CS 211: Intro to Computer Architecture

14.2: Caching 3 & Course Wrap-Up

Minesh Patel

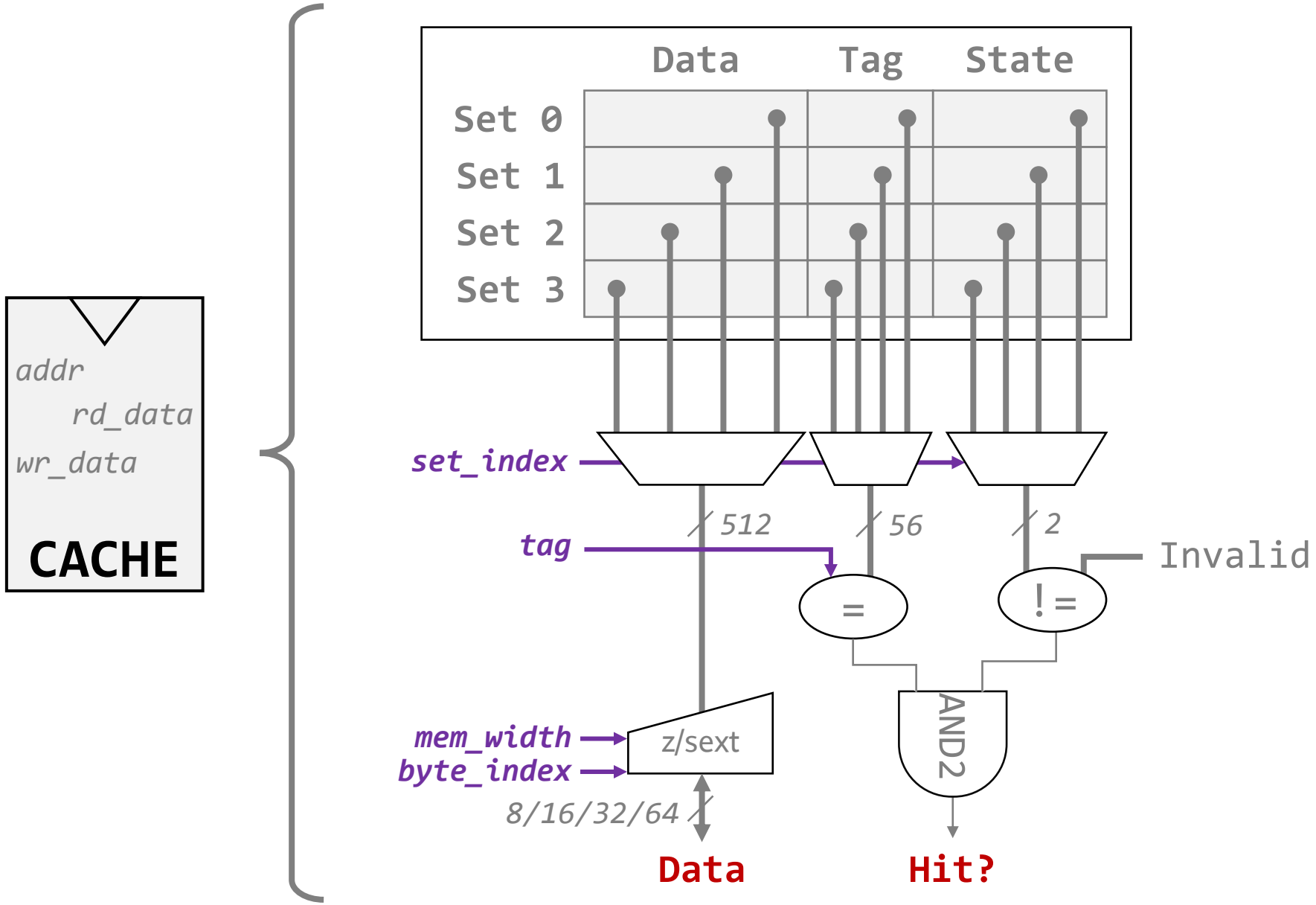
Spring 2025 – Thursday 1 May

Announcements

