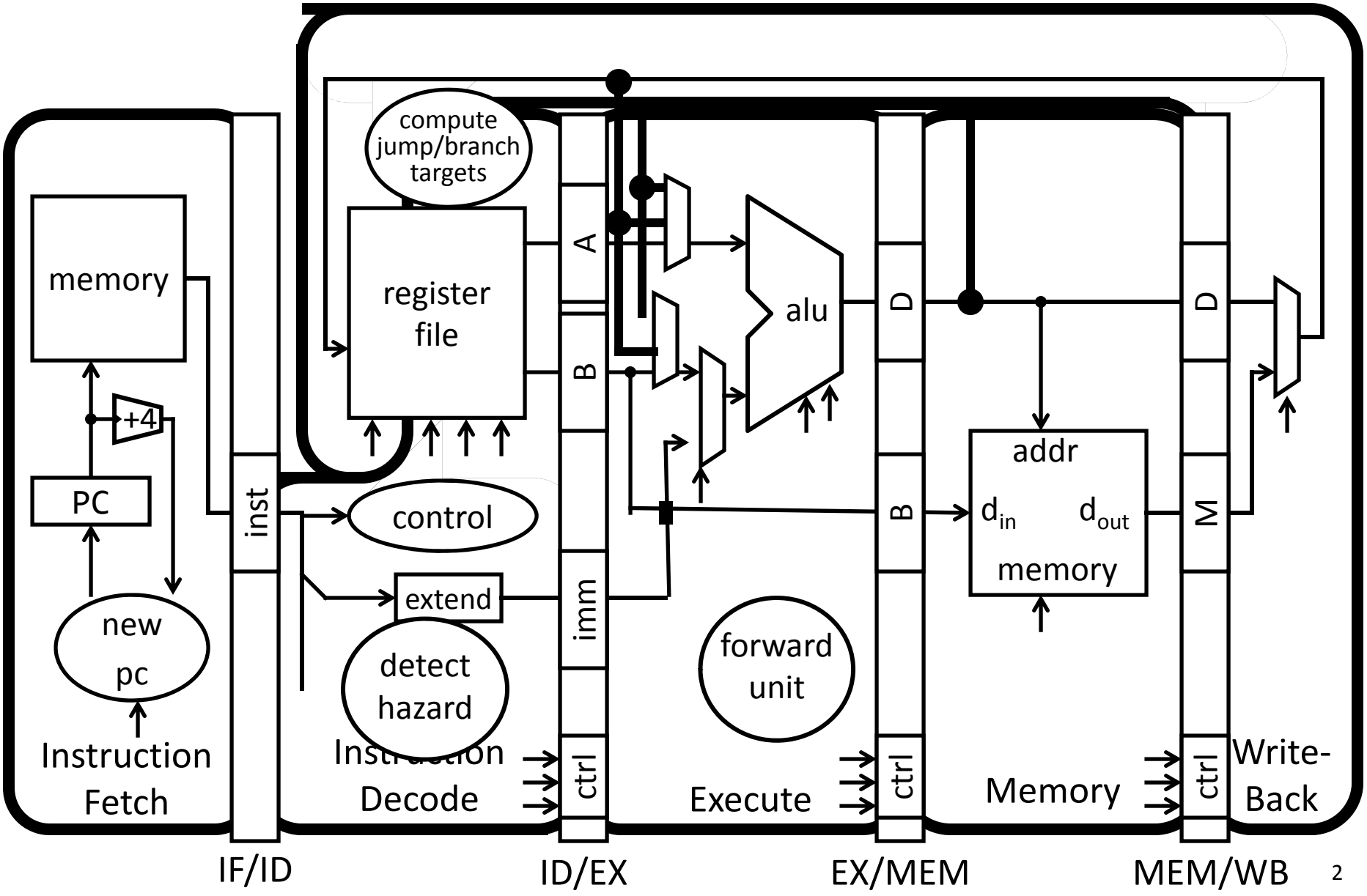
Caches

Hakim Weatherspoon
CS 3410, Spring 2012
Computer Science
Cornell University

See P&H 5.1, 5.2 (except writes)

Big Picture: Memory

Memory: big & slow vs Caches: small & fast



Administrivia

Prelim2 ***today***, Thursday, March 29th at 7:30pm

- Location is Phillips 101 and prelim2 starts at 7:30pm

Project2 due next Monday, April 2nd

Goals for Today: caches

Examples of caches:

- Direct Mapped
- Fully Associative
- N-way set associative

Performance and comparison

- Hit ratio (conversly, miss ratio)
- Average memory access time (AMAT)
- Cache size

Cache Performance

Average Memory Access Time (AMAT)

Cache Performance (very simplified):

- L1 (SRAM): 512 x 64 byte cache lines, direct mapped
  - Data cost: 3 cycle per word access
  - Lookup cost: 2 cycle
- Mem (DRAM): 4GB
  - Data cost: 50 cycle per word, plus 3 cycle per consecutive word

Performance depends on:

- Access time for hit, miss penalty, hit rate

Misses

Cache misses: classification

The line is being referenced for the first time

- Cold (aka Compulsory) Miss

The line was in the cache, but has been evicted

Avoiding Misses

Q: How to avoid…

Cold Misses

- Unavoidable? The data was never in the cache…
- Prefetching!

