Memory Management

Memory Technologies

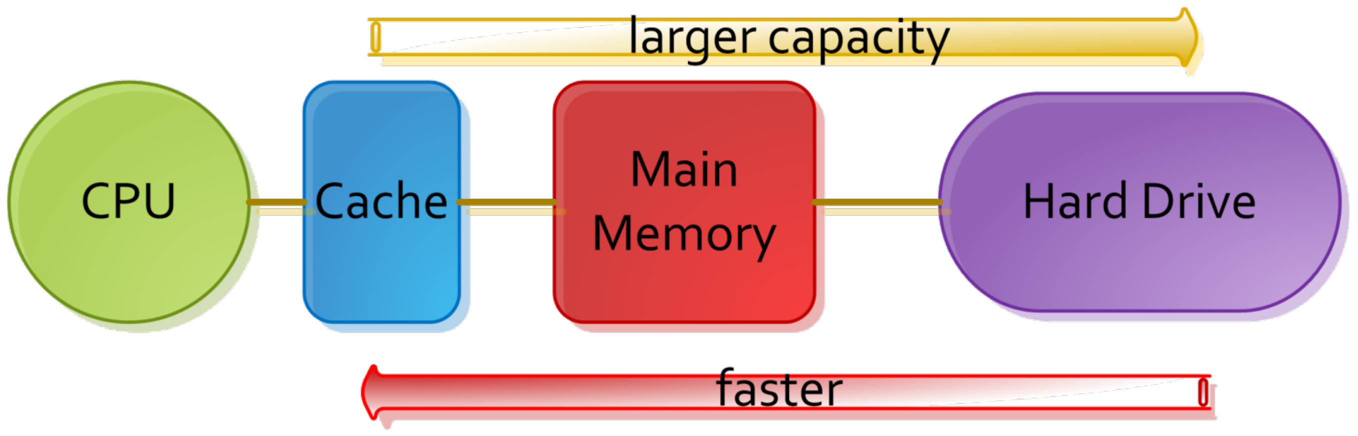
	Capacity	Latency	Cost/GB
Registers	1000s of bits	20 ps	$$$$
SRAM	~10KB-10MB	1-10 ns	~$1000
DRAM	~10GB	80 us	~10
Hard disk	~1TB	10 ms	~0.10

Different technologies have vastly different tradeoffs.

small, low latency

large, high latency

Memory Hierarchy Levels

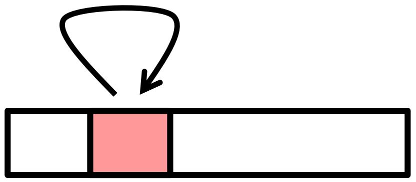


Levels in a typical memory hierarchy.

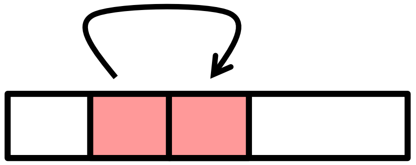
Closer to CPU, memory becomes faster , smaller, as well as more expensive per bit

Locality

- Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently
- Temporal locality:
  - Recently referenced items are likely to be referenced again 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;
```

- Data references
  - Reference array elements in succession — Spatial locality
  - Reference variable **`sum`** each iteration. — Temporal locality
- Instruction references
  - Reference instructions in sequence. — Spatial locality
  - Cycle through loop repeatedly. — Temporal locality

Locality Example

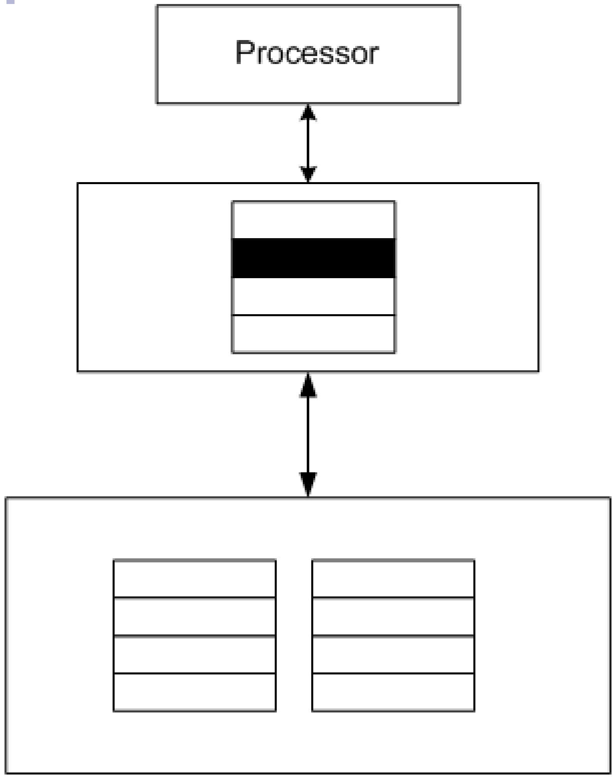
- Claim: Being able to look at code and get a qualitative sense of its locality is a key skill for a professional programmer.

- Question: Which function has good locality with respect to array `a`?

