

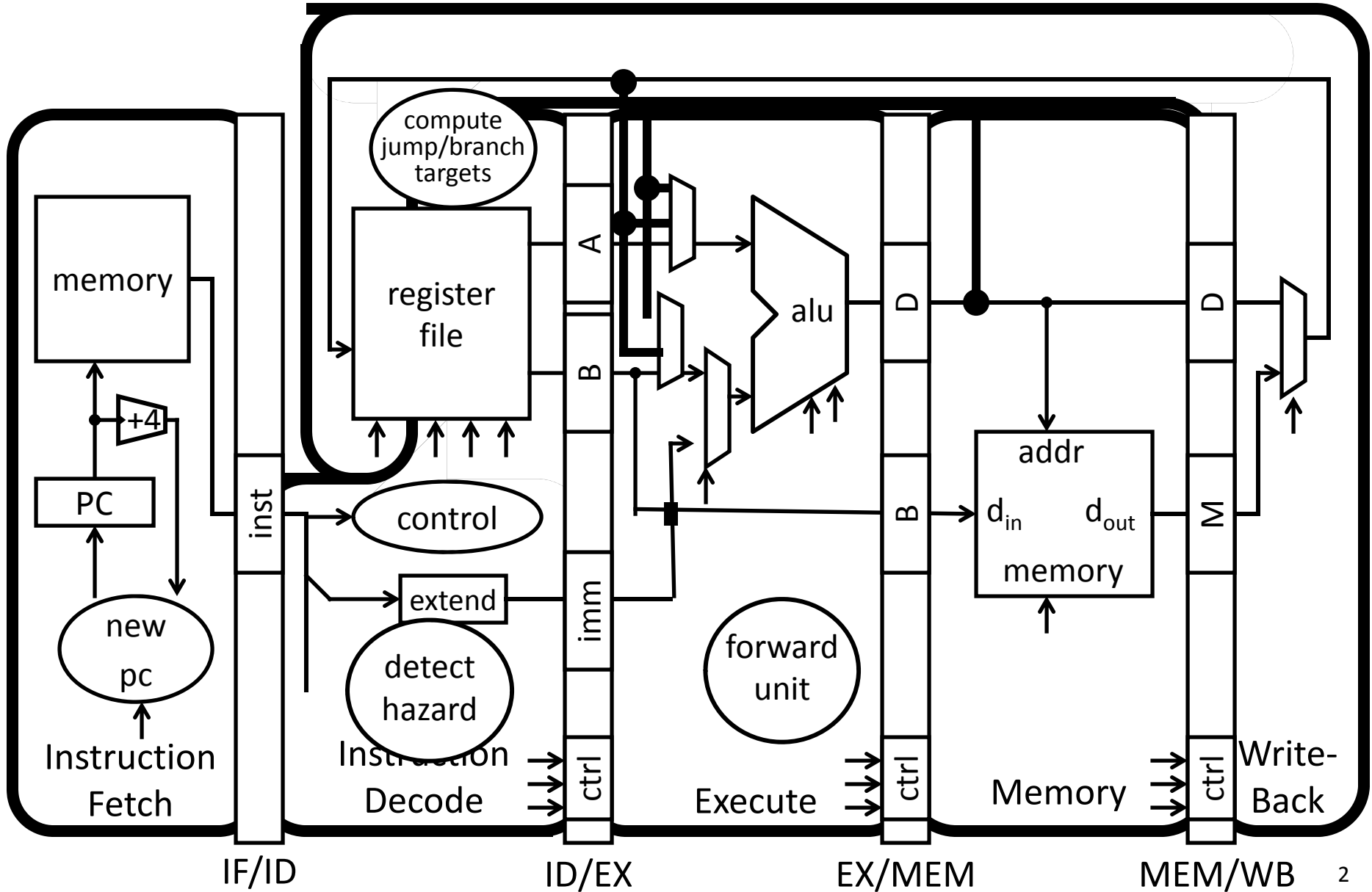
Caches

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



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

16 words

$$AMAT = \% \text{ hit} \times \text{hit time} + \% \text{ miss} \times \text{miss time}$$

$$\text{Cost hit} = 5 \text{ cycles}$$

$$\text{Cost miss} = 2 + 50 + 16 * 3$$

90%

Performance depends on:

Access time for hit, miss penalty, hit rate

$$= 5 + 50 + 16 * 3$$

$$4.5 + 10 = 14.5 \text{ cycles}$$

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

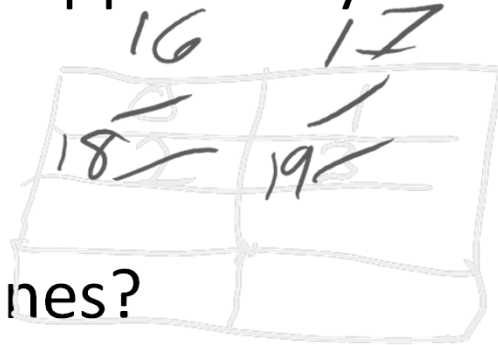
- Buy more SRAM
- Use a more flexible cache design

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?