Other Misses

- Buy more SRAM
- Use a more flexible cache design

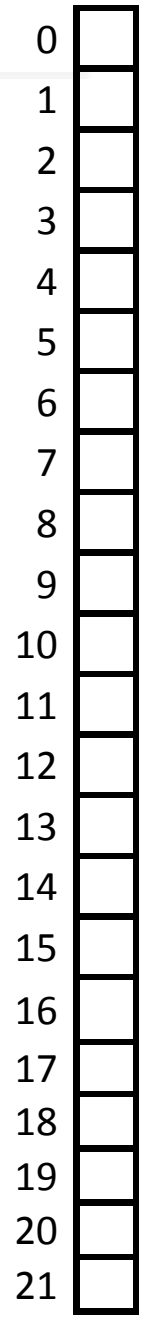
Bigger cache doesn't always help…

Mem access trace: 0, 16, 1, 17, 2, 18, 3, 19, 4, …

Hit rate with four direct-mapped 2-byte cache lines?

With eight 2-byte cache lines?

With four 4-byte cache lines?



Misses

Cache misses: classification

The line is being referenced for the first time

- Cold (aka Compulsory) Miss

The line was in the cache, but has been evicted…

… because some other access with the same index

- Conflict Miss

… because the cache is too small

- i.e. the *working set* of program is larger than the cache
- Capacity Miss

Avoiding Misses

Q: How to avoid…

Cold Misses

- Unavoidable? The data was never in the cache…
- Prefetching!