```
int sum_array_rows(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;
}
```

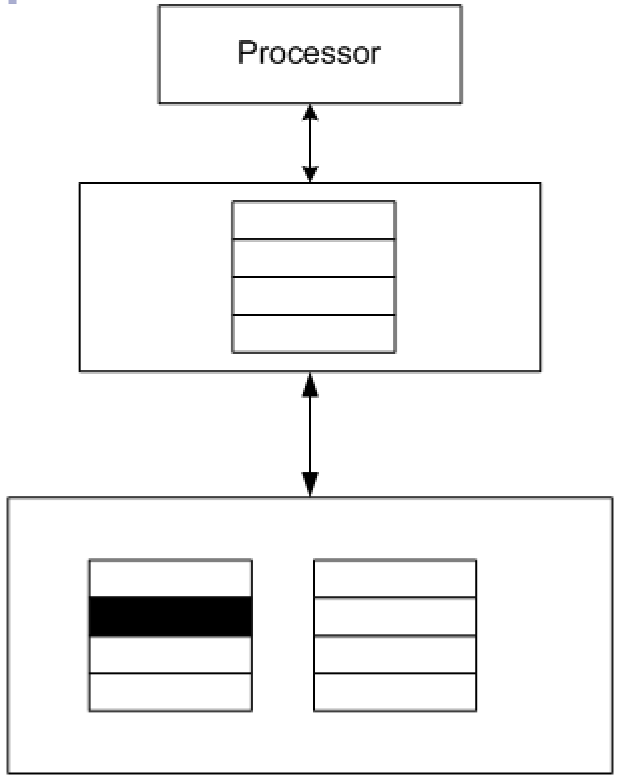
```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

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

Memory Hierarchy Levels



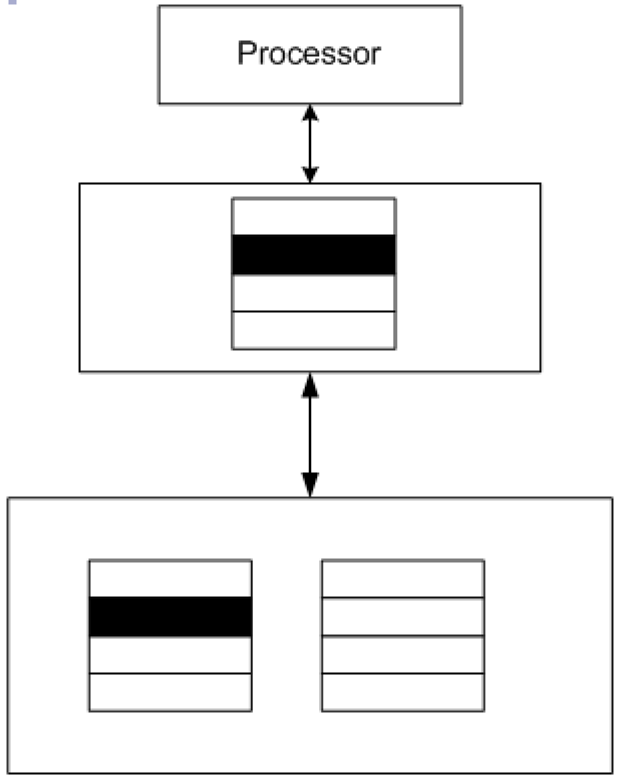
- Block (aka line): unit of copying
  - May be multiple words
- If accessed data is present in upper level
  - Hit: access satisfied by upper level
    - Hit ratio: hits/accesses

Memory Hierarchy Levels



- If accessed data is absent
  - Miss:
    - Miss ratio: misses/accesses
      = 1 – hit ratio
  - Copy the block from lower level to upper level
  - Then accessed data is available in upper level

Memory Hierarchy Levels



- If accessed data is absent
  - Miss
    - Miss ratio: misses/accesses
      = 1 – hit ratio
  - Copy the block from lower level to upper level
  - Then accessed data is available in upper level

Cache Performance Metrics

- Hit Time
  - Time to deliver a line in the cache to the processor
    - includes time to determine whether the line is in the cache
  - Typical numbers:
    - 1~4 clock cycle for L1
    - 10~100 clock cycles for L2
- Miss Penalty
  - Additional time required because of a miss
    - typically 50-200 cycles for main memory