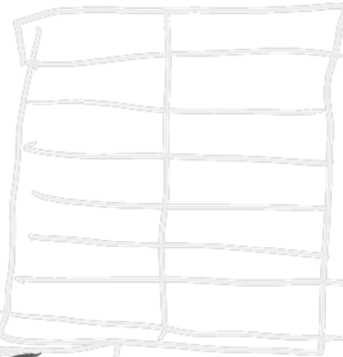
hit rate = 0%



16 = 010000
Index

With eight 2-byte cache lines?

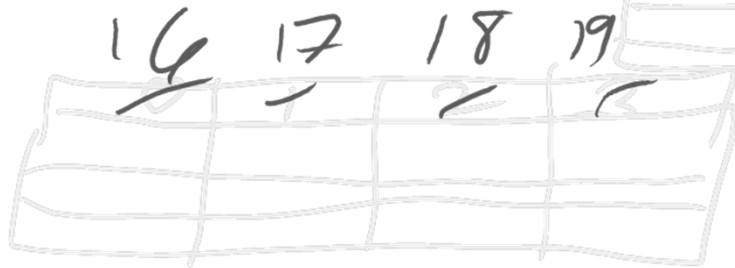
0%



16 = 010000
Index

With four 4-byte cache lines?

0%



16 = 010000
Index off