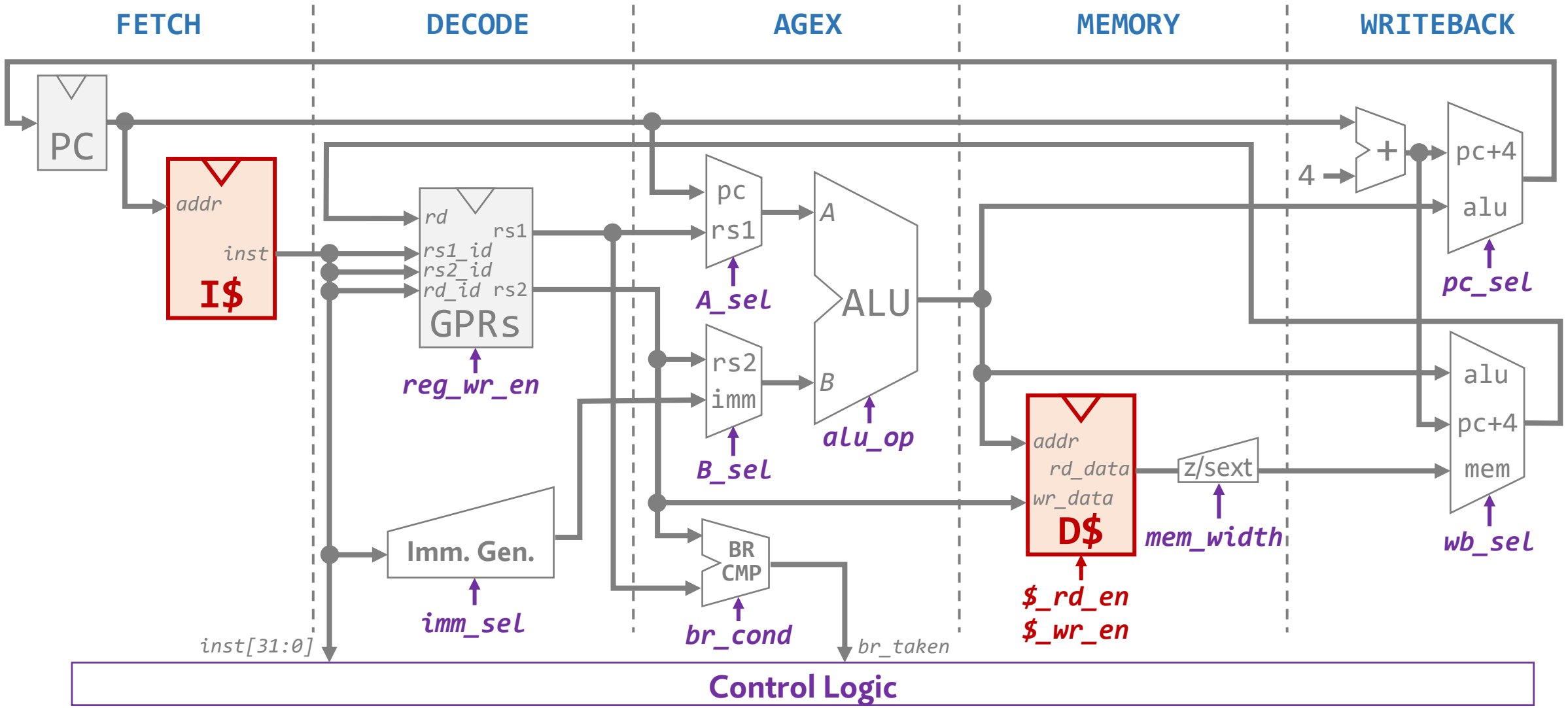
- Ongoing
  - **WA10:** due **Friday (May 2) @ 11:59 pm**
  - **PA5:** due **Monday (May 5) @ 11:59 pm**
    - Autograder: Today!