Let’s think about those numbers

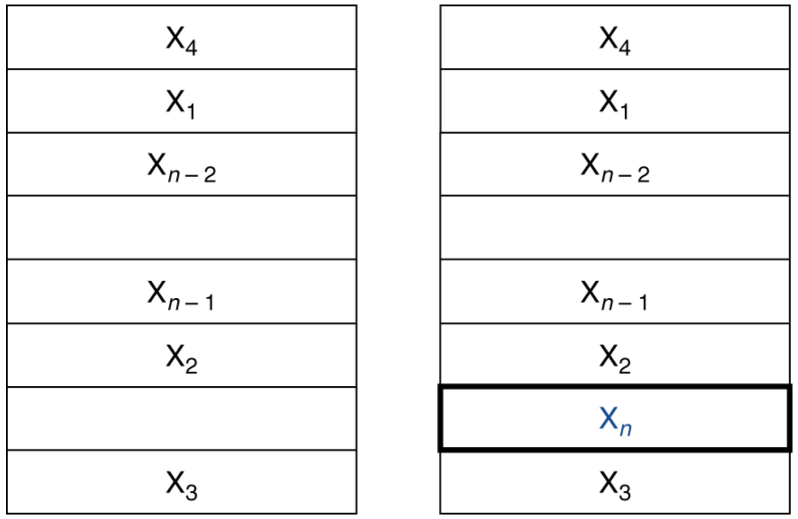
- Huge difference between a hit and a miss
  - Could be 100x, if just L1 and main memory
- Would you believe 99% hits is twice as good as 97%?
  - Consider:
    cache hit time of 1 cycle
    miss penalty of 100 cycles
  - Average access time: HitTime + MissRatio x MissPenalty

    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”

Cache Memory

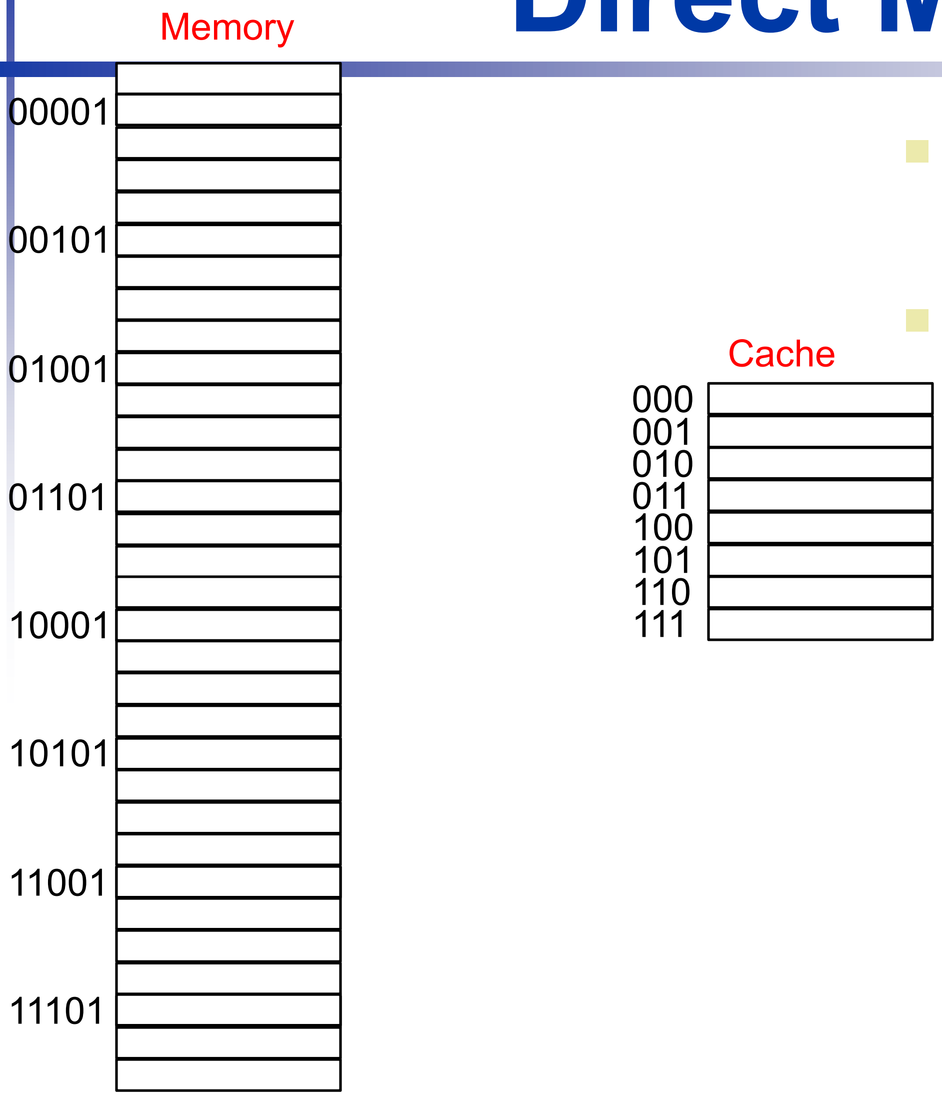
- Cache memory
  - The level of the memory hierarchy closest to the CPU
- Given accesses $X_1, \ldots, X_{n-1}, X_n$



a. Before the reference to X_n b. After the reference to X_n

- How do we know if the data is present?
- Where do we look?