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

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

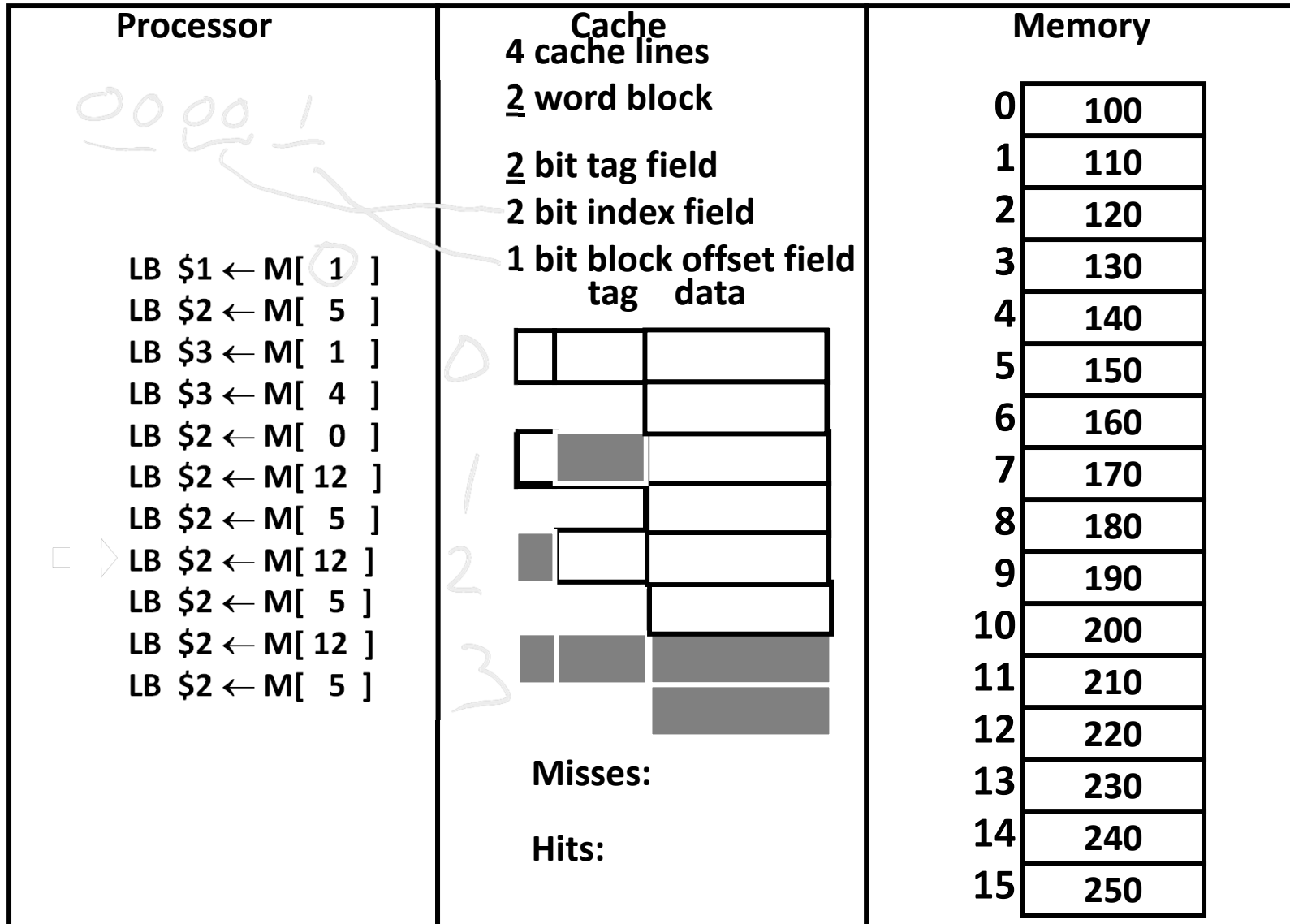
Three common designs

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