- Upcoming
  - **Final Exam:** May 14th
    - Pending: List of topics
    - Pending: Practice questions for this half of the semester

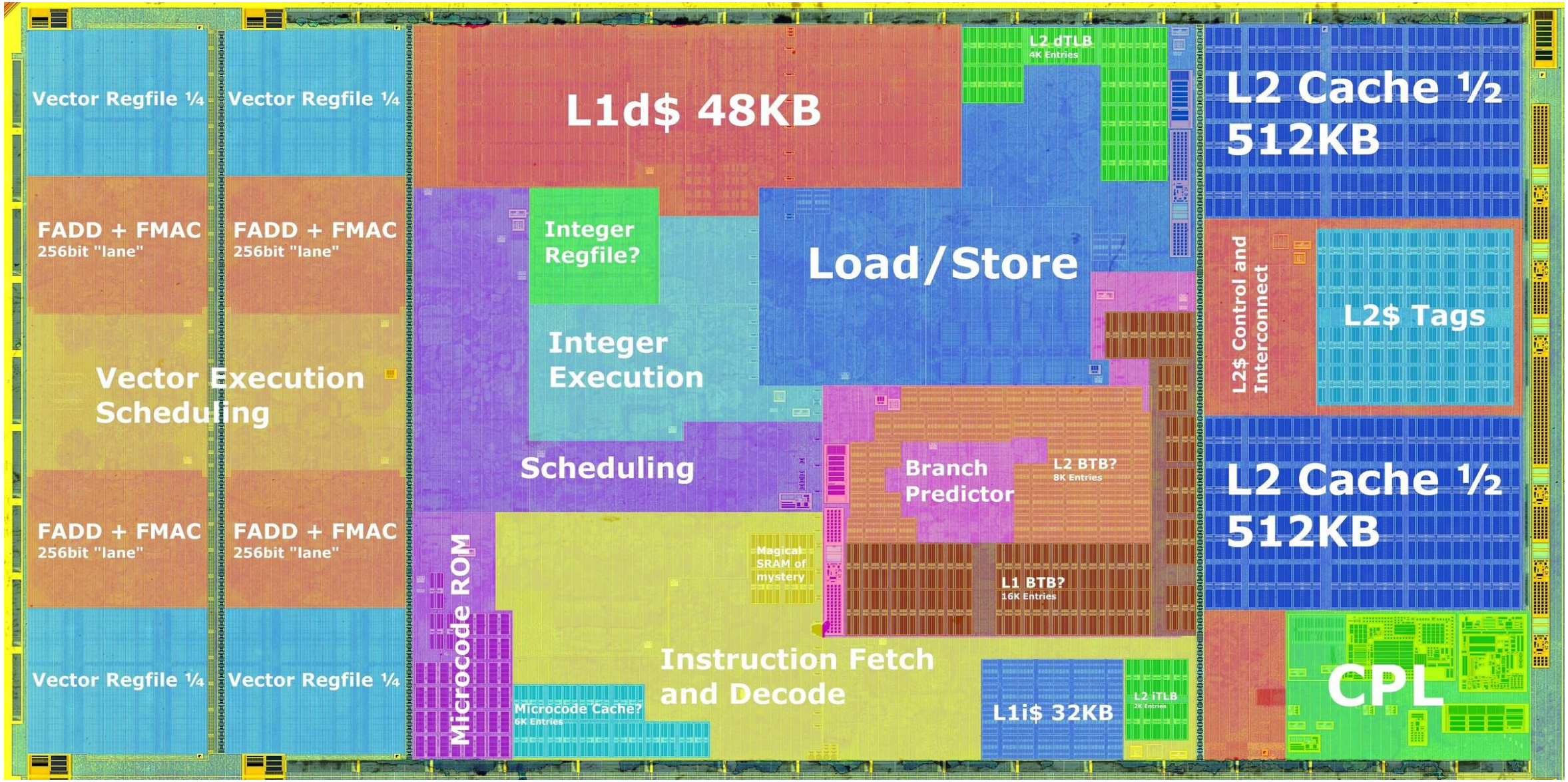
Recall: Caches Are Datapath Elements



Recall: Instruction Cache (I$) and Data Cache (D$)