Direct Mapped Cache



- Location determined by address
- Direct mapped: only one choice
  - (Block address in memory) modulo (#Blocks in cache)
- Use low-order address bits

Tags and Valid Bits

- How do we know which particular block is stored in a cache location?
  - Store block address as well as the data
  - Actually, only need the high-order bits
  - Called the tag
- What if there is no data in a location?
  - Valid bit: 1 = present, 0 = not present
  - Initially: valid bit =0;

Cache Example

- 8-blocks, 1 word/block, direct mapped
- Initial state

Index	V	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	0		
111	0		

Cache Example

Cache: word address means block address

Index	V	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	0		
111	0		

Memory:

Word addr	Binary addr	Hit/miss	Cache index
22	10 110	Miss	110

Cache Example

Cache: word address means block address

Index	V	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Memory:

Word addr	Binary addr	Hit/miss	Cache index
22	10 110	Miss	110

Cache Example

Cache: word address means block address

Index	V	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Memory:

Word addr	Binary addr	Hit/miss	Cache index
26	11 010	Miss	010

Cache Example

Cache: word address means block address

Index	V	Tag	Data
000	0		
001	0		
010	**1**	**11**	**Mem[11010]**
011	0		
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Memory:

Word addr	Binary addr	Hit/miss	Cache index
26	11 010	Miss	010

Cache Example

Cache: word address means block address

Index	V	Tag	Data
000	0		
001	0		
010	1	11	Mem[11010]
011	0		
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Memory:

Word addr	Binary addr	Hit/miss	Cache index
22	10 110	Hit	110
26	11 010	Hit	010

Cache Example

Cache: word address means block address

Index	V	Tag	Data
000	0		
001	0		
010	1	11	Mem[11010]
011	0		
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Memory:

Word addr	Binary addr	Hit/miss	Cache index
18	10 010	Miss	010

Cache Example

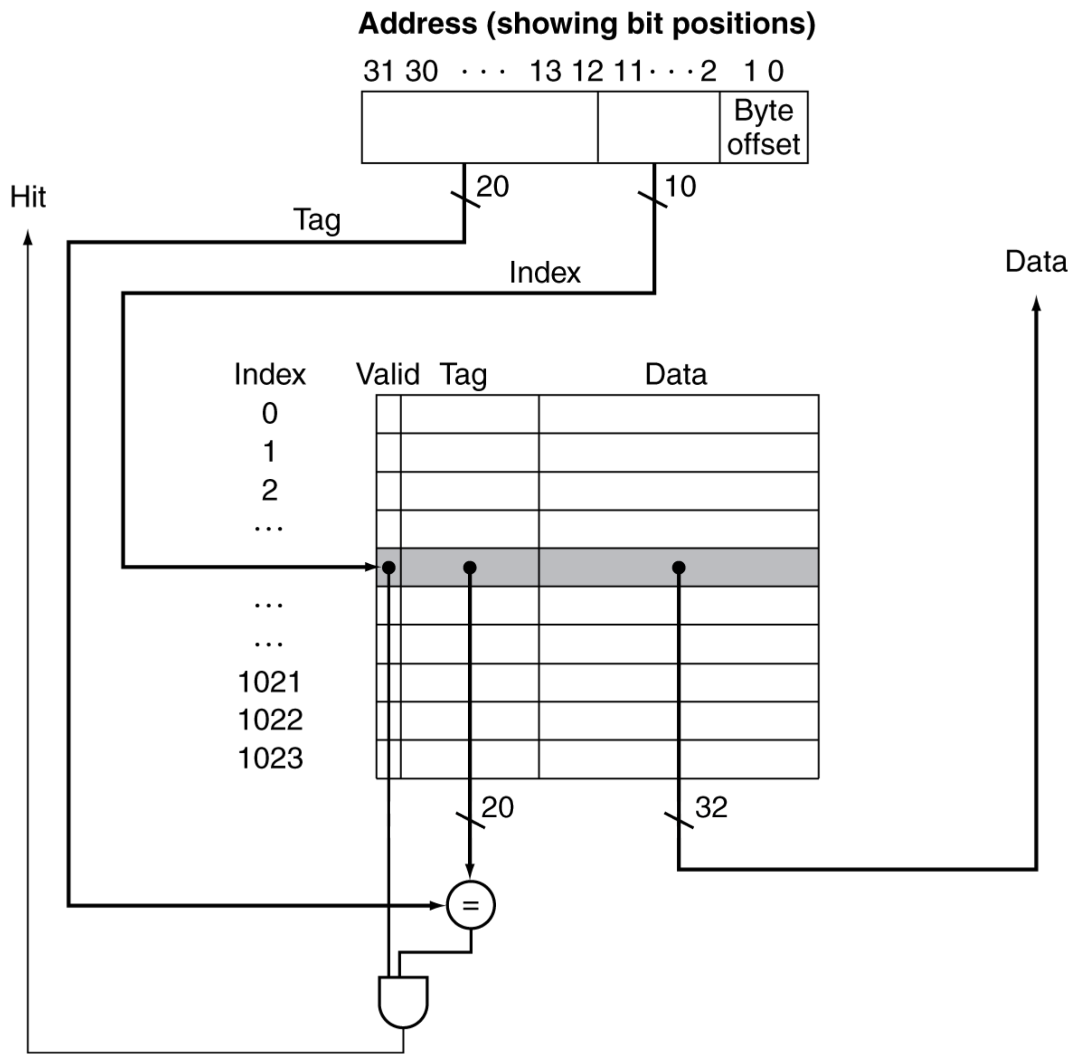
Cache: word address means block address

Index	V	Tag	Data
000	0		
001	0		
010	**1**	**10**	**Mem[10010]**
011	0		
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