Capacity Misses

- Buy more SRAM

Conflict Misses

- Use a more flexible cache design

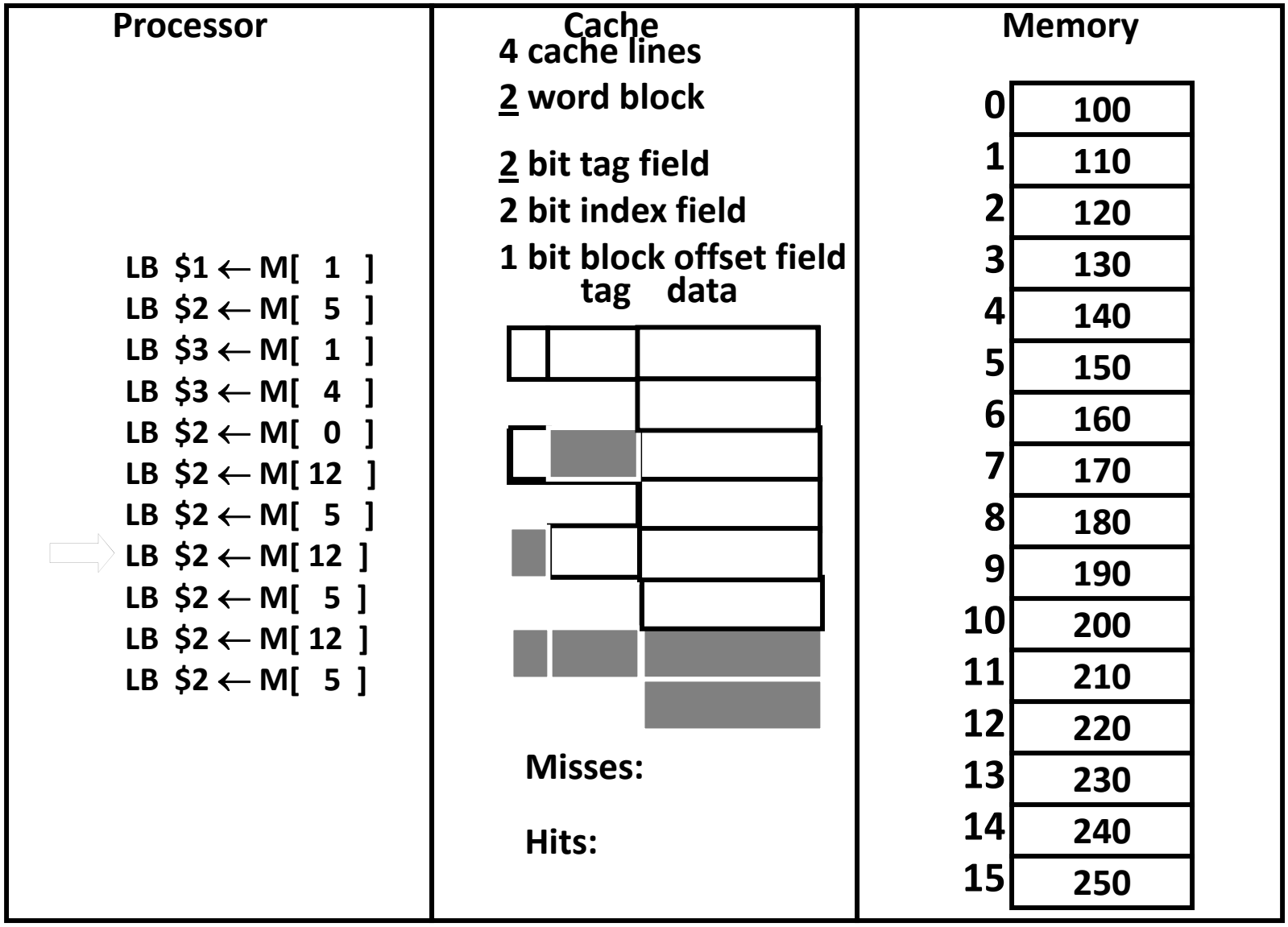
Three common designs

A given data block can be placed…

- … in any cache line → Fully Associative
- … in exactly one cache line → Direct Mapped
- … in a small set of cache lines → Set Associative