AMD Zen 5 (Ryzen 9000) Die Shot



https://tpucdn.com/img/AFnVIoGFWSCE6YXO_thm.jpg

Agenda

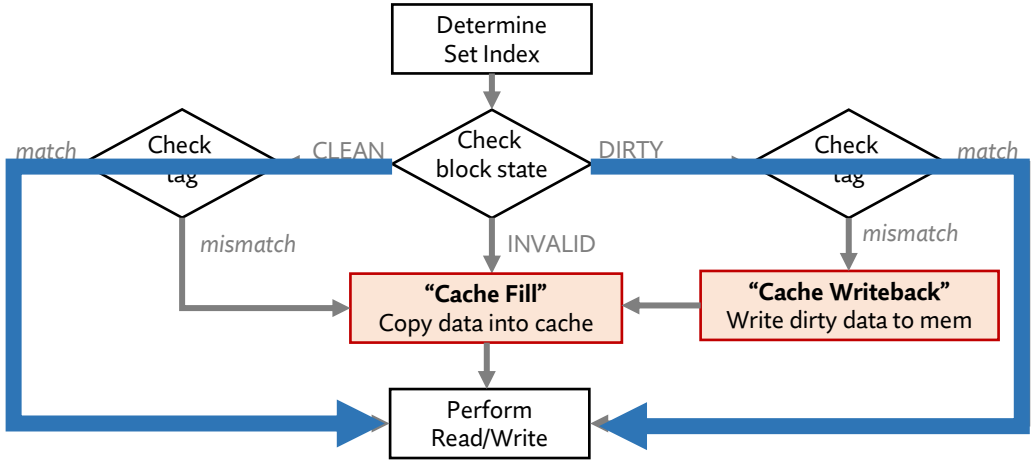
- **Simulating Cache Behavior**

- Types of Cache Misses
  - Cold Misses
  - Set Associative Caches

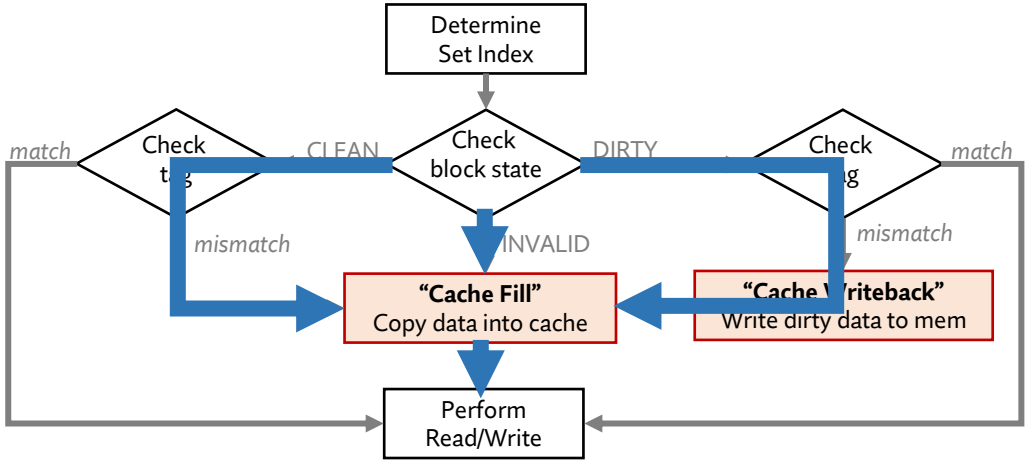
- Course Wrap-Up

Recall: Cache Control Operations

Cache Hit (tag match)