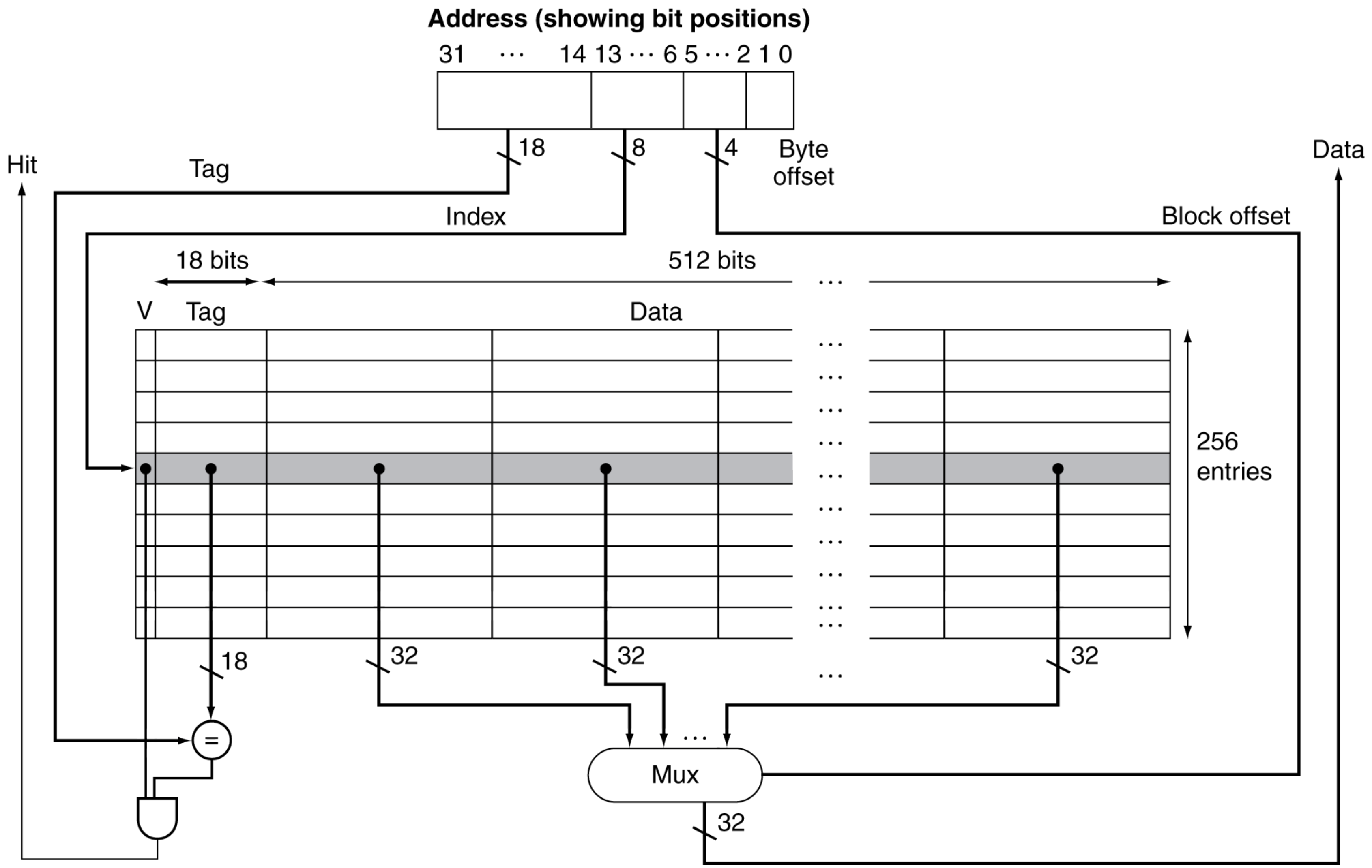
Memory:

Word addr	Binary addr	Hit/miss	Cache index
18	10 010	Miss	010

Address Subdivision



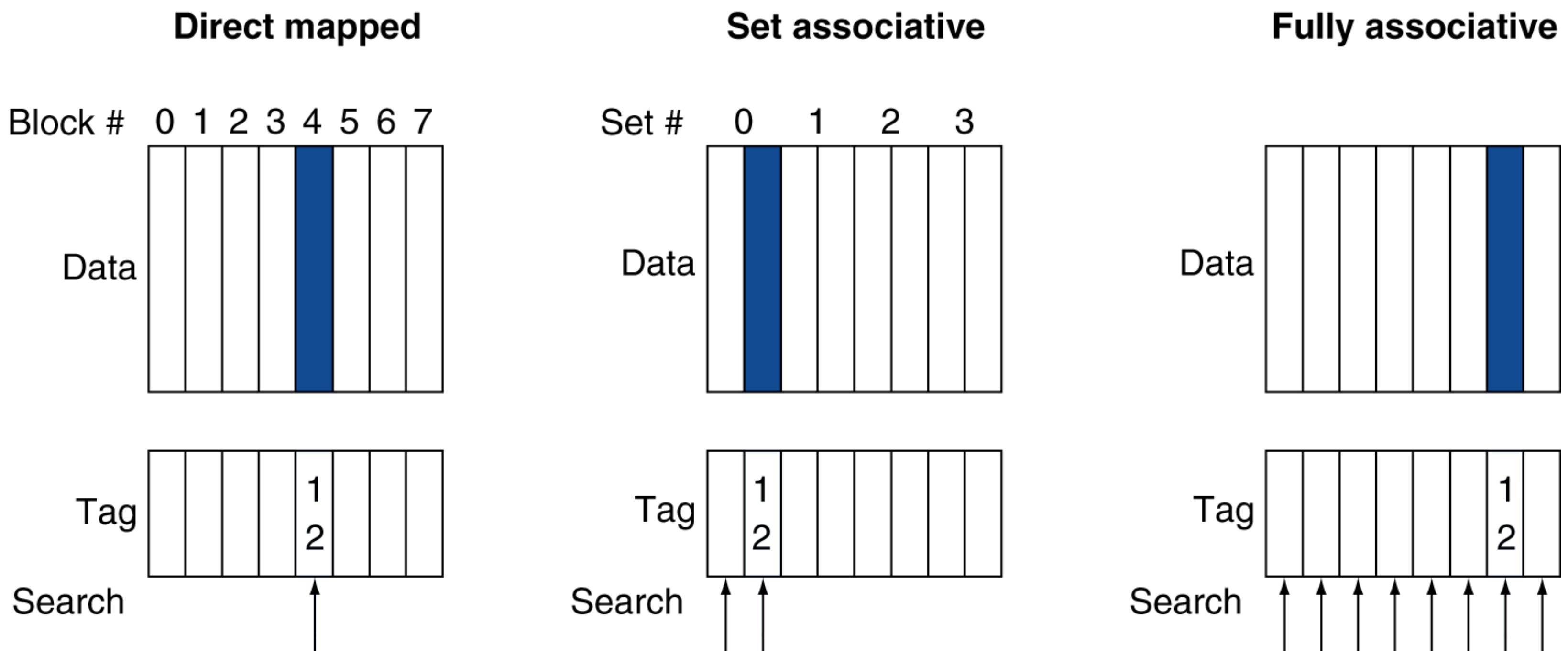
Example: Intrinsity FastMATH



Block Size Tradeoffs

- Larger block sizes
  - Take advantage of spatial locality
  - Incur larger miss penalty since it takes longer to transfer the block into the cache
  - Can increase the average hit time and miss rate

