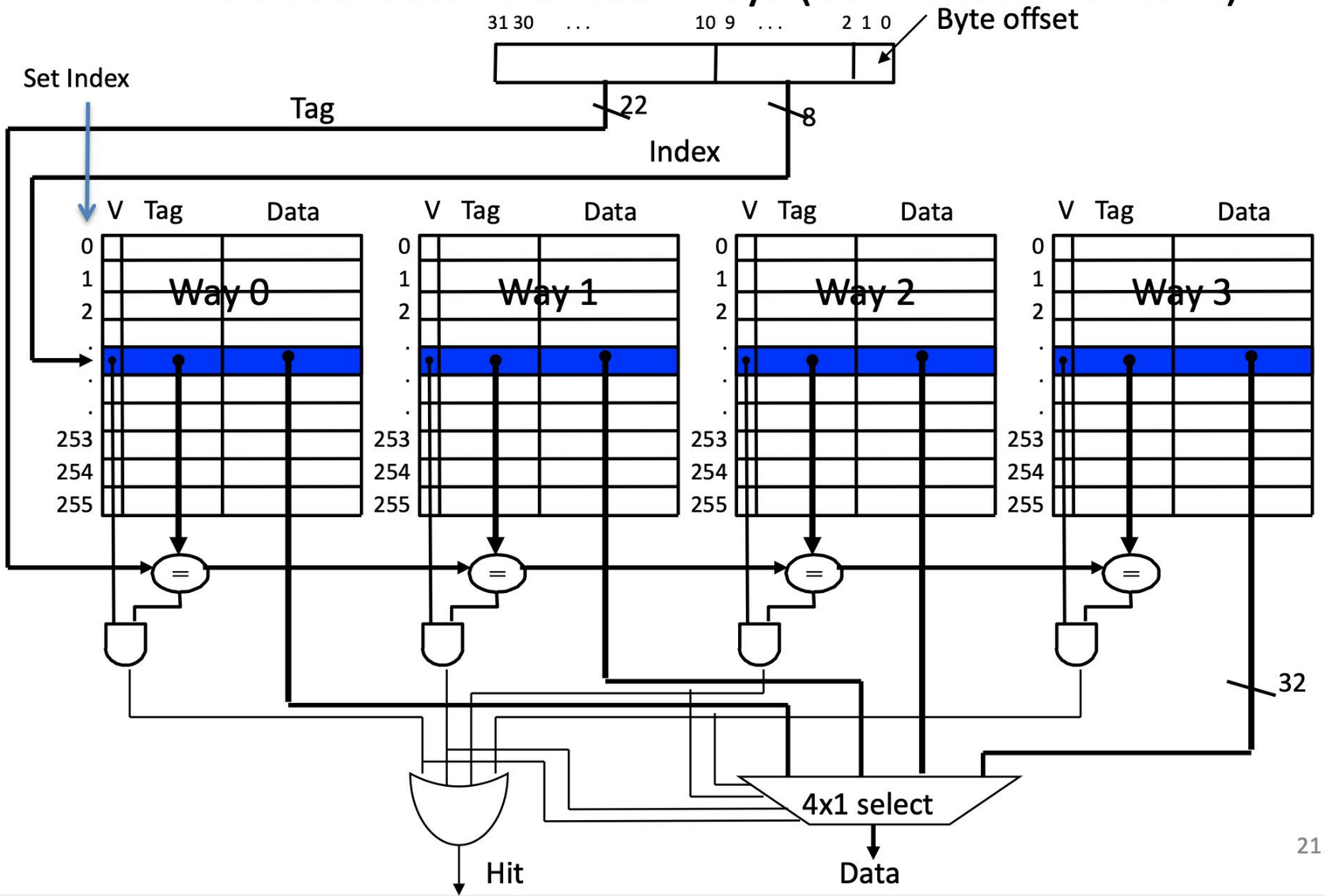


What kind of locality are we taking advantage of?

N-way set-associative cache

$$N * \#\ \text{sets} = \#\ \text{blocks}$$

- 2^8 = 256 sets each with four ways (each with one block)



3Cs

Compulsory: First time you ask the cache for a certain block. A miss that must occur when you first bring in a block. Reduce compulsory misses by having longer cache lines (bigger blocks), which bring in the surrounding addresses along with our requested data. Can also pre-fetch blocks beforehand using a hardware prefetcher (a special circuit that tries to guess the next few blocks that you will want).

Conflict: Occurs if, hypothetically, you went through the ENTIRE string of accesses with a fully associative cache (with an LRU replacement policy) and wouldn't have missed for that specific access. Increasing the associativity or improving the replacement policy would remove the miss.

Capacity: Capacity misses are independent of the associativity of your cache. If you hypothetically ran the ENTIRE string of memory accesses with a fully associative cache (with an LRU replacement policy) of the same size as your cache, and it was a miss for that specific access, then this miss is a capacity miss. The only way to remove the miss is to increase the cache capacity.

Example

Assume we have a direct-mapped byte-addressed cache with capacity 32B and block size of 8B and 32 bits in each address

Address	T/I/O	Hit, Miss, Replace
0x00000004	Tag 0, Index 0, Offset 4	M, Compulsory
0x00000005	Tag 0, Index 0, Offset 5	H
0x00000068	Tag 3, Index 1, Offset 0	M, Compulsory
0x000000C8	Tag 6, Index 1, Offset 0	R, Compulsory
0x00000068	Tag 3, Index 1, Offset 0	R, Conflict
0x000000DD	Tag 6, Index 3, Offset 5	M, Compulsory
0x00000045	Tag 2, Index 0, Offset 5	R, Compulsory
0x00000004	Tag 0, Index 0, Offset 4	R, Capacity
0x000000C8	Tag 6, Index 1, Offset 0	R, Capacity

ooo|o o|ooo

you went through the ENTIRE string of accesses with a fully associative cache (with an LRU replacement policy):
hit: **Conflict**
miss: **Capacity**