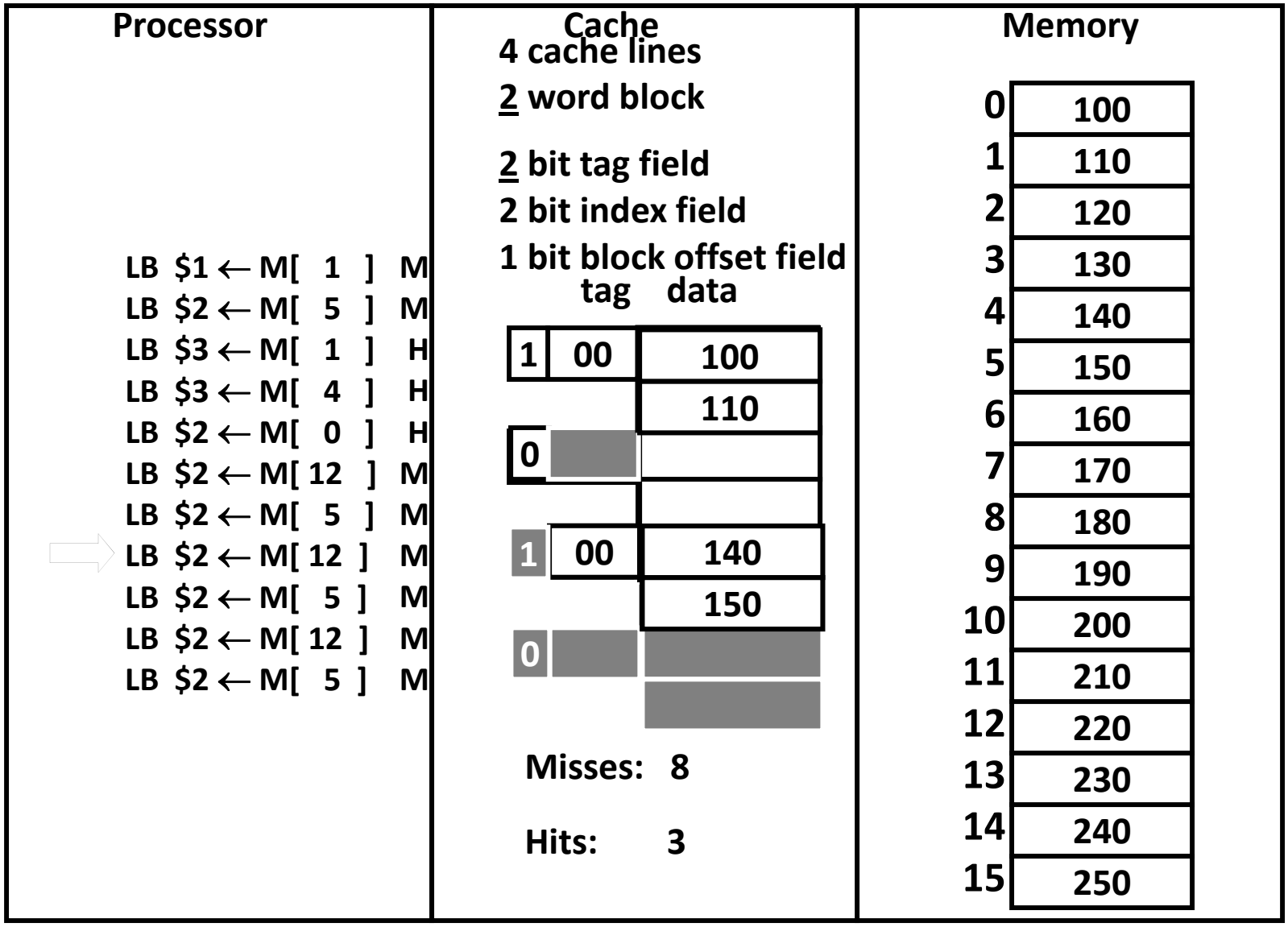
Comparison: Direct Mapped

Using **byte addresses** in this example! Addr Bus = 5 bits



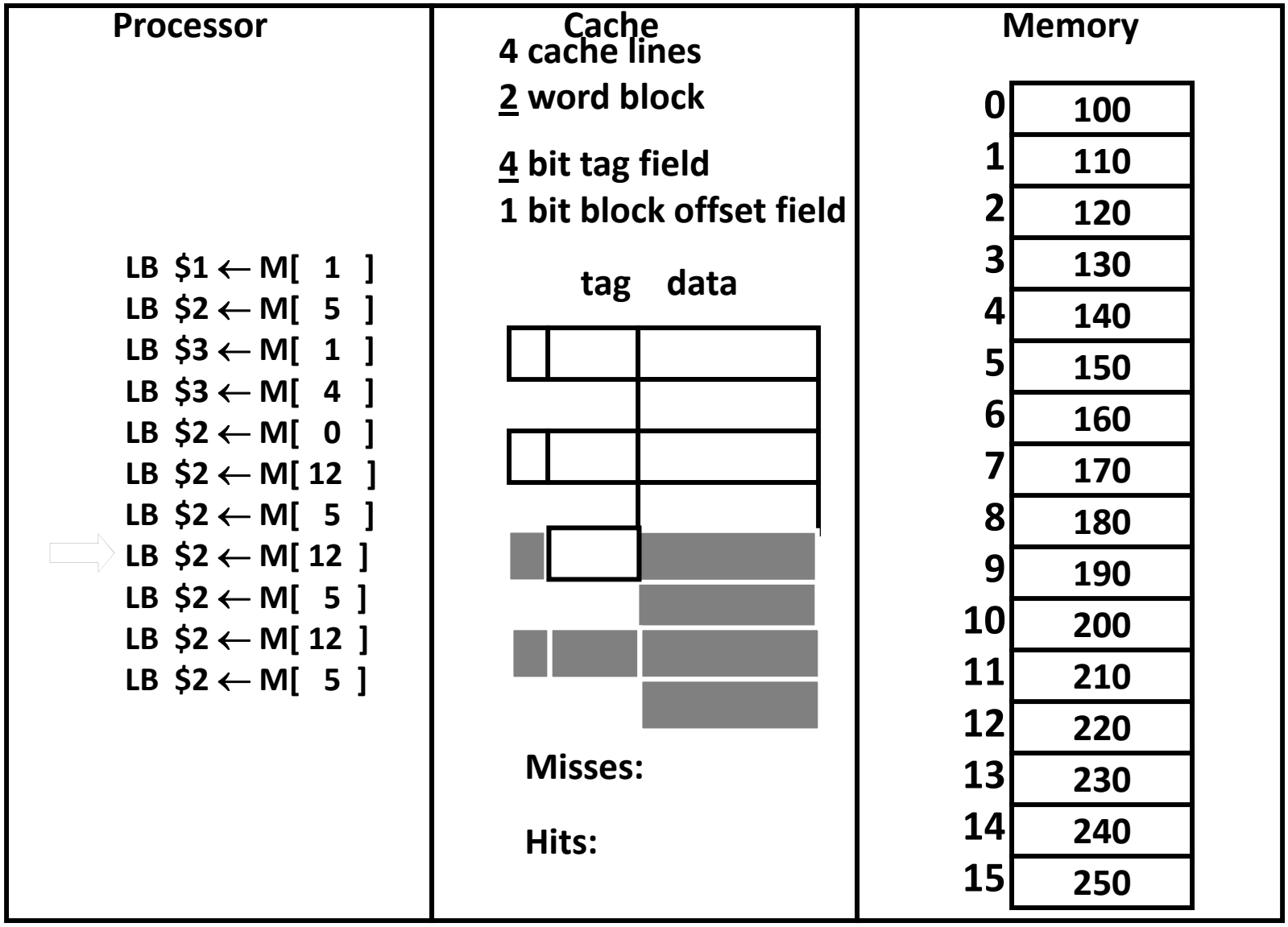
Comparison: Direct Mapped

Using **byte addresses** in this example! Addr Bus = 5 bits



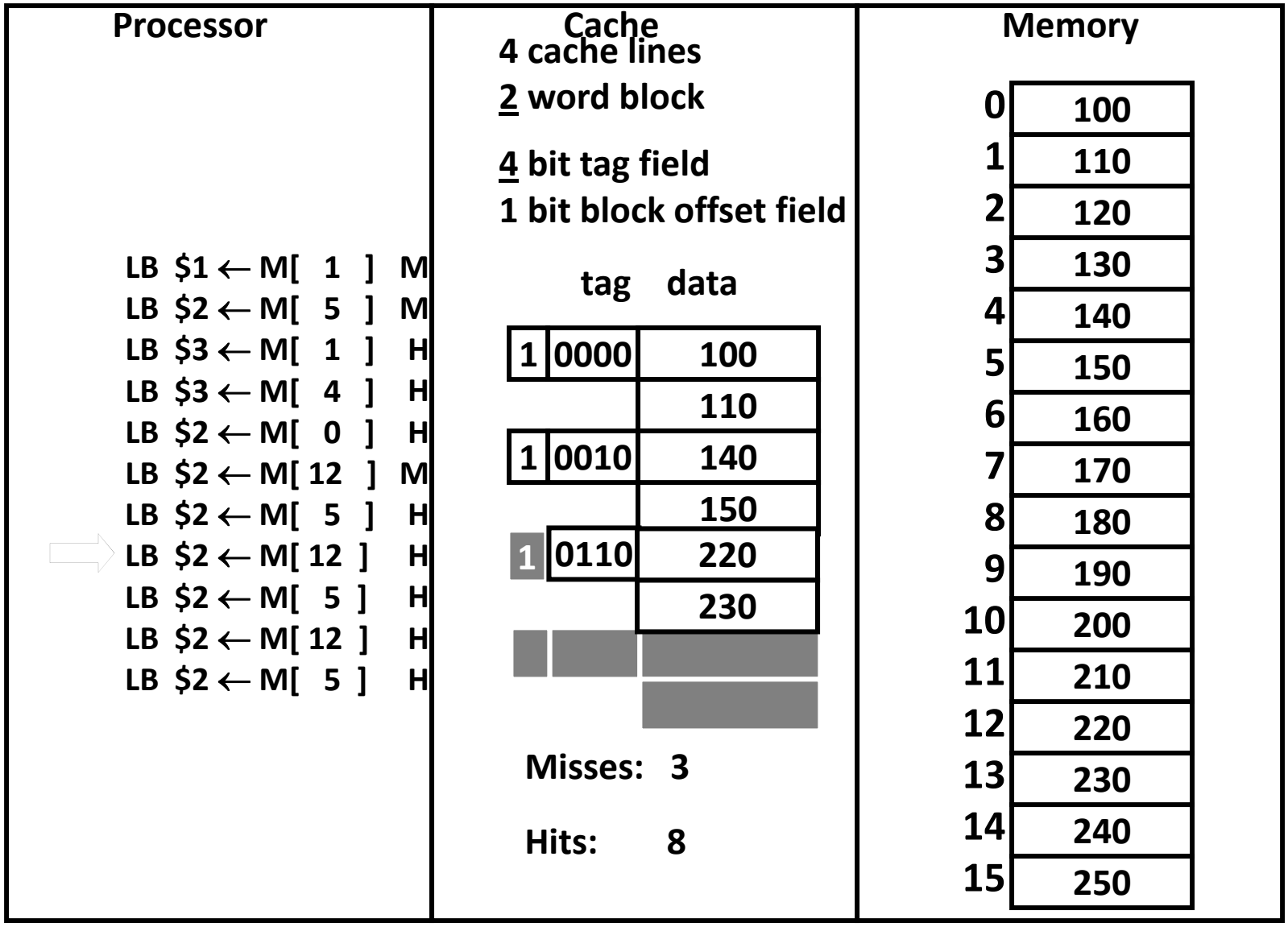
Comparison: Fully Associative

Using **byte addresses** in this example! Addr Bus = 5 bits



Comparison: Fully Associative