Associative Cache Example

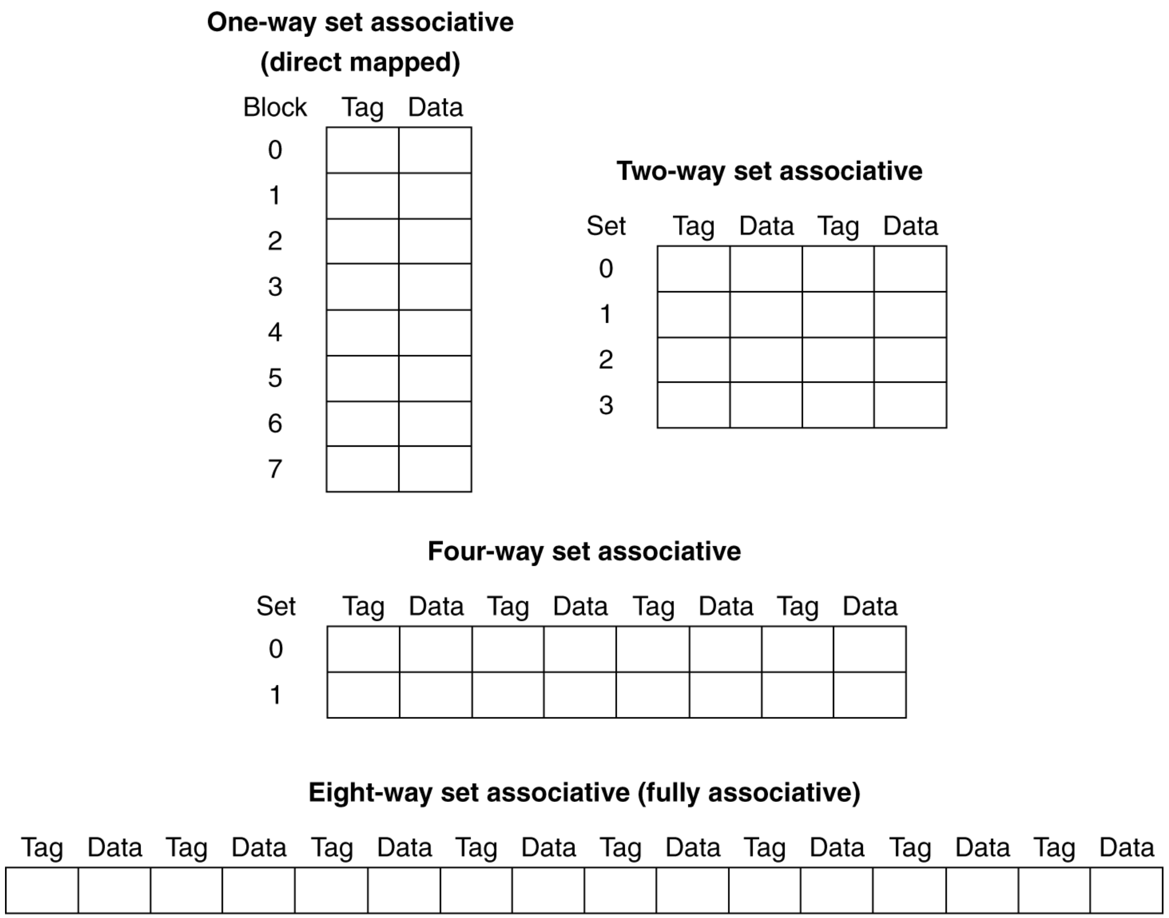


Reducing Cache Miss: #1 Associative Caches

- Direct mapped cache
  - A memory block maps to exactly one cache block
- Fully associative
  - Allow a given block to go in any cache entry
  - Requires all entries to be searched at once
  - Comparator per entry (expensive)
- *n*-way set associative
  - Each set contains *n* entries
  - Block address determines which set
    - (Block address) modulo (#Sets in cache)
  - Search all entries in a given set at once
  - *n* comparators (less expensive)

Spectrum of Associativity

- For a cache with 8 entries



Associativity Example

- Compare 4-block caches
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 6, 8
- Direct mapped

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0						
8						
0						
6						
8						

Associativity Example

- Compare 4-block caches
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 6, 8
- Direct mapped

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0					
8	0					
0	0					
6	2					
8	0					

Associativity Example

- Compare 4-block caches
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 6, 8
- Direct mapped

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0					
6	2					
8	0					

Associativity Example

- Compare 4-block caches
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 6, 8
- Direct mapped

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	Mem[0]			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2					
8	0					

Associativity Example

- Compare 4-block caches
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 6, 8
- Direct mapped

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2	miss	Mem[0]		**Mem[6]**	
8	0					

Associativity Example

- Compare 4-block caches
  - Direct mapped, 2-way set associative, fully associative
  - Block address access sequence: 0, 8, 0, 6, 8
- Direct mapped

address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2	miss	Mem[0]		**Mem[6]**	
8	0	miss	**Mem[8]**		Mem[6]	

Associativity Example

- 2-way set associative

address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0						
8						
0						
6						
8						

Associativity Example

- 2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0					
8	0					
0	0					
6	0					
8	0					

Associativity Example

- 2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	Mem[0]			
8	0					
0	0					
6	0					
8	0					

Associativity Example

- 2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0					
6	0					
8	0					

Associativity Example

- 2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0					
8	0					

Associativity Example

- 2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

Associativity Example

- Fully associative

Block address		Hit/miss	Cache content after access			
0						
8						
0						
6						
8						

Associativity Example

- Fully associative

Block address		Hit/miss	Cache content after access			
0		miss	Mem[0]			
8						
0						
6						
8						

Associativity Example

- Fully associative

Block address		Hit/miss	Cache content after access			
0		miss	Mem[0]			
8		miss	Mem[0]	Mem[8]		
0						
6						
8						

Associativity Example

- Fully associative

Block address		Hit/miss	Cache content after access			
0		miss	Mem[0]			
8		miss	Mem[0]	Mem[8]		
0		hit	**Mem[0]**	Mem[8]		
6						
8						

Associativity Example

- Fully associative

Block address		Hit/miss	Cache content after access			
0		miss	Mem[0]			
8		miss	Mem[0]	Mem[8]		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	Mem[6]	
8						

