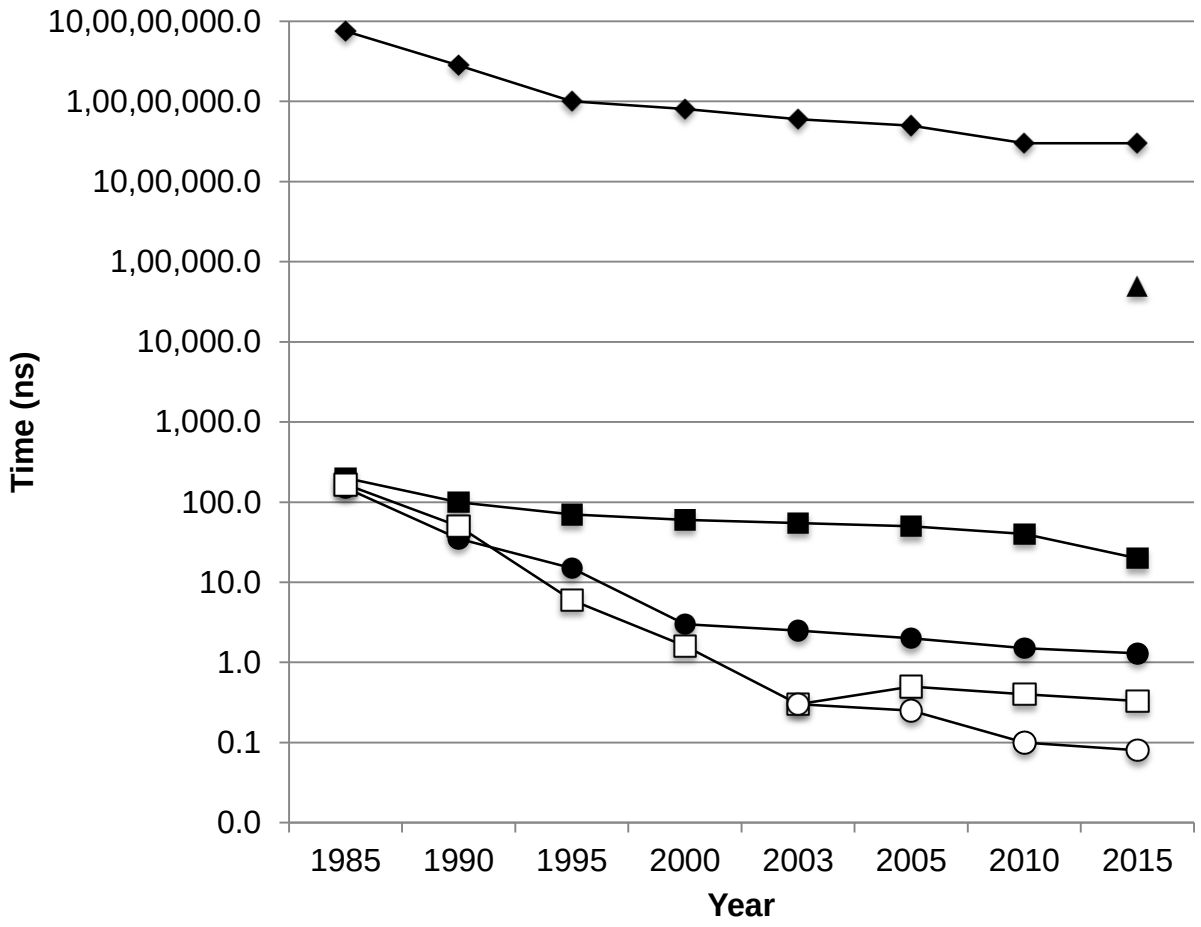
Lecture-4
(Cache organization: 10K feet view)
CS422-Spring 2019