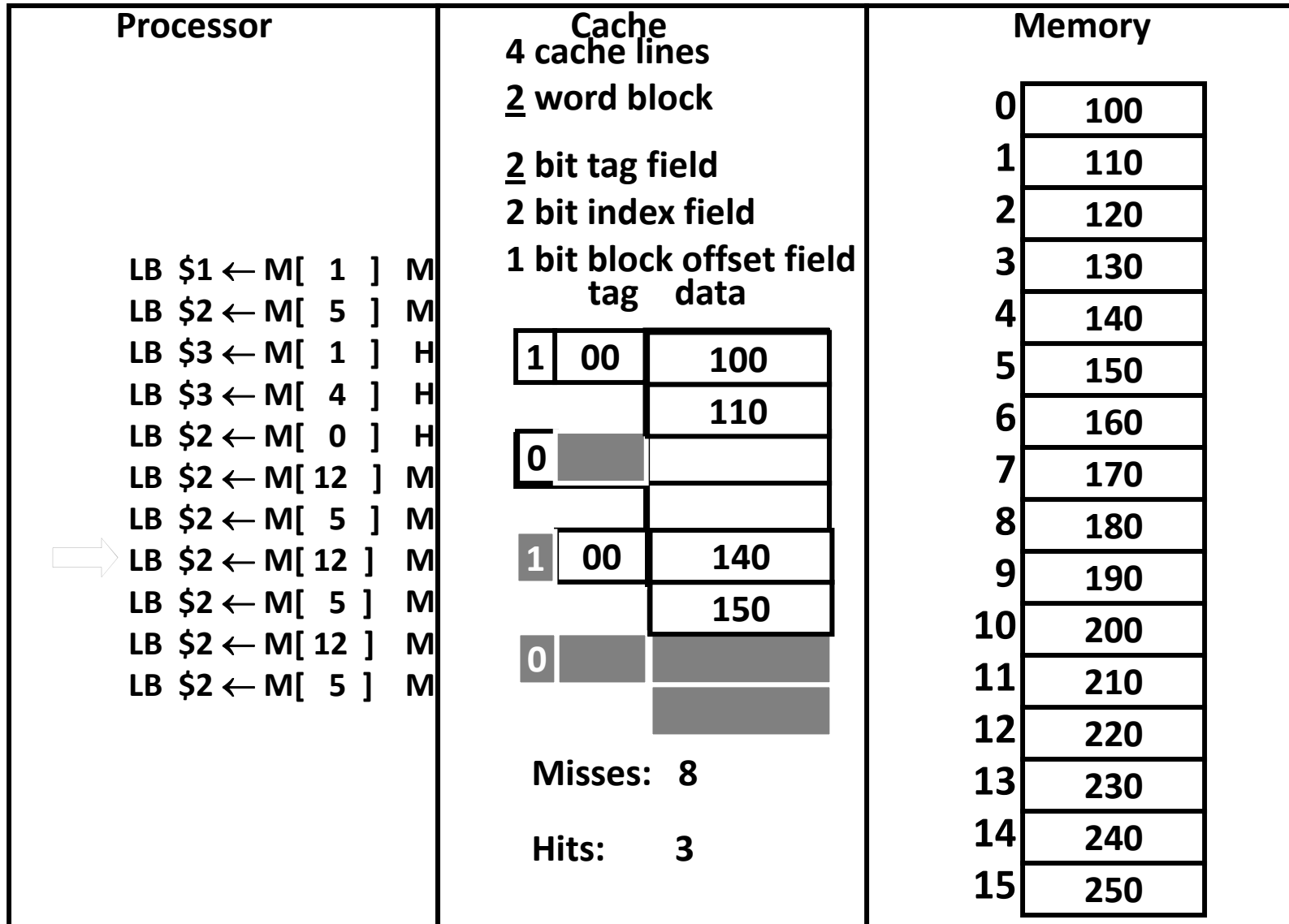
Comparison: Direct Mapped

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



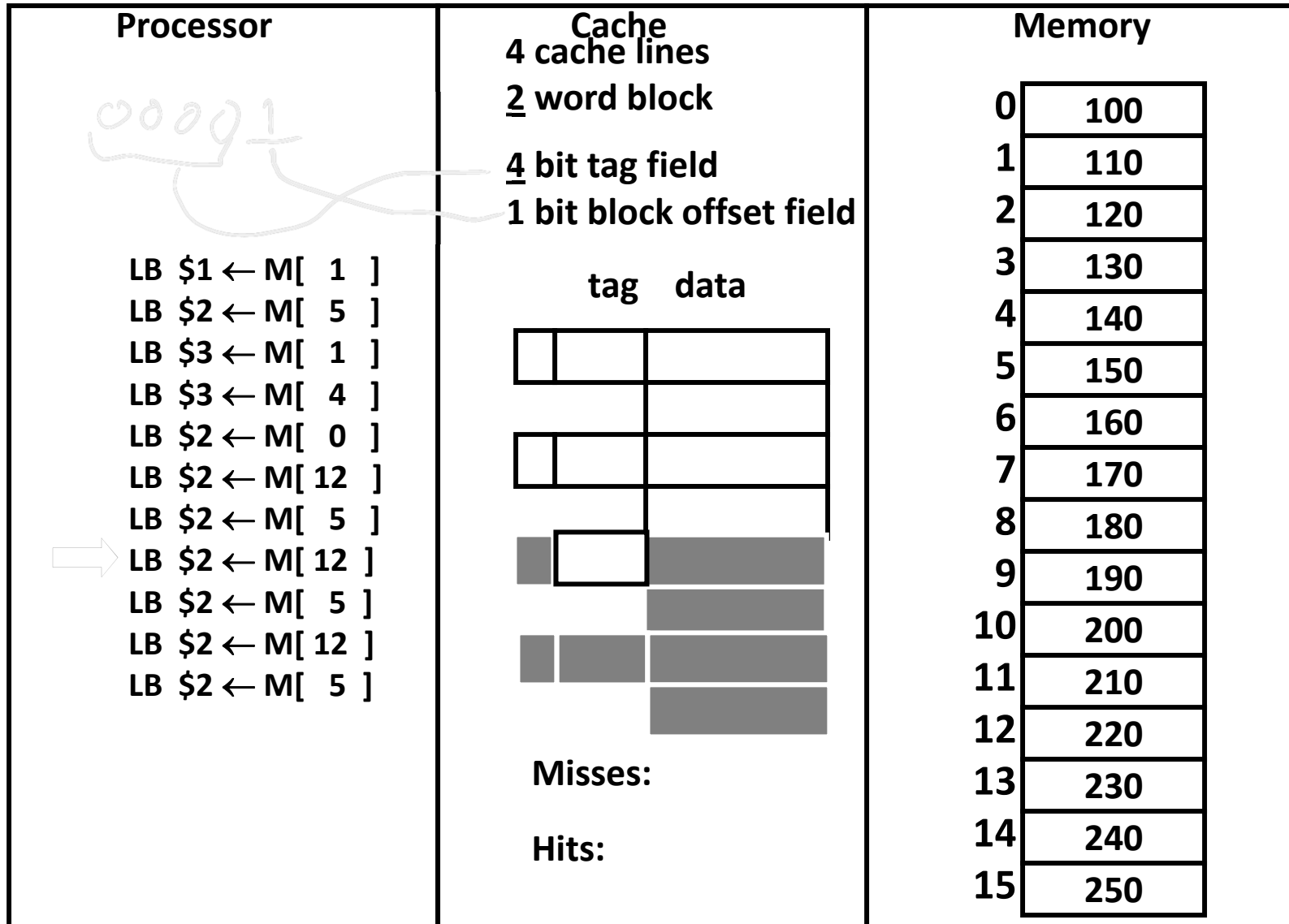
Comparison: Direct Mapped

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



