CSCI 210: Computer Architecture
Lecture 36: Associative Caches

Stephen Checkoway

Oberlin College

May 20, 2022

Slides from Cynthia Taylor

Announcements

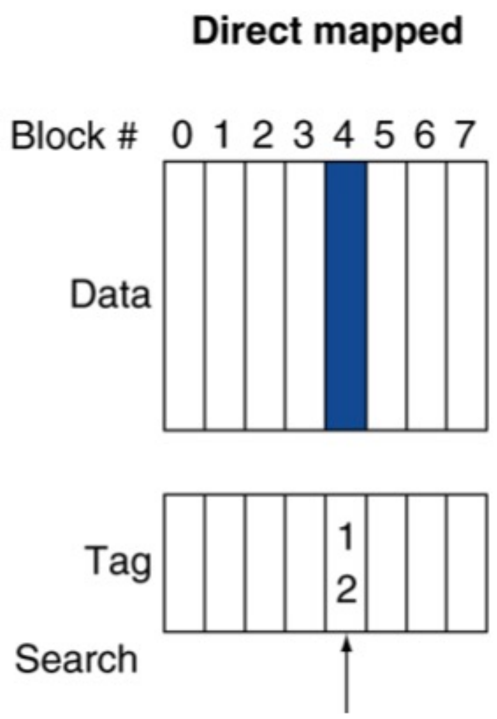
- Problem Set 12 due Thursday
- Cache Lab (final project) due on the day of the final exam
- Course evals now available!
  - Extra credit for everyone if more than 90% of the class fills them out
- Office Hours today 13:30 – 14:30

ASSOCIATIVE CACHES

Direct-mapped Cache

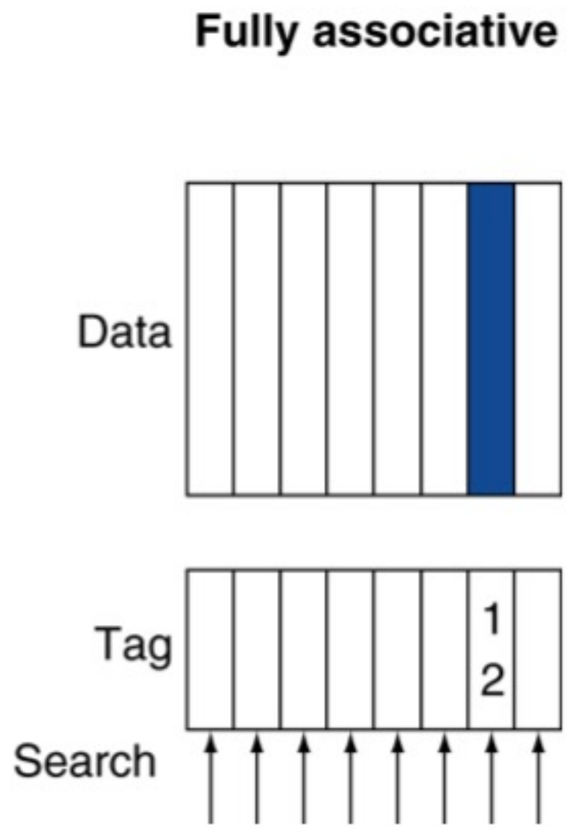
- Each block goes into **1** spot
- Only search one entry
- Associativity = 1

- What if we allow blocks to go into more than one spot?



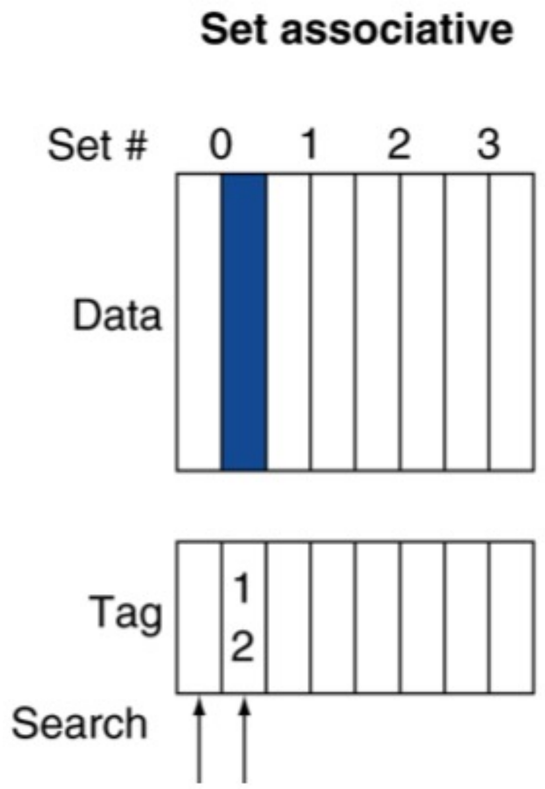
Fully-associative Cache

- Allow a given block to go in any cache entry
- Requires all entries to be searched at once
- Comparator per entry (expensive)