Biswa@CSE-IITK

10,000 Feet View on Caches

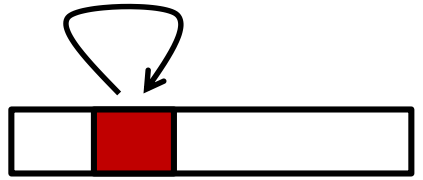


$$: Cache
Speculation technique

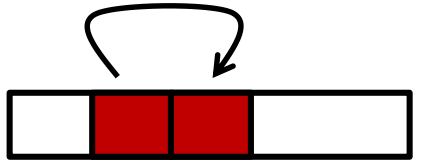
Speculation works because of locality

Locality

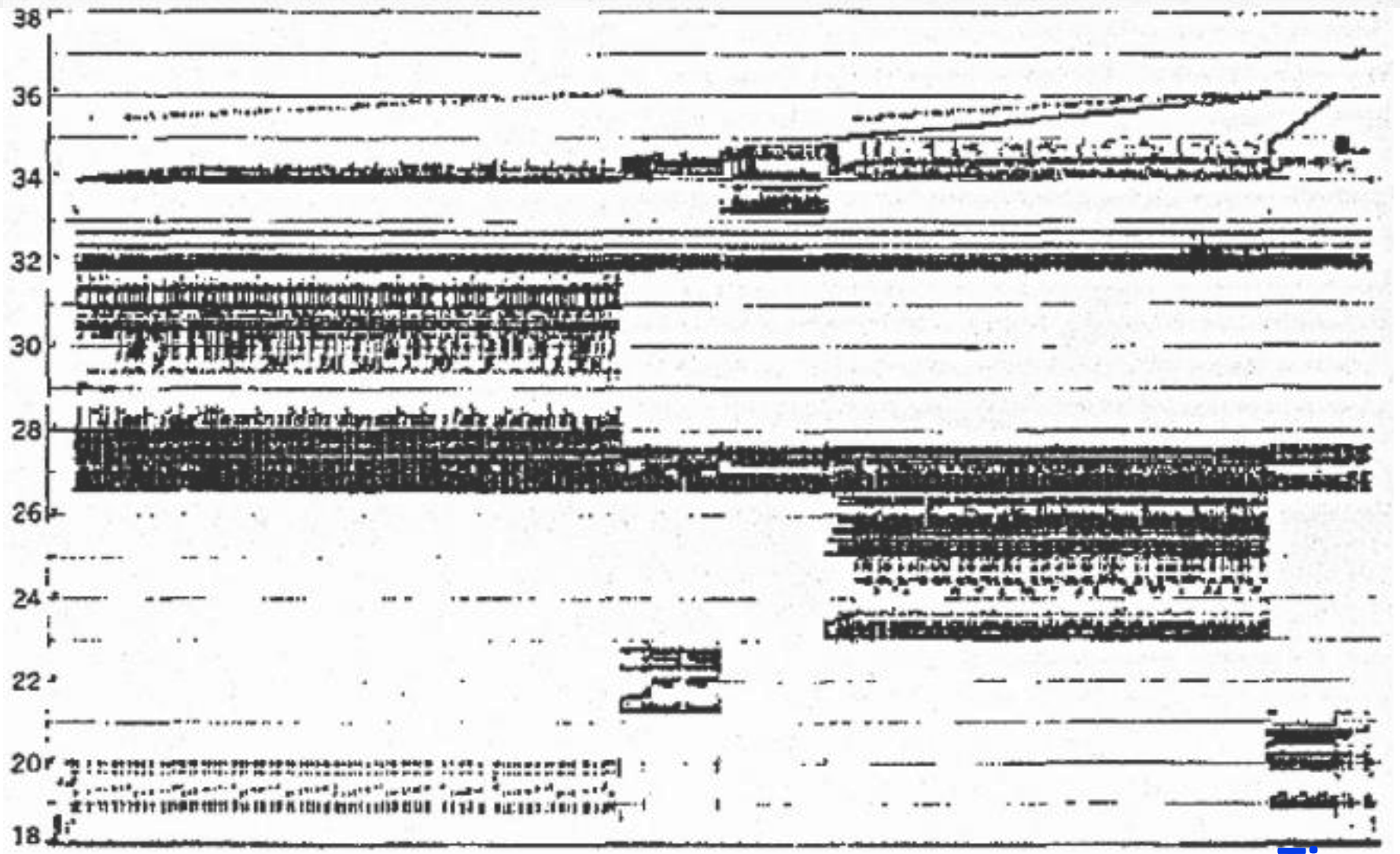
- Temporal locality:
  - Recently referenced items are likely to be referenced again