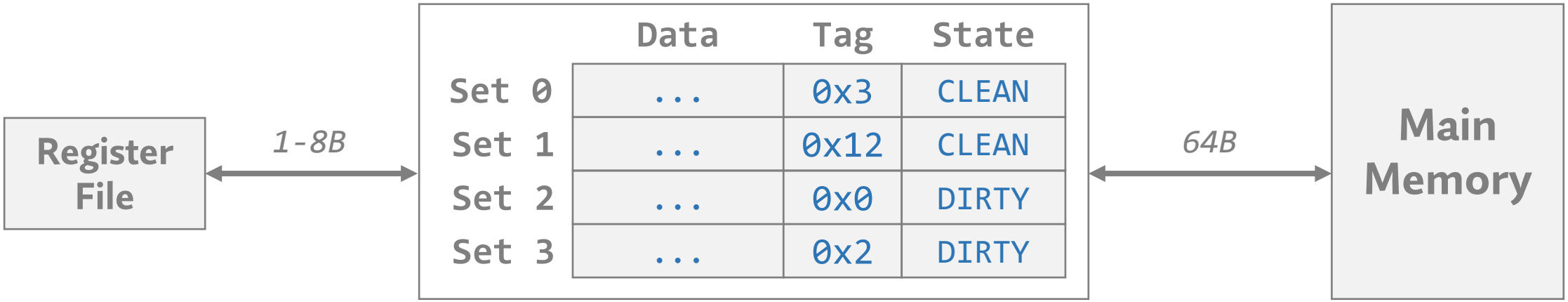


Cache Miss (tag mismatch)



Cache Conflicts

- **Conflict**: different addresses map to the same set (**hash collision**)



Cache

```
li   a0, 0x130
ld   a5, 0(a0)
```

```
li   a0, 0x15f
sb   a5, 0(a0)
```

```
li   a0, 0x280
lh   a5, 0(a0)
```

```
li   a0, 0x8c0
sw   a5, 0(a0)
```

Cache Simulation Example

- Assume 8-bit addresses

Tag	Set Idx	Byte Idx
a_3a_2	a_1a_0	-

Cache (4 Sets; 1B blocks)

	Data	Tag	State
Set 0			
Set 1			
Set 2			
Set 3			

Memory

0x00	A
	B
	C
	D
	E
	F
	G
	H
	I
	J
	K
	L
	M
	N
0x0f	O
	P

Memory Access Trace

```
lb    @ 0xf
sb Q @ 0x1
sb R @ 0xf
lb    @ 0x9
sb S @ 0x7
sb T @ 0x2
lb    @ 0x7
```

Cache Simulation Example: Nonzero Byte Index

- Assume 8-bit addresses

Tag	Set Idx	Byte Idx
a_3a_2	a_1	a_0

Cache (4 Sets; 1B blocks)

	Data	Tag	State
Set 0			
Set 1			

Memory

0x00	A
	B
	C
	D
	E
	F
	G
	H
	I
	J
	K
	L
	M
	N
0x0f	O
	P

Memory Access Trace

```
lb    @ 0xf
sb Q @ 0x1
sb R @ 0x4
```

Agenda

- Simulating Cache Behavior
- **Cache Misses and Performance**
  - Cold Misses
  - Set Associative Caches
- Course Wrap-Up

Three Types of Cache Misses

1. **Cold (compulsory) Miss:** never seen this block before
   - Unavoidable without *prediction*

2. **Conflict Miss**: blocks compete for a set
   - Can make the cache bigger
   - Can change the set index hash function
   - Can allow *(set associative caches)*