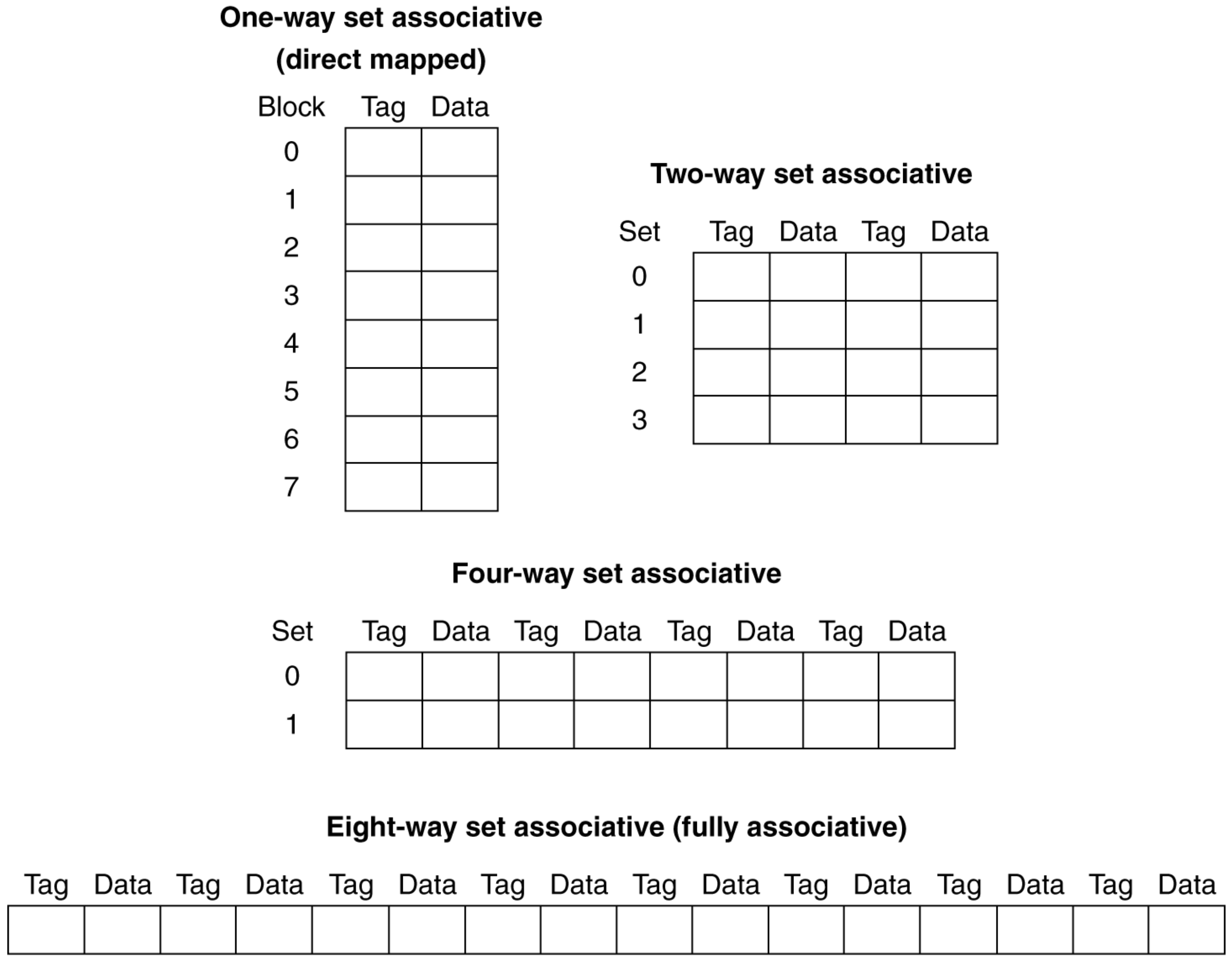


n-way Set-associative Cache

- Each set contains *n* entries
- Block number 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 8-entry cache