Using **byte addresses** in this example! Addr Bus = 5 bits



Comparison: 2 Way Set Assoc

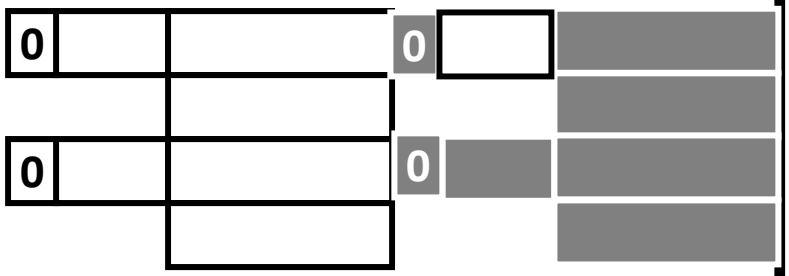
Using **byte addresses** in this example! Addr Bus = 5 bits

Processor

LB $1 ← M[1]
LB $2 ← M[5]
LB $3 ← M[1]
LB $3 ← M[4]
LB $2 ← M[0]
LB $2 ← M[12]
LB $2 ← M[5]
LB $2 ← M[12]
LB $2 ← M[5]
LB $2 ← M[12]
LB $2 ← M[5]

Cache

2 sets
2 word block
3 bit tag field
1 bit set index field
1 bit block offset field



Misses:

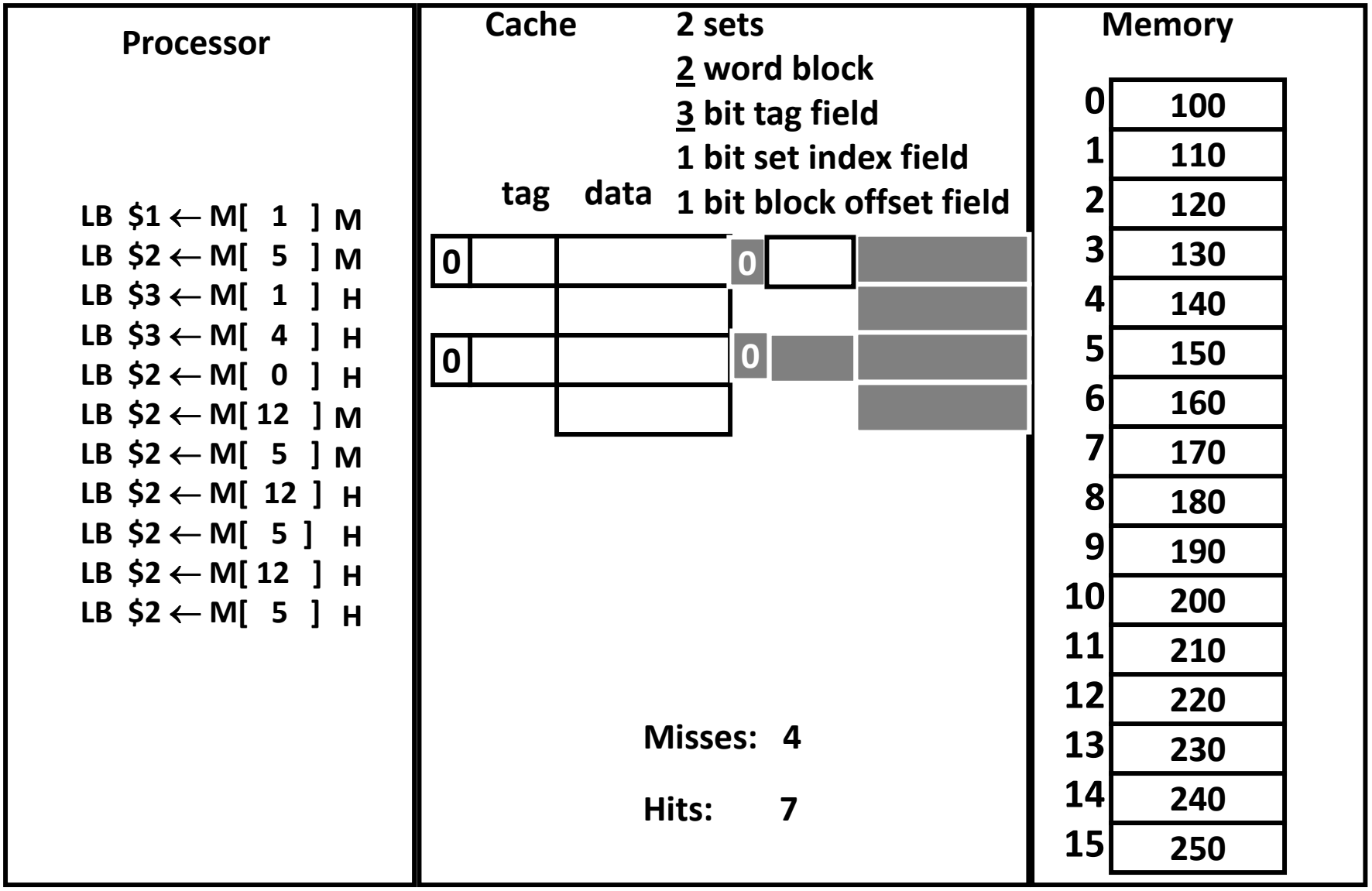
Hits:

Memory

0	100
1	110
2	120
3	130
4	140
5	150
6	160
7	170
8	180
9	190
10	200
11	210
12	220
13	230
14	240
15	250