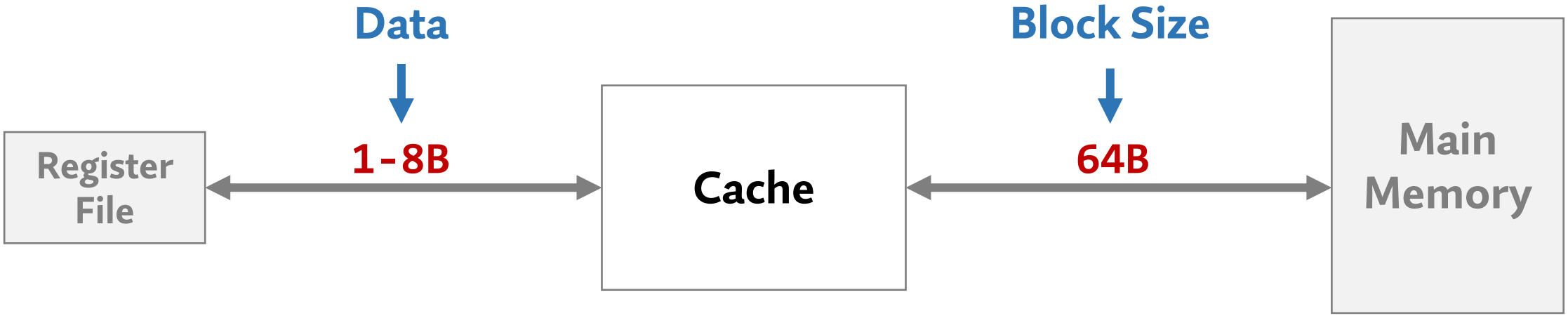
3. **Capacity Miss**: cache is too small

Agenda

- Simulating Cache Behavior
- Cache Misses and Performance
  - **Cold Misses**
  - Set Associative Caches
- Course Wrap-Up

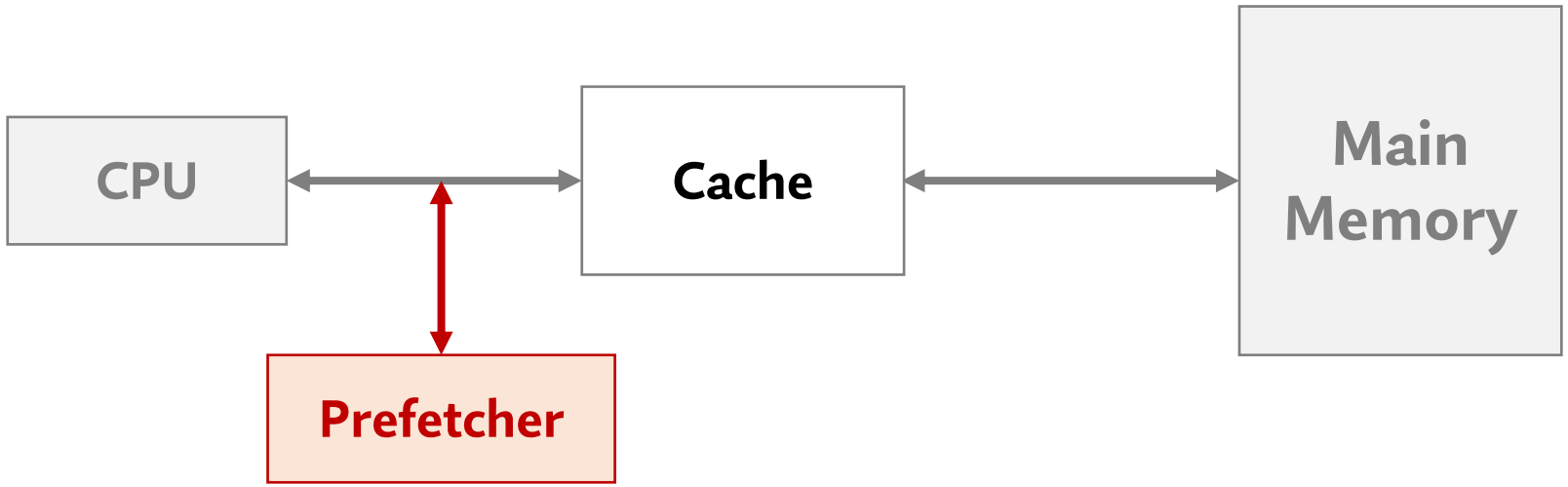
Mitigating Cold Misses

- Requires **predicting** that we need the block beforehand
  - "block size" is already a form of prediction