- Spatial locality:
  - Items with nearby addresses tend to be referenced again

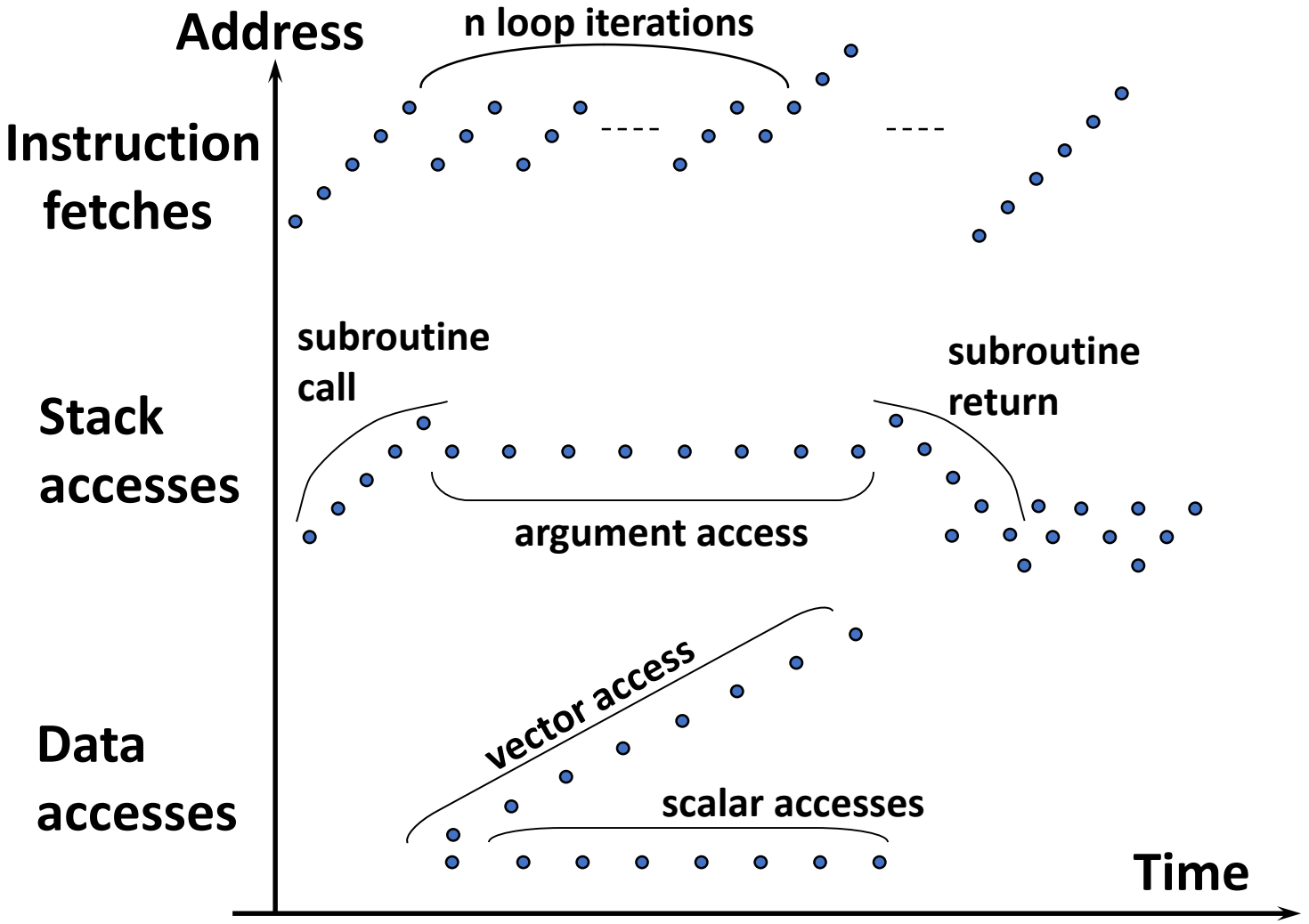


Access Patterns



Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)

Examples

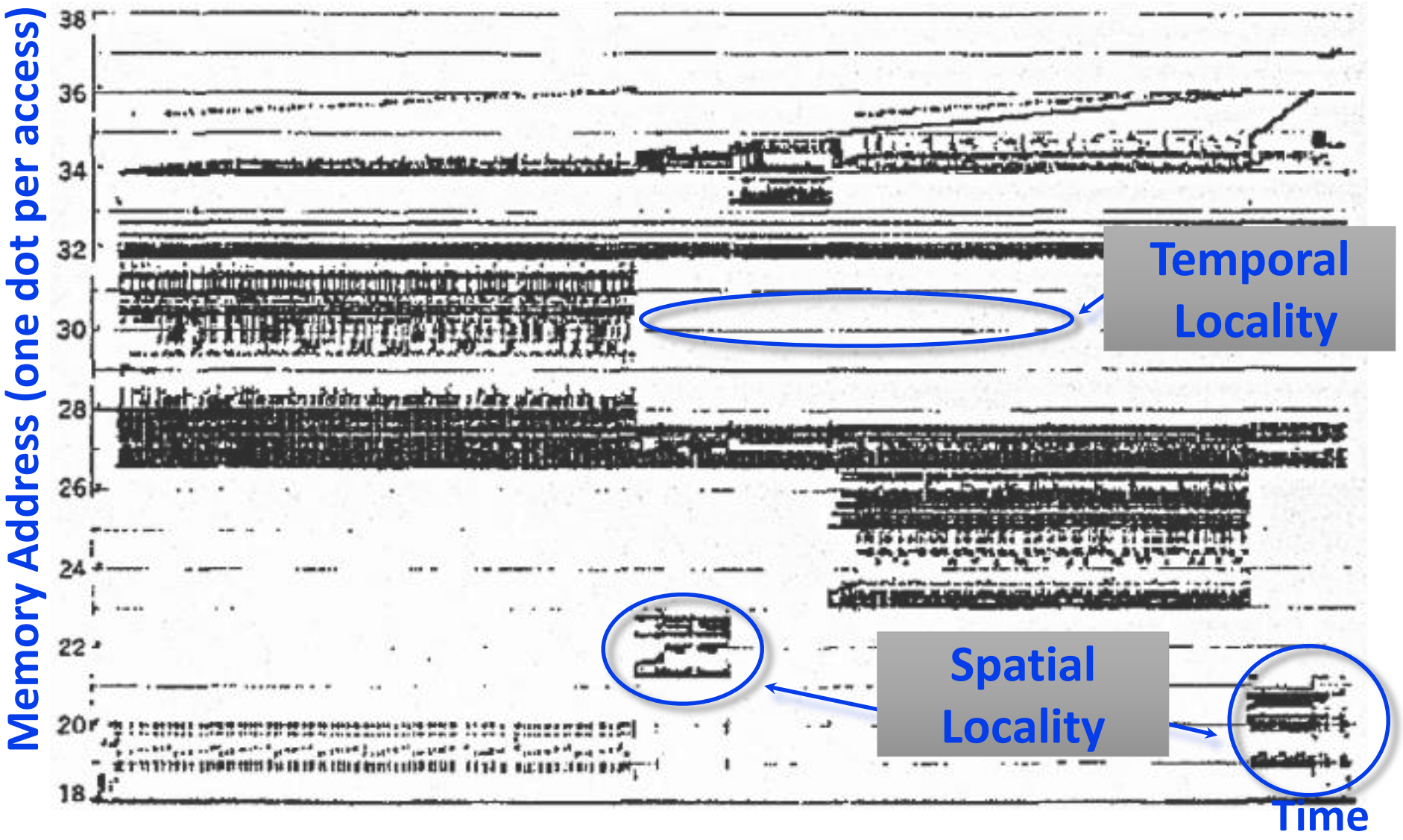

Address
n loop iterations
Instruction fetches
subroutine call
subroutine return
Stack accesses
argument access
vector access
scalar accesses
Data accesses
Time

Locality of Reference

- **Temporal Locality**: If a location is referenced it is likely to be referenced again in the near future.