Associativity Example

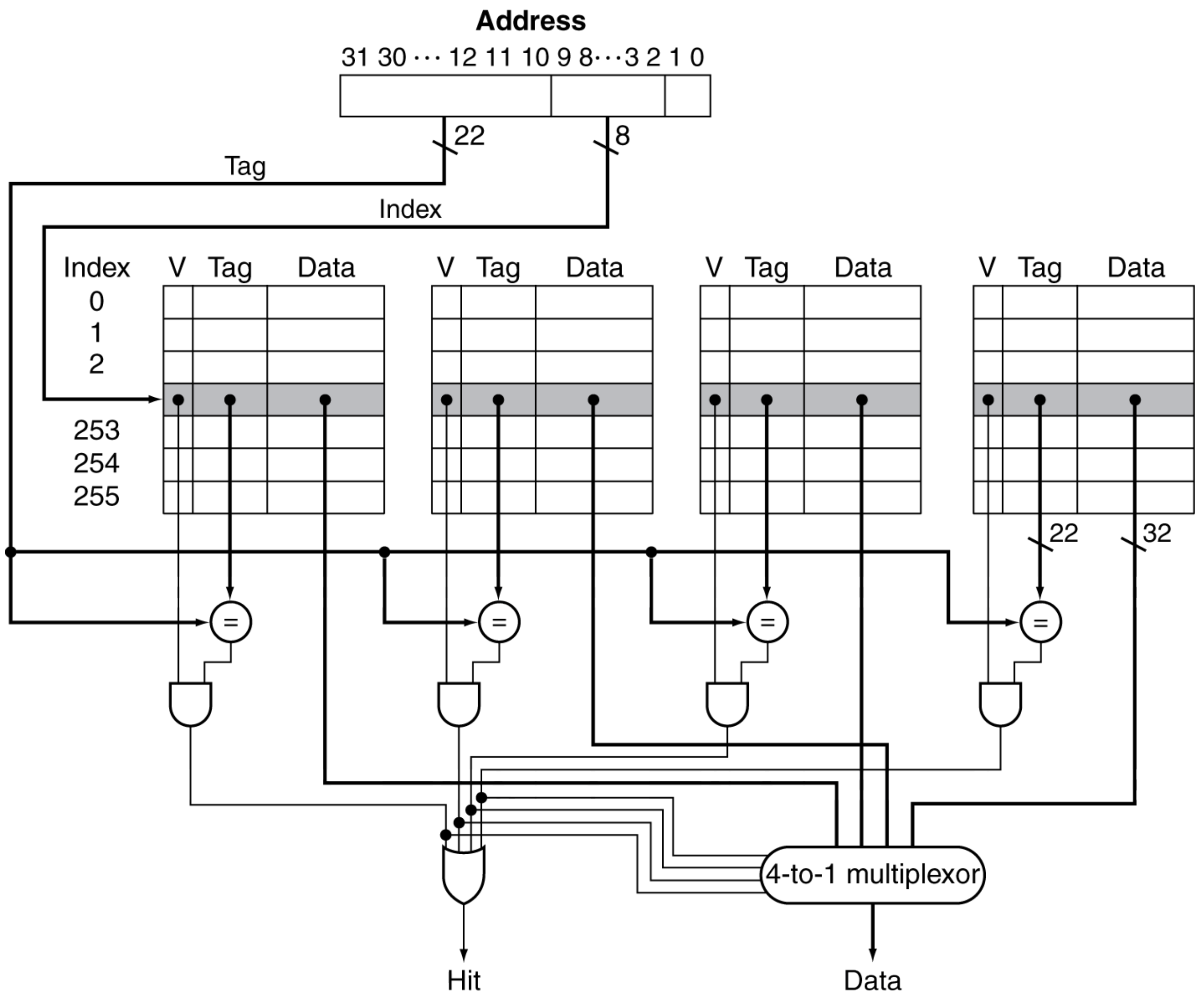
- Fully associative

Block address		Hit/miss	Cache content after access			
0		miss	Mem[0]			
8		miss	Mem[0]	Mem[8]		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	Mem[6]	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

How Much Associativity

- Simulation of a system with 64KB D-cache, 16-word blocks, SPEC2000: (data miss rate)
  - 1-way: 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%
  - 8-way: 8.1%

Set Associative Cache Organization



Associativity Implies Choices

- Direct-mapped
  - Compare addr. with only one tag
  - Location A can be stored in exactly one cache line
- N-way set-associative
  - Compare addr. with N tags simultaneously
  - Location A can be stored in exactly one set, but in any of the N cache lines belonging to that set
- Fully associative
  - Compare addr. With each tag simultaneously
  - Location A can be stored in any cache line

Replacement Policy

- Direct mapped: no choice
- Set associative
  - Prefer non-valid entry, if there is one
  - Otherwise, choose among entries in the set
- Optimal policy: Replace the block that is accessed furthest in the future
  - Requires knowing the future
- Predict the future from looking at the past
  - If a block has not been used recently, it’s often less likely to be accessed in the near future (a locality argument)
- Least-recently used (LRU)
  - Choose the one unused for the longest time

Write Policy

- Write-through:
  - CPU writes are cached, but also written to main memory immediately
  - Stalling the CPU until write is completed
  - Simple, slow
- Write-back:
  - CPU writes are cached, but not written to main memory until we replace the block
  - Commonly implemented in current systems
  - Fast, more complex

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

Write-Back with ‘Dirty’ Bits

- Add 1 bit per block to record whether block has been written to.
- Only write back dirty blocks