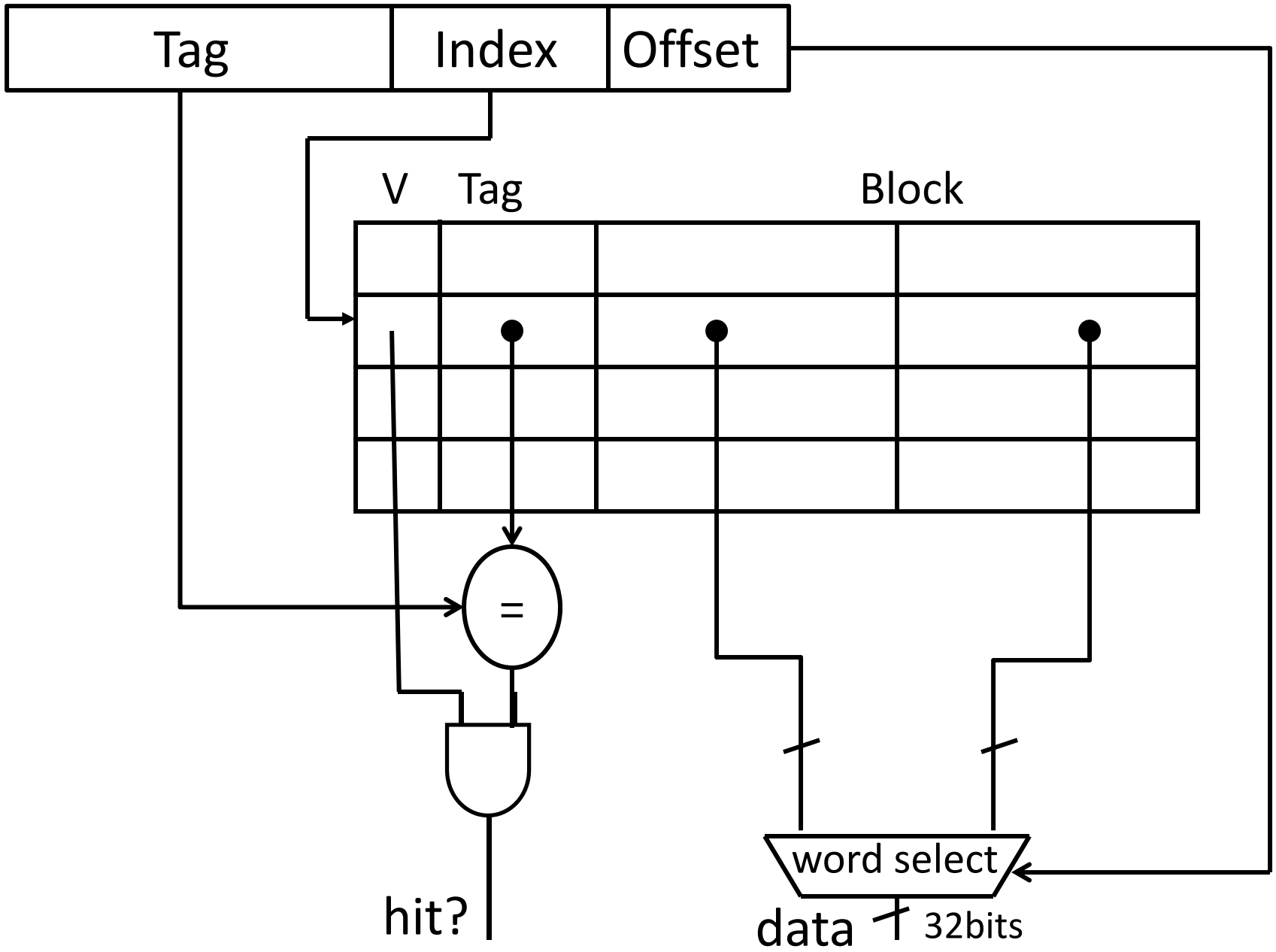
Comparison: 2 Way Set Assoc

Using **byte addresses** in this example! Addr Bus = 5 bits



Cache Size

Direct Mapped Cache (Reading)



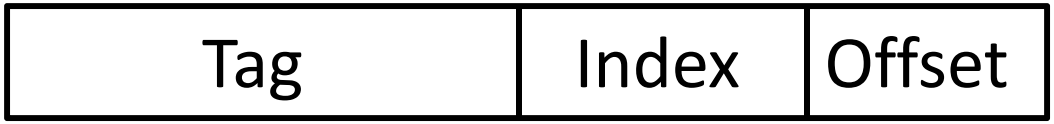
Direct Mapped Cache Size

n bit index, *m* bit offset

Q: How big is cache (data only)?

Q: How much SRAM needed (data + overhead)?

Direct Mapped Cache Size



n bit index, *m* bit offset

Q: How big is cache (data only)?

Q: How much SRAM needed (data + overhead)?