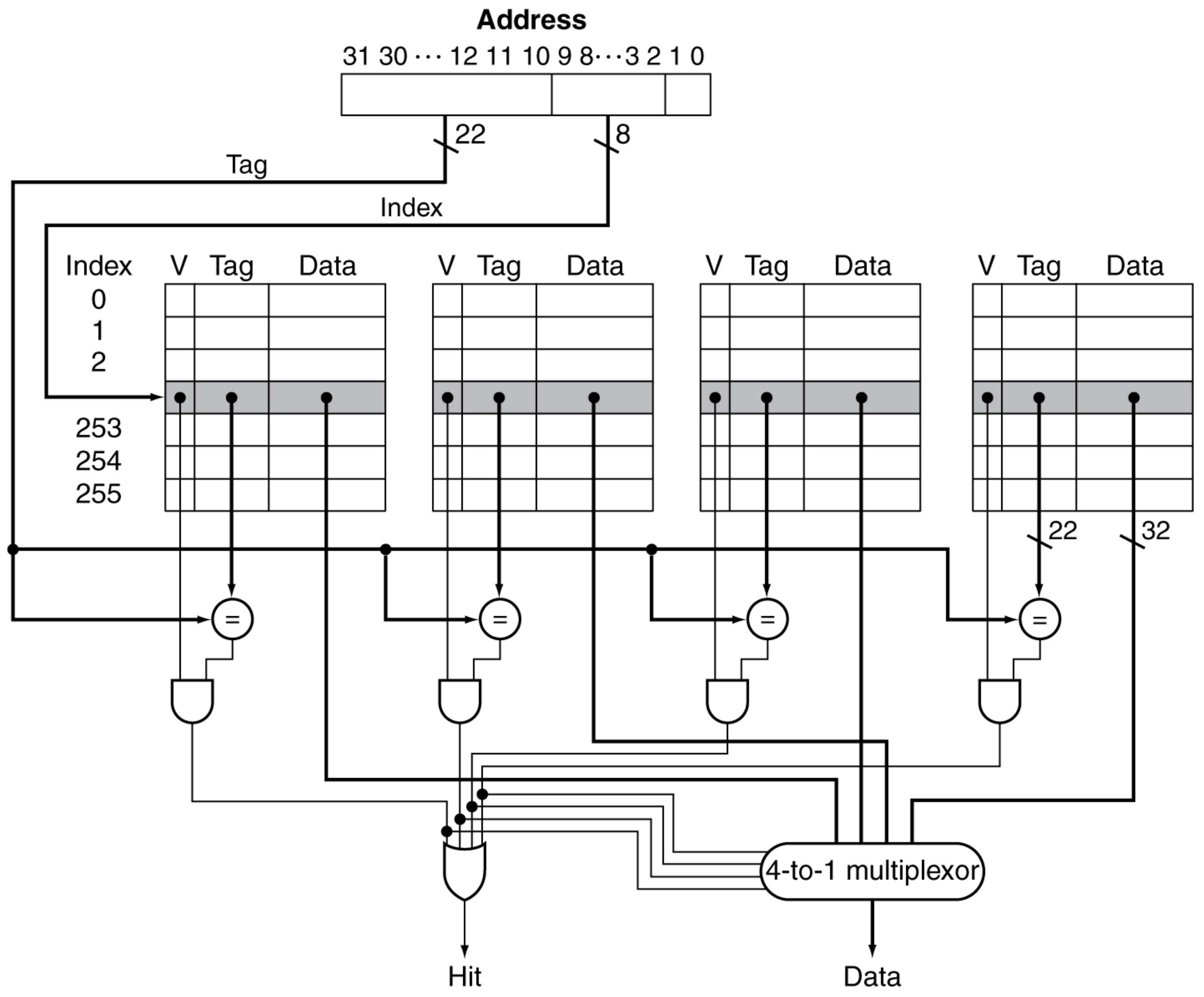
Memory addresses, block addresses, offsets

0	0	0	1	0	1	0	1	1	1	0	0	1	0	0	1	1	0	1	0	1	1	0	0	1	0	1	0	0	0	1	1

- Block size of 32 bytes (not bits!)
- 16-block, 2-way set associative cache
- Each address
  - A (32 – 5)-bit block address (in purple and blue)
  - A 5-bit offset into the block (in green)
- Block address can be divided into
  - A (32 – 3 – 5)-bit **tag** (purple)
  - A 3-bit cache **index** (blue)

V	Tag	Data	V	Tag	Data
0			0		
0			0		
0			1	3F2084	…
0			0		
0			0		
1	15C9AC	…	1	28477D	…
0			0		
0			0		

Set Associative Cache Organization



Given a 256-entry, 8-way set associative cache with a block size of 64 bytes, how many bits are in the tag, index, and offset?