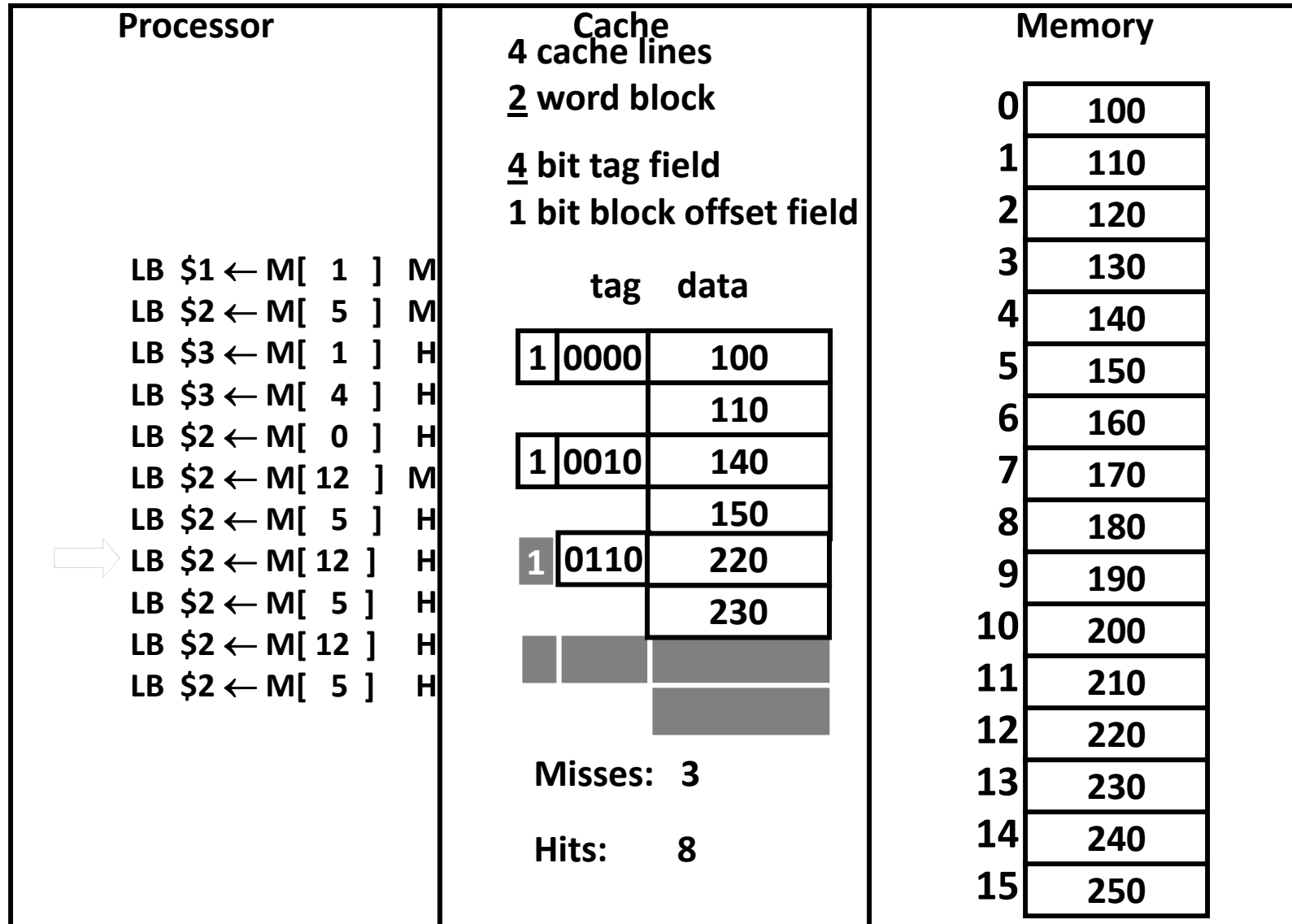
Comparison: Fully Associative

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



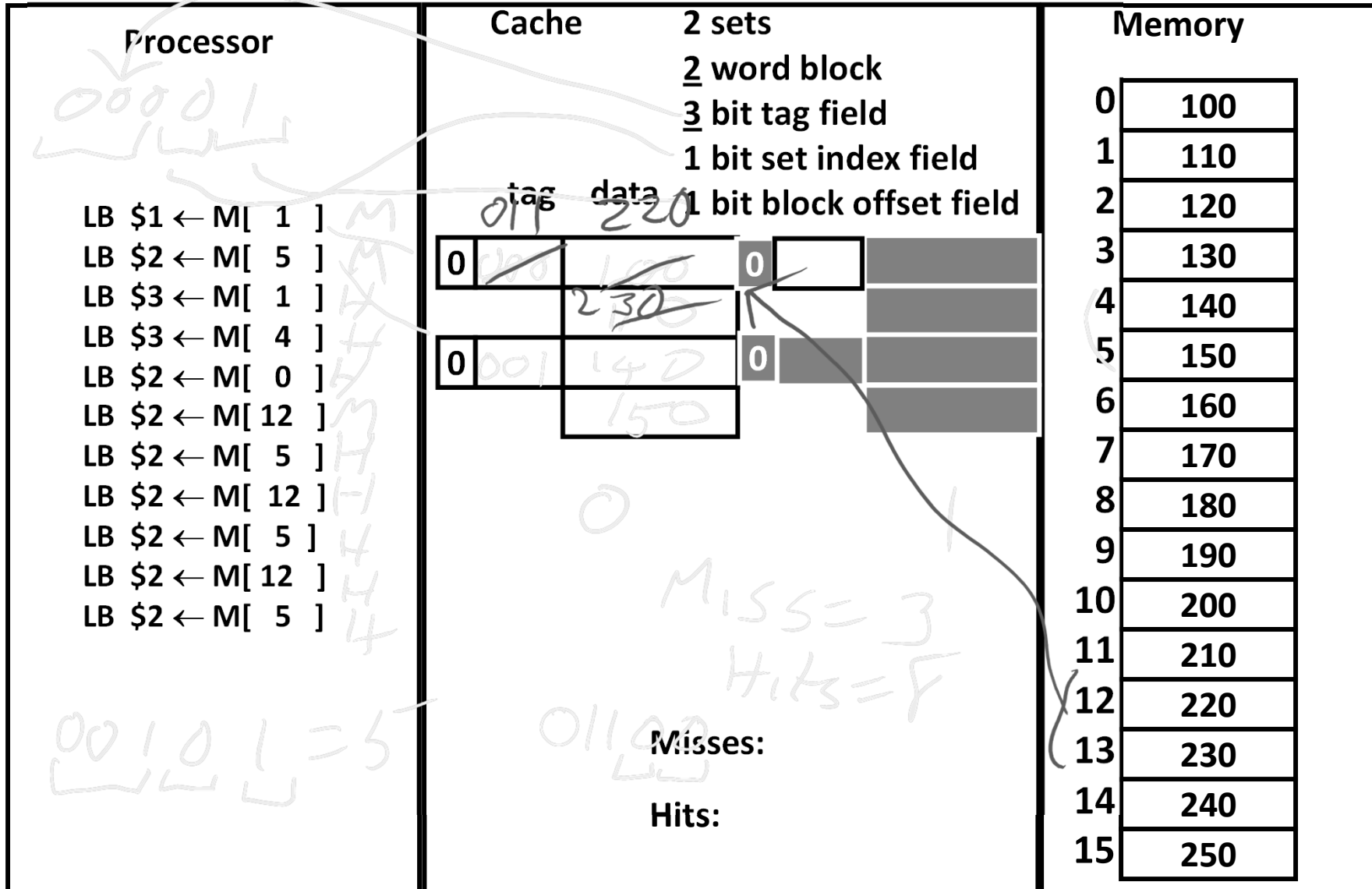
Comparison: Fully Associative

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