- **Spatial Locality**: If a location is referenced it is likely that locations near it will be referenced in the near future.

Again


Memory Address (one dot per access)
38
36
34
32
30
28
26
24
22
20
18
Temporal Locality
Spatial Locality
Time

Locality: Example

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

Spatial/Temporal Locality?

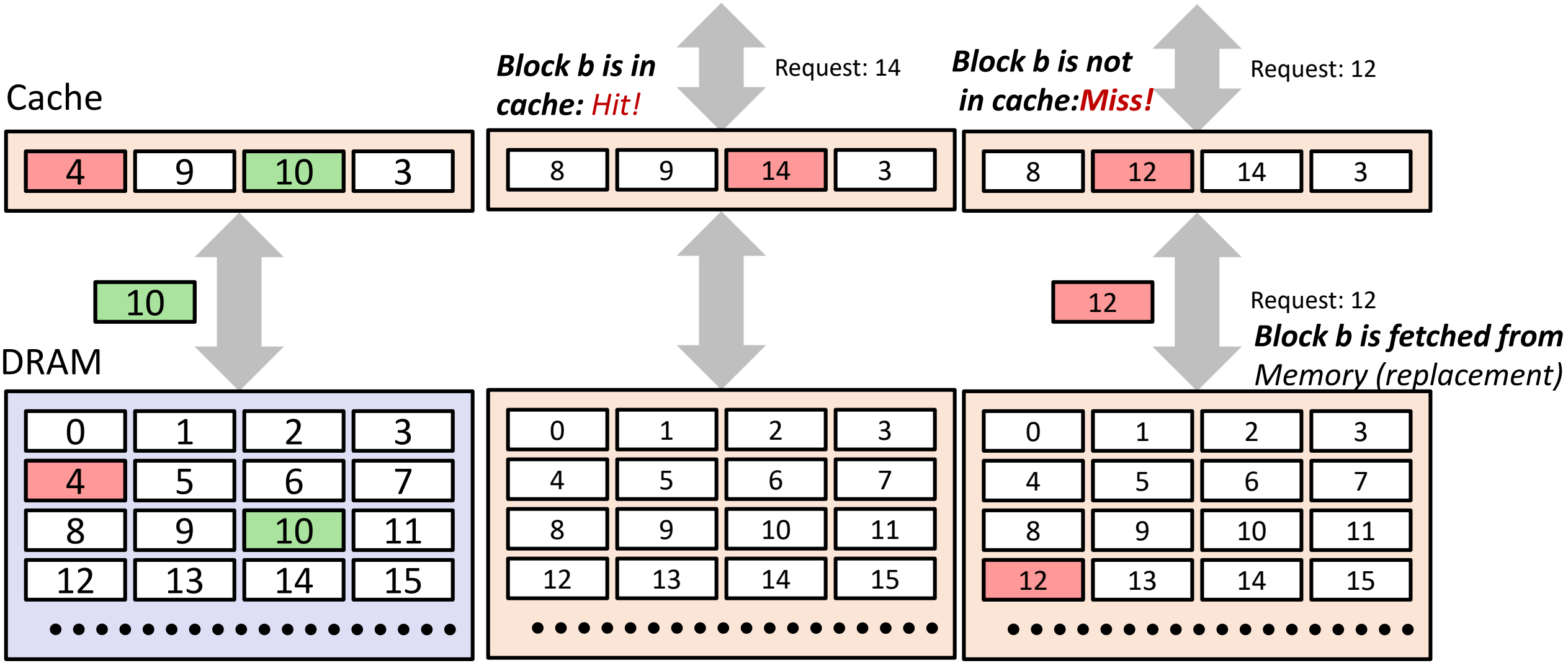
- Data references
  - Reference array elements in succession (stride-1 reference pattern). — spatial
  - Reference variable **sum** each iteration. — temporal