	Tag bits	Index bits	Offset bits
A	32 – 5 – 6 = 21	5	6
B	32 – 3 – 5 = 24	3	5
C	32 – 8 – 6 = 18	8	6
D	32 – 6 – 5 = 21	6	5
E	32 – 6 – 3 = 23	6	3

Given a 256-entry, fully associative cache with a block size of 64 bytes, how many bits are in the tag, index, and offset?

	Tag bits	Index bits	Offset bits
A	32 – 5 – 6 = 21	1	6
B	32 – 3 – 5 = 24	3	5
C	32 – 8 – 6 = 18	8	6
D	32 – 6 – 5 = 21	6	5
E	32 – 0 – 6 = 26	0	6

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: 0, 8, 0, 6, 8

- 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					

- Fully associative

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

Replacement Policy

- Direct mapped: no choice

- Set associative
  - Prefer non-valid entry, if there is one
  - Otherwise, choose among entries in the set
  - Goal: Choose an entry we will not use in the future

Replacement Policy

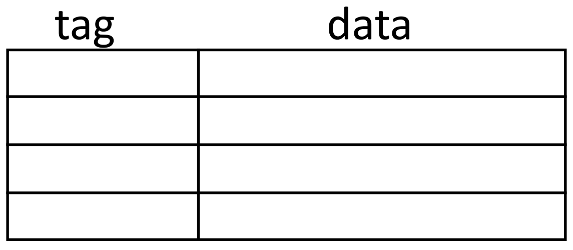
- Least-recently used (LRU)
  - Choose the one unused for the longest time
    - Simple for 2-way, manageable for 4-way, too hard beyond that
- Random
  - Gives approximately the same performance as LRU for high associativity

Three types of cache misses

- Compulsory (or cold-start) misses
  - first access to the data.
- Capacity misses
  - we missed only because the cache isn't big enough.
- Conflict misses
  - we missed because the data maps to the same index as other data that forced it out of the cache.

<u>block address of misses</u>

4
8
12
4
8
20
4
8
20
24
12
8
4



DM cache

Cache miss example (from StackOverflow)

32 kB direct-mapped cache

1. You repeatedly iterate over a 128 kB array
   - All misses but the first access to each line are capacity misses because the array does not fit in cache; the first are compulsory misses
2. You iterate over two 8 kB arrays that map to the same cache indices
   - These are conflict misses because if you changed the locations of the arrays to be consecutive, then both would fit in the cache

https://stackoverflow.com/a/33336918

Cache Miss Type

Suppose you experience a cache miss on a block (let's call it block A). You have accessed block A in the past. There have been precisely 1027 different blocks accessed between your last access to block A and your current miss. Your block size is 32-bytes and you have a 64 kB cache (recall a kB = 1024 bytes). What kind of miss was this?

Selection	Cache Miss
A	Compulsory
B	Capacity
C	Conflict
D	Both Capacity and Conflict
E	None of the above

CACHE SIMULATOR PROJECT

Cache Simulator

- Take in a trace of load/stores from a real program
- Simulate running the program on a given cache
- Calculate how well a given cache would perform for that trace

Cache Parameters

- Always: Write-allocate, write-back, LRU replacement

- Change:
  - Cache size
  - Block size
  - Associativity
  - Miss penalty

Address Trace

# Load/Store Address InstructionCount

```
# 0 7fffed80 1
# 0 10010000 10
# 0 10010060 3
# 1 10010030 4
# 0 10010004 6
# 0 10010064 3
# 1 10010034 4
```

L/S: 0 for load, 1 for store

Simulation Results

```
Simulation results:
    execution time                    52268708 cycles
    instructions                       5136716
    memory accesses                    1957764
    overall miss rate                     0.79
    load miss rate                        0.88
    CPI                                  10.18
    average memory access time           24.07 cycles
    dirty evictions                     225876
    load_misses                        1525974
    store_misses                         30034
    load_hits                           205909
    store_hits                          195847
```

What do you need to do?

- Create data structures that emulate a cache
- For each instruction, find where it would go in the cache, check if it’s already there
- Calculate number of miss penalty cycles, load misses, store misses, instructions, etc.

Reading

- Next lecture: More Caches!
  - Section 6.4

- Problem Set 12 due Friday

- Cache lab