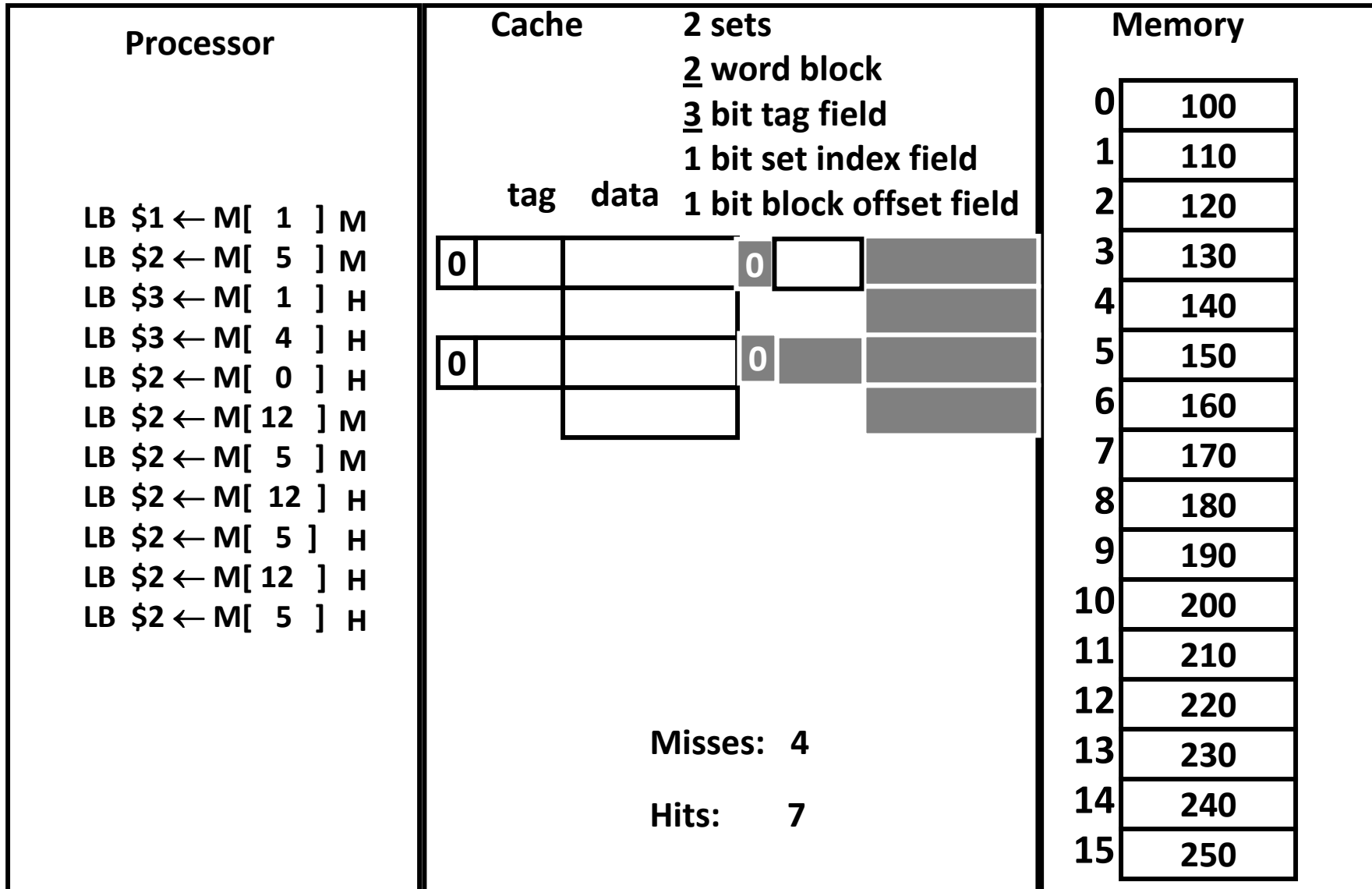
Comparison: 2 Way Set Assoc

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



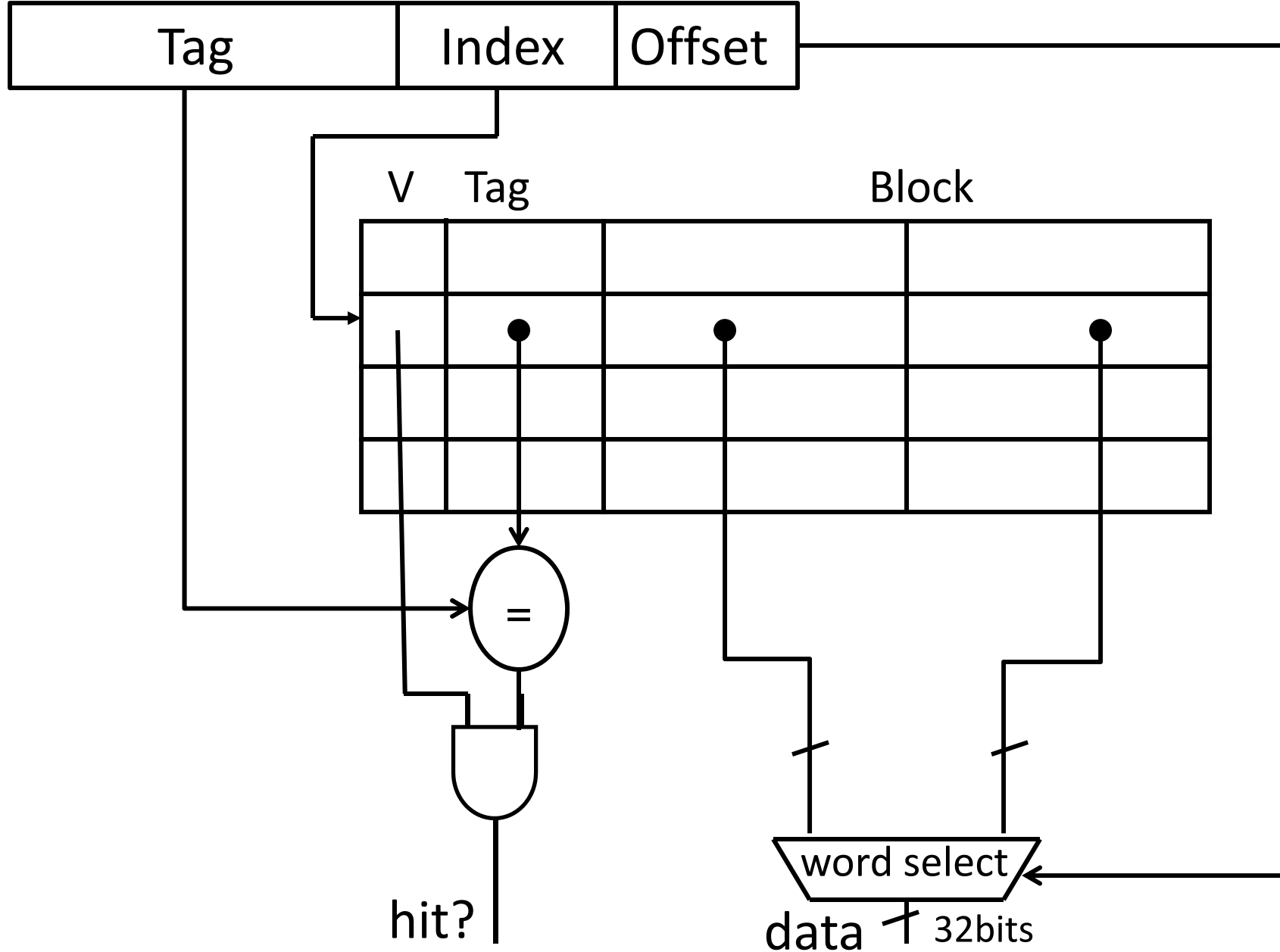
Comparison: 2 Way Set Assoc

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



Cache Size

Direct Mapped Cache (Reading)



Direct Mapped Cache Size

Tag	Index	Offset
-----	-------	--------

byte
addressable

n bit index, m bit offset

Q: How big is cache (data only)?