Wake-up Test: Improve Spatial Locality

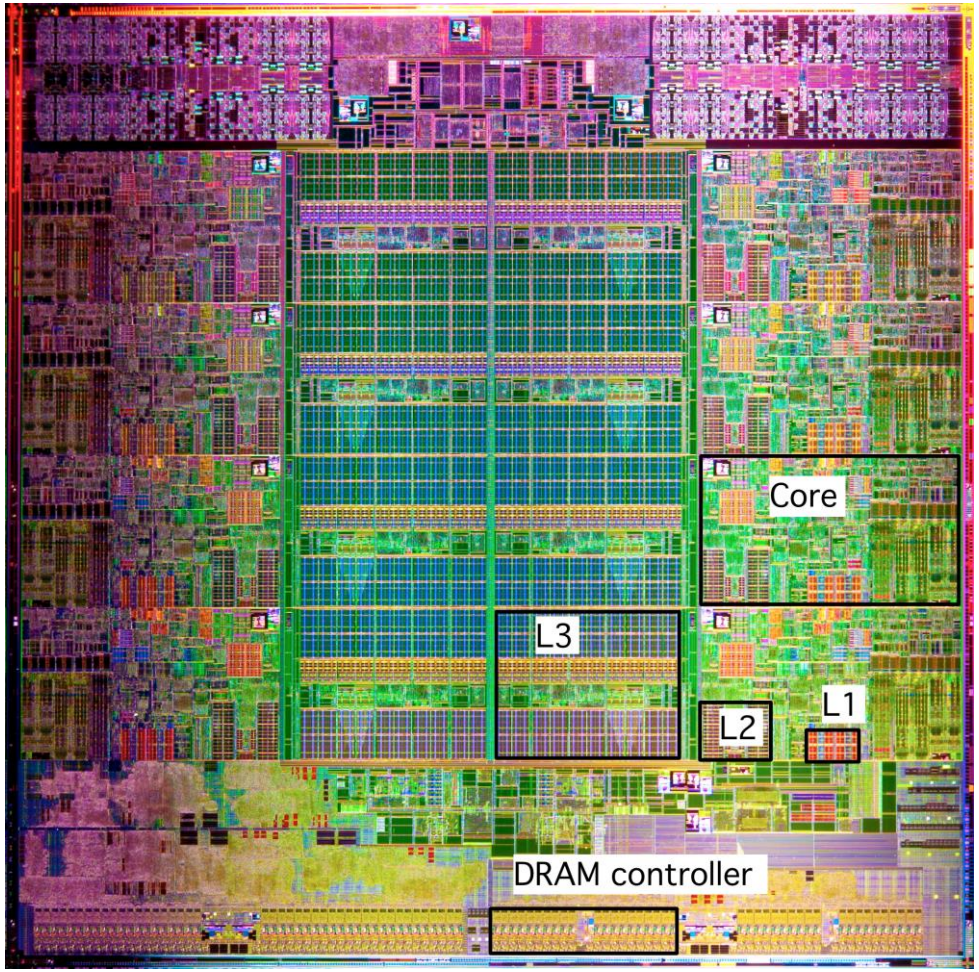
```
int sum_array_3d(int a[M][N][N])
{
    int i, j, k, sum = 0;

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

Cache and DRAM



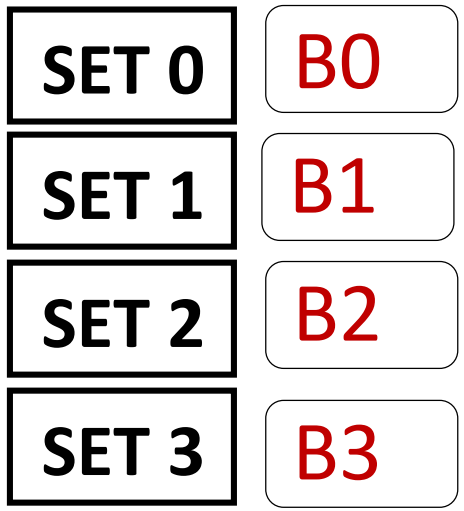
Look Like This



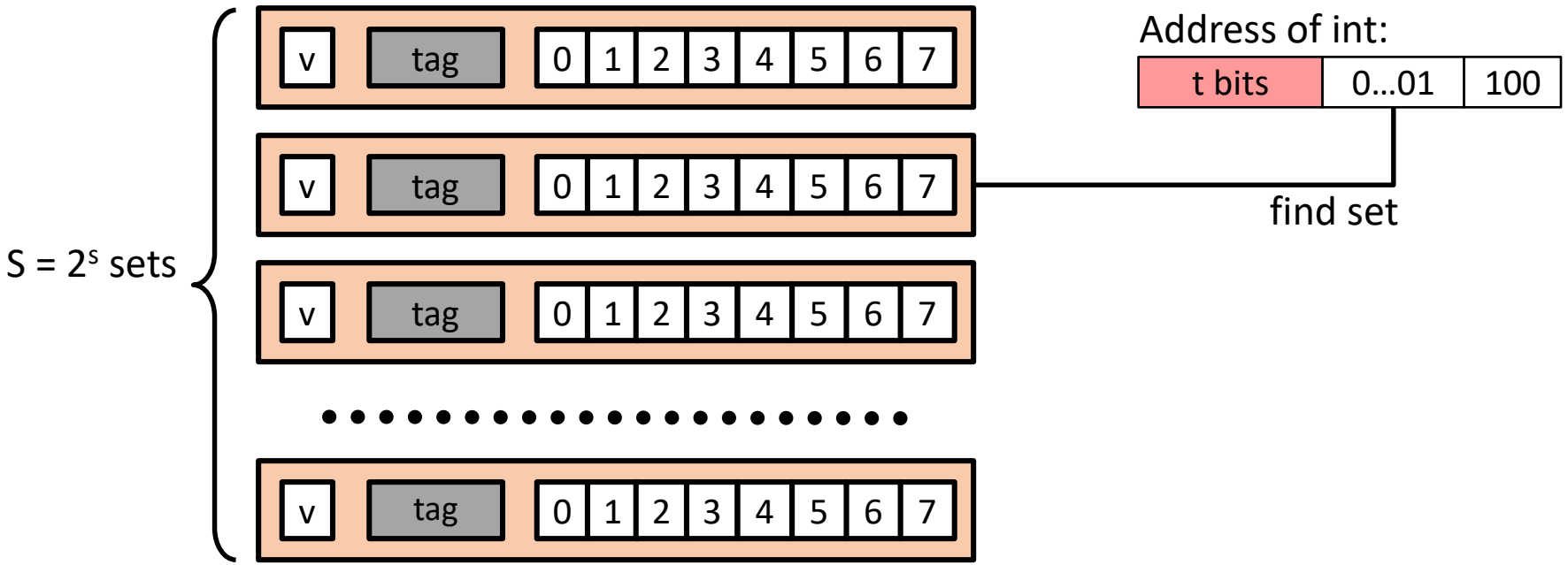