Cache of size 2^n blocks

Block size of 2^m bytes

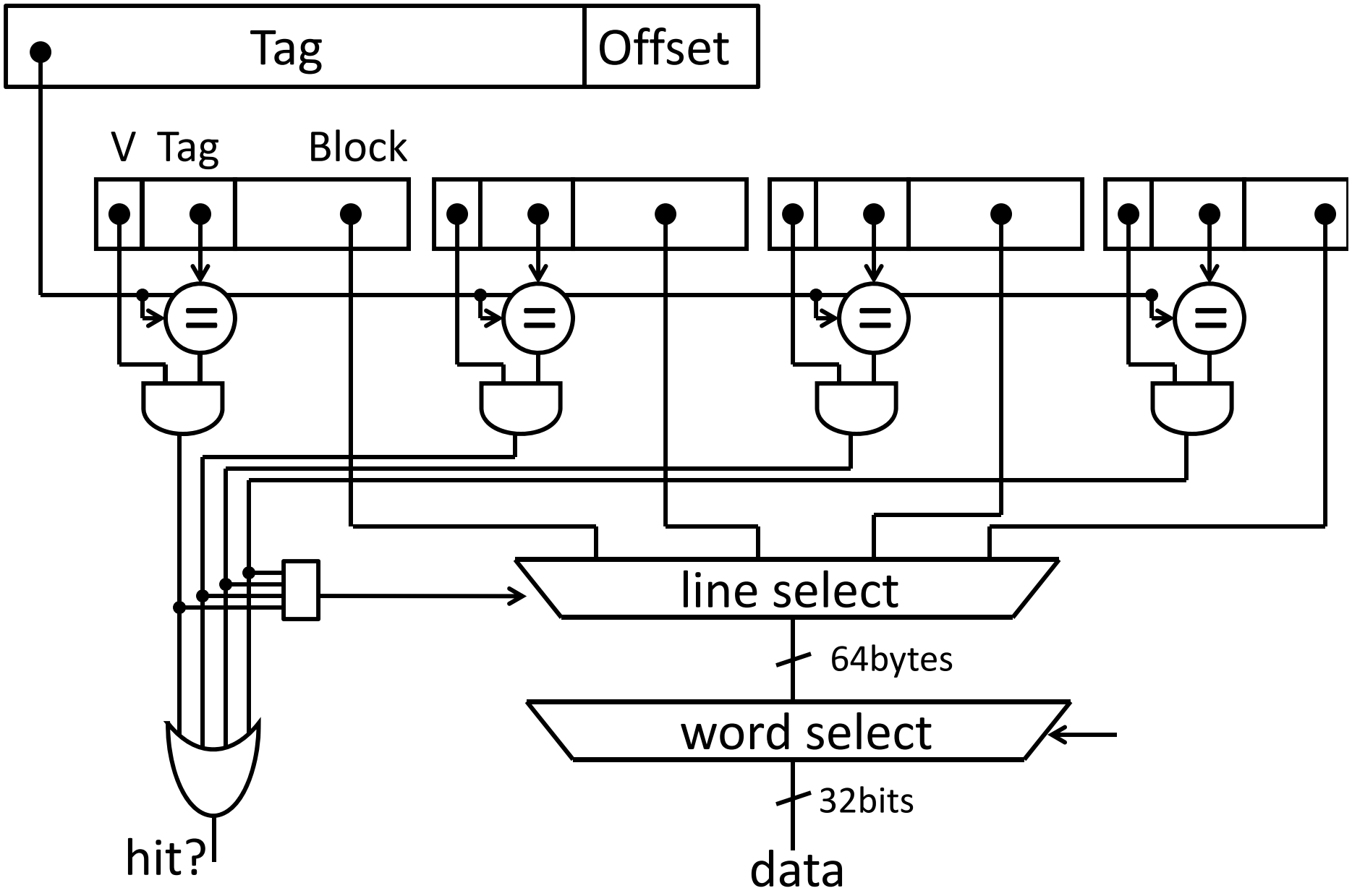
Tag field: 32 – (n + m)

Valid bit: 1

Bits in cache: 2^n x (block size + tag size + valid bit size)

= 2^n (2^m bytes x 8 bits-per-byte + (32-n-m) + 1)

Fully Associative Cache (Reading)



Fully Associative Cache Size

m bit offset , 2^n cache lines

Q: How big is cache (data only)?

Q: How much SRAM needed (data + overhead)?

Fully Associative Cache Size

m bit offset , 2^n cache lines

Q: How big is cache (data only)?

Q: How much SRAM needed (data + overhead)?

Cache of size 2^n blocks

Block size of 2^m bytes

Tag field: 32 – m

Valid bit: 1

Bits in cache: 2^n x (block size + tag size + valid bit size)

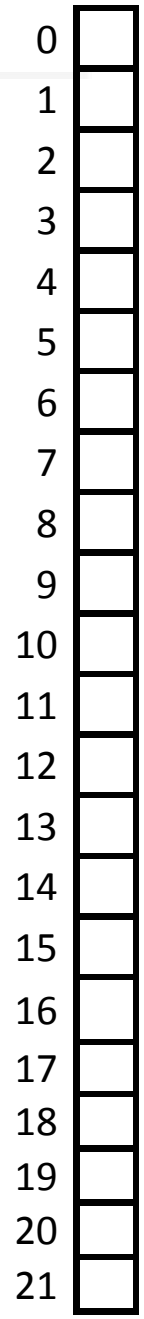
= 2^n (2^m bytes x 8 bits-per-byte + (32-m) + 1)

Fully-associative reduces conflict misses...

… assuming good eviction strategy

Mem access trace: 0, 16, 1, 17, 2, 18, 3, 19, 4, 20, …

Hit rate with four fully-associative 2-byte cache lines?



… but large block size can still reduce hit rate

vector add trace: 0, 100, 200, 1, 101, 201, 2, 202, …

Hit rate with four fully-associative 2-byte cache lines?

With two fully-associative 4-byte cache lines?

Misses

Cache misses: classification

Cold (aka Compulsory)

- The line is being referenced for the first time

Capacity

- The line was evicted because the cache was too small
- i.e. the *working set* of program is larger than the cache

Conflict

- The line was evicted because of another access whose index conflicted

Cache Tradeoffs

Direct Mapped		Fully Associative
+ Smaller	Tag Size	Larger –
+ Less	SRAM Overhead	More –
+ Less	Controller Logic	More –
+ Faster	Speed	Slower –
+ Less	Price	More –
+ Very	Scalability	Not Very –
– Lots	# of conflict misses	Zero +
– Low	Hit rate	High +
– Common	Pathological Cases?	?

Summary

Caching assumptions

- small working set: 90/10 rule
- can predict future: spatial & temporal locality

Benefits

- big & fast memory built from (big & slow) + (small & fast)

Tradeoffs:

associativity, line size, hit cost, miss penalty, hit rate

- Fully Associative → higher hit cost, higher hit rate
- Larger block size → lower hit cost, higher miss penalty

Next up: other designs; writing to caches