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

2^m bytes per block

2^n blocks

$$2^n \cdot 2^m = 2^{n+m}$$

overhead

$$\text{tag} = 32 - n - m$$

$$\text{valid} = 1$$

$$2^n \cdot (\text{tag} + \text{valid})$$

Direct Mapped Cache Size

Tag	Index	Offset
-----	-------	--------

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

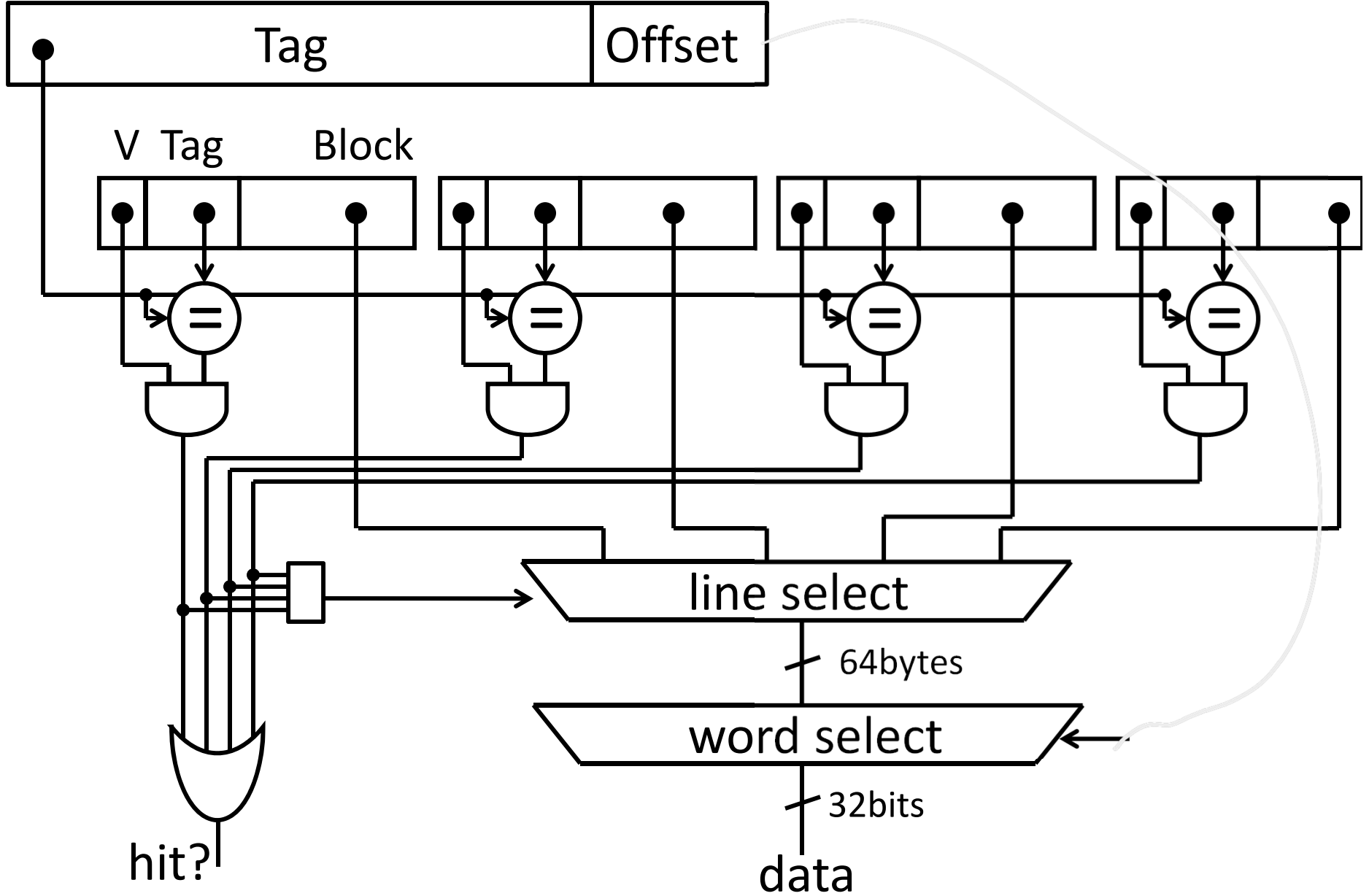
Block size of 2^m bytes

Tag field: $32 - (n + m)$

Valid bit: 1

Bits in cache: $2^n \times (\text{block size} + \text{tag size} + \text{valid bit size})$
 $= 2^n (2^m \text{ bytes} \times 8 \text{ bits-per-byte} + (32-n-m) + 1)$

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)?

$$\# \text{ cache lines} \times \text{block size} \\ 2^n \times 2^m \text{ bytes} = 2^{n+m}$$

overhead
tag = $32 - m$
valid = 1

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: $2^n \times (\text{block size} + \text{tag size} + \text{valid bit size})$
 $= 2^n (2^m \text{ bytes} \times 8 \text{ bits-per-byte} + (32-m) + 1)$

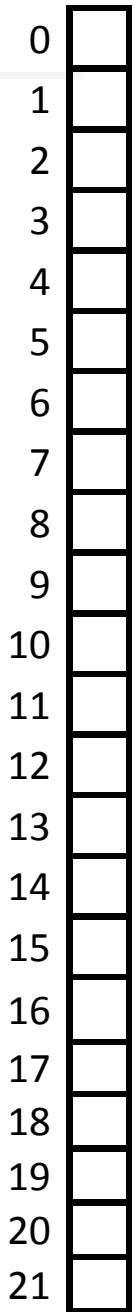
Fully-associative reduces conflict misses...

... 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?

hit rate 50%



... 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?

50%

0	1
100	101
200	201

With two fully-associative 4-byte cache lines?

0%

200 201		202	203
0	1	2	3
100	101	102	103
0	1	2	3

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?

?

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

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