Intel Sandy Bridge Processor Die

L1: 32KB Instruction + 32KB Data
L2: 256KB
L3: 3–20MB

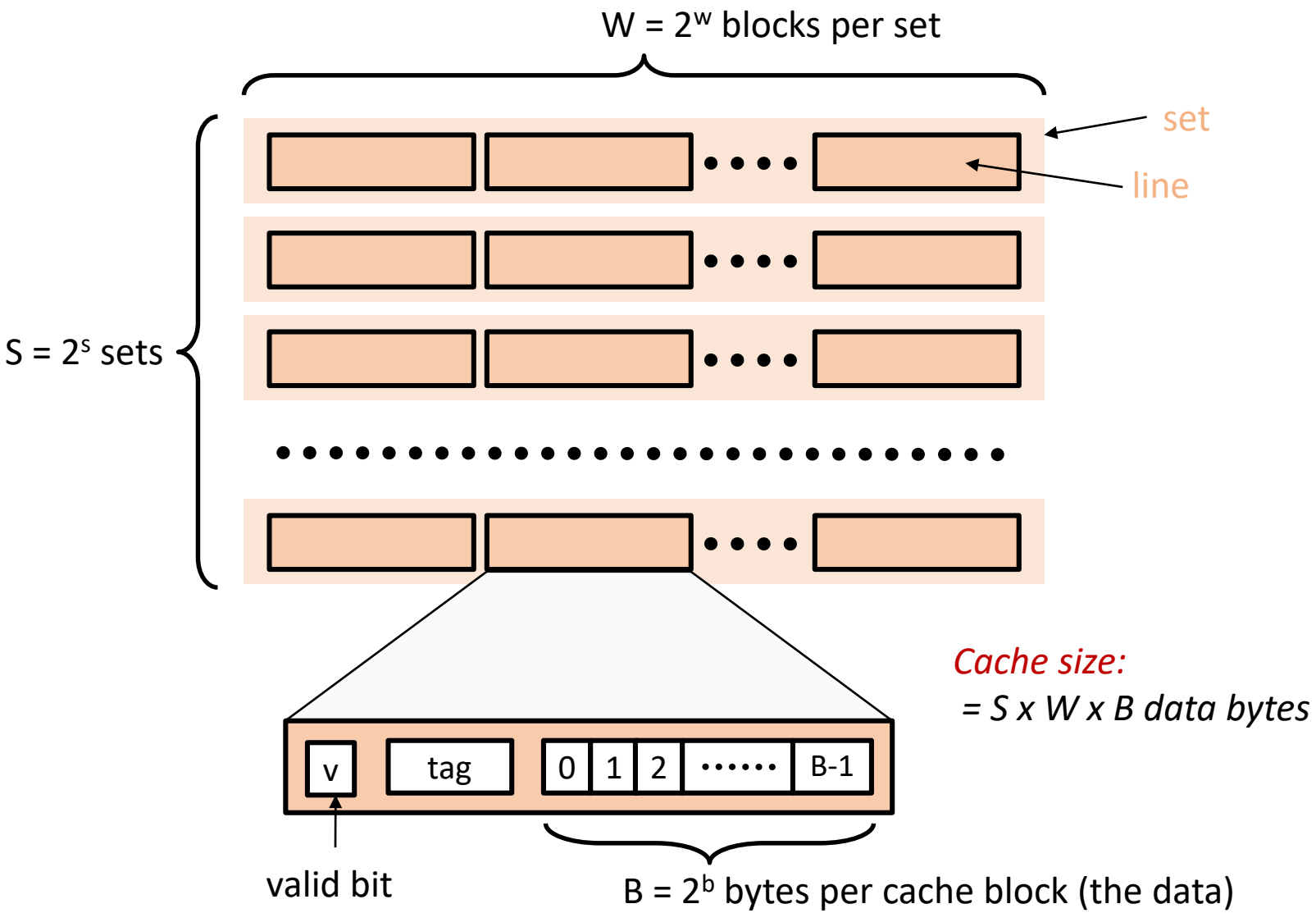
Cache Mapping



Direct Mapped: One block=One set



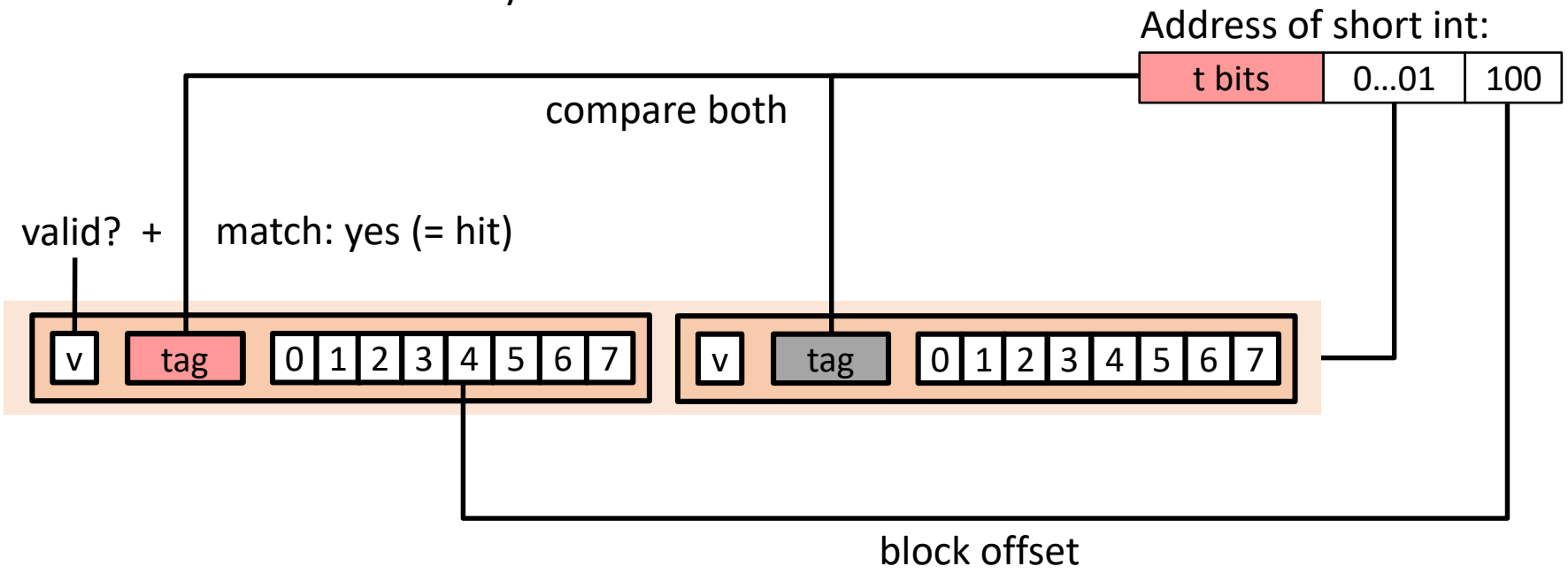
Set Associative



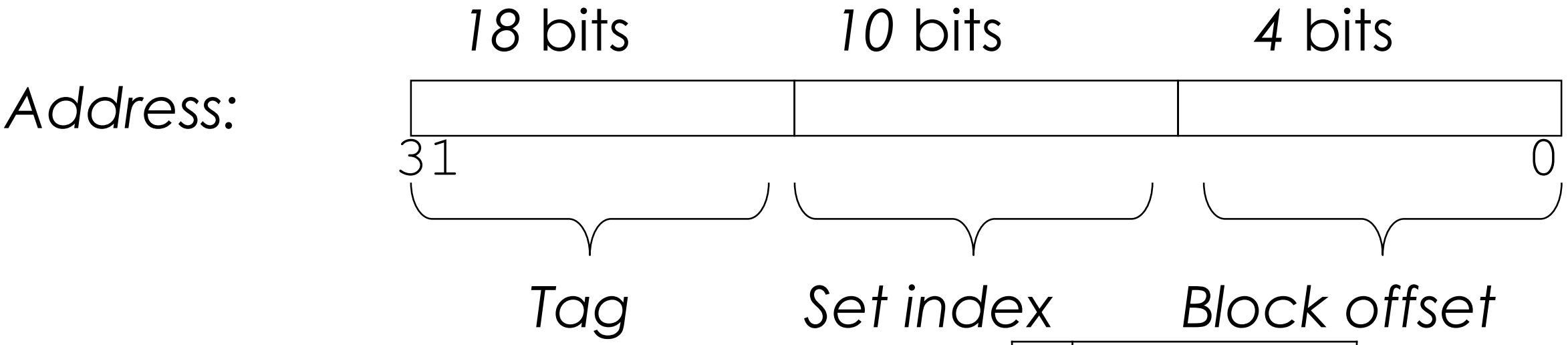
Cache size:
= S x W x B data bytes

Set Associative in Action:

Way = 2: Two lines per set
Assume: cache block size B=8 bytes



Wake-up test again: #ints inside a block?



	# of int in block
A.	0
B.	1
C.	2
D.	4
E.	Unknown: We need more info

Wake-up test again:

If N = 16, how many bytes does the loop access of a?

```
int bootcamp(int* a, int N)
{
    int i;
    int sum = 0;
    for(i = 0; i < N; i++)
    {
        sum += a[i];
    }
    return sum;
}
```

	Accessed Bytes
A	4
B	16
C	64
D	256

Performance

- Average Memory Access Time (AMAT)
- Hit Time + Miss Rate * Miss Penalty
- Try to improve Hit Time (programmer can’t do much)
- Improve Miss Rate (Yes, you can)
- Miss Penalty (Yes, a bit tricky)

The 3Cs

- Cold (compulsory) miss
  - Cold misses occur because the cache starts empty and this is the first reference
- Capacity miss
  - Occurs when the set of active cache blocks (working set) is larger than the cache.
- Conflict miss

The 3Rs

- Reduce: Misses
- Rearrange: Layout
- Reuse: Exploit spatial and temporal locality

Performance

- Huge difference between a hit and a miss
  - Could be 100x, if just L1 and main memory

- Would you believe 99% hits is twice as good as 97%?
  - Consider this simplified example:
    cache hit time of 1 cycle
    miss penalty of 100 cycles

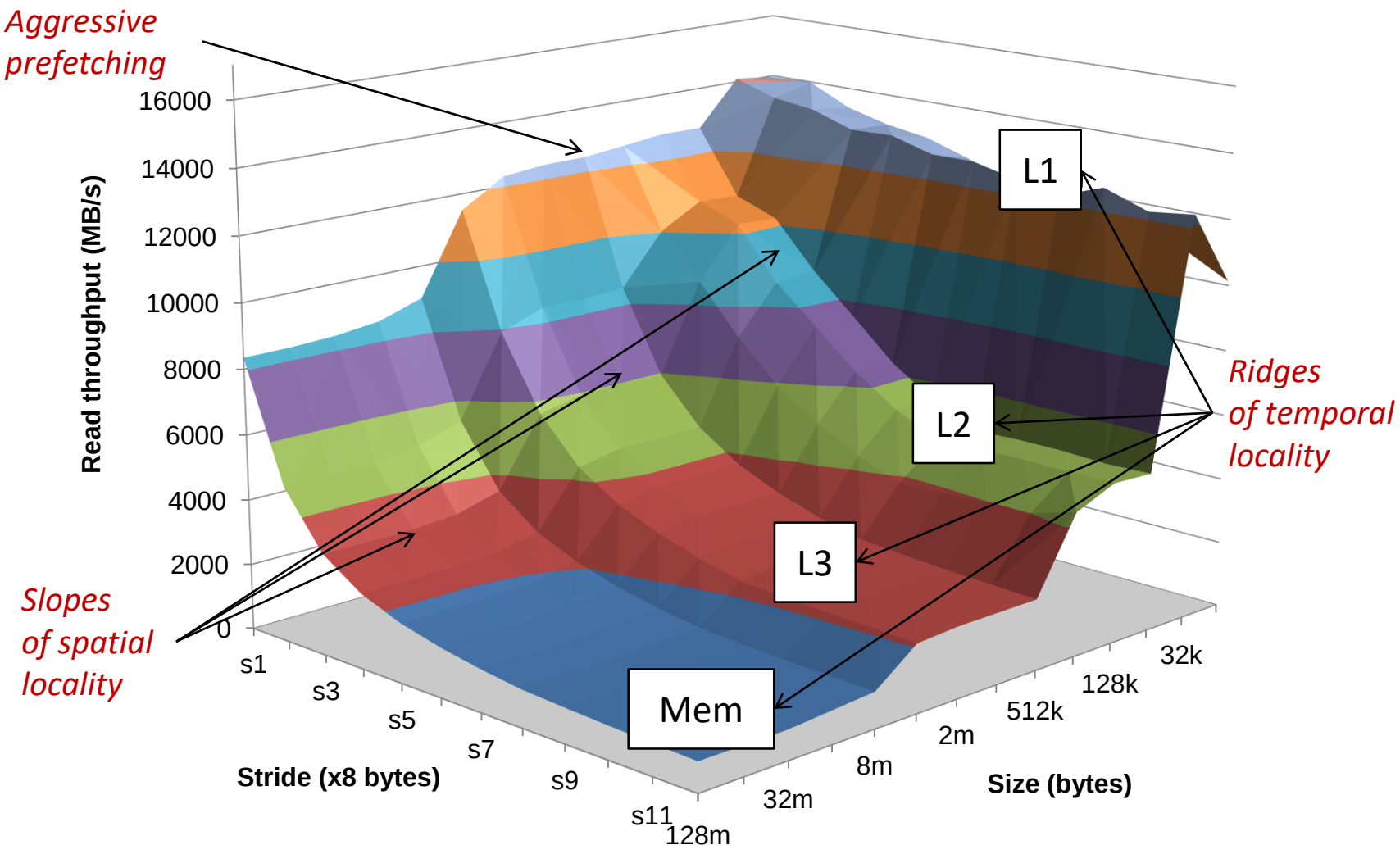
  - Average access time:

    97% hits: 1 cycle + 0.03 x 100 cycles = **4 cycles**

    99% hits: 1 cycle + 0.01 x 100 cycles = **2 cycles**

- This is why “miss rate” is used instead of “hit rate”

Memory Mountain



Core i7 Haswell

2.1 GHz
32 KB L1 d-cache

256 KB L2 cache

8 MB L3 cache

64 B block size