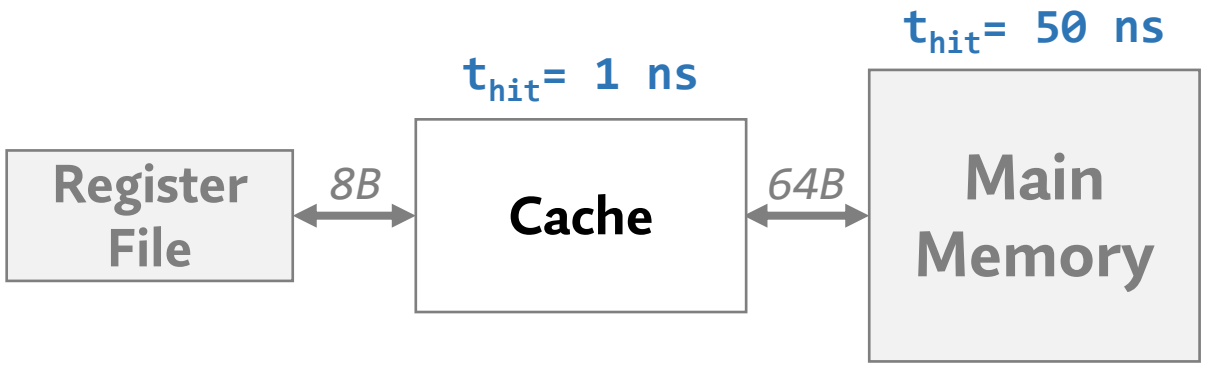
Fancier Prediction: Prefetching

- **Prefetching**: fill data in the cache before we even try to access it



- Lots of prediction algorithms out there:
  - E.g., Replay sequences (e.g., `A, B, C, B, A, ..., A, B, C, <?>`)
  - E.g., Extrapolate patterns (e.g., `A, A+2, A+4, <?>`)

Reasoning about Hits/Misses



- What are the **upper/lower bounds** on memory performance?

C Code

```
void func(uint64_t *a, uint64_t N)
{
    for(int i = 0; i < N; i++)
        a[i]++;
}
```

ASM Code

```
func:
    slli a1, a1, 3
    add  a1, a0, a1
.done:
    bne  a0, a1, .loop
    ret
.loop:
    ld   a5, 0(a0)
    addi a0, a0, 8
    addi a5, a5, 1
    sd   a5, -8(a0)
    j    .done
```

Average Cache Performance

- More reasonable estimate: **average memory access latency**

Depends on the hardware
(cache capacity, block size, etc.)

Depends on the application
(its memory access pattern)

$$t_{avg} = t_{hit} + \%_{miss} \times t_{miss}$$

The cost of a miss ("miss penalty")

$\%_{miss}$ (Miss Rate) Example

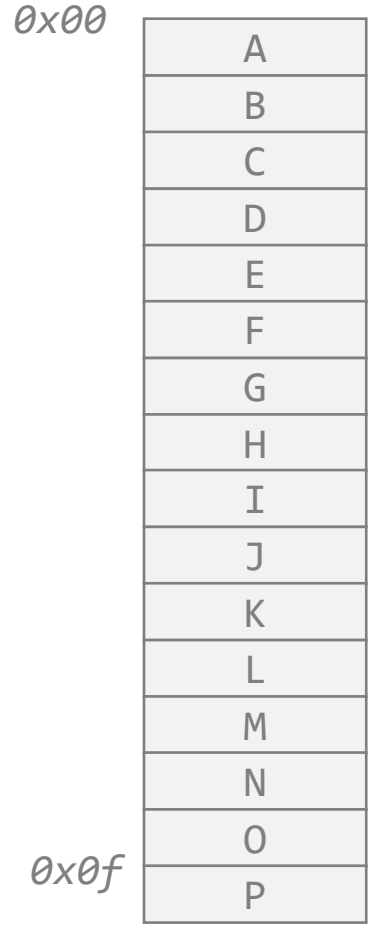
- Assume 8-bit pointers

Tag	Set Idx	Byte Idx
a_3a_2	a_1a_0	-

Cache (4 Sets; 1B blocks)

	Data	Tag	State
Set 0			
Set 1			
Set 2			
Set 3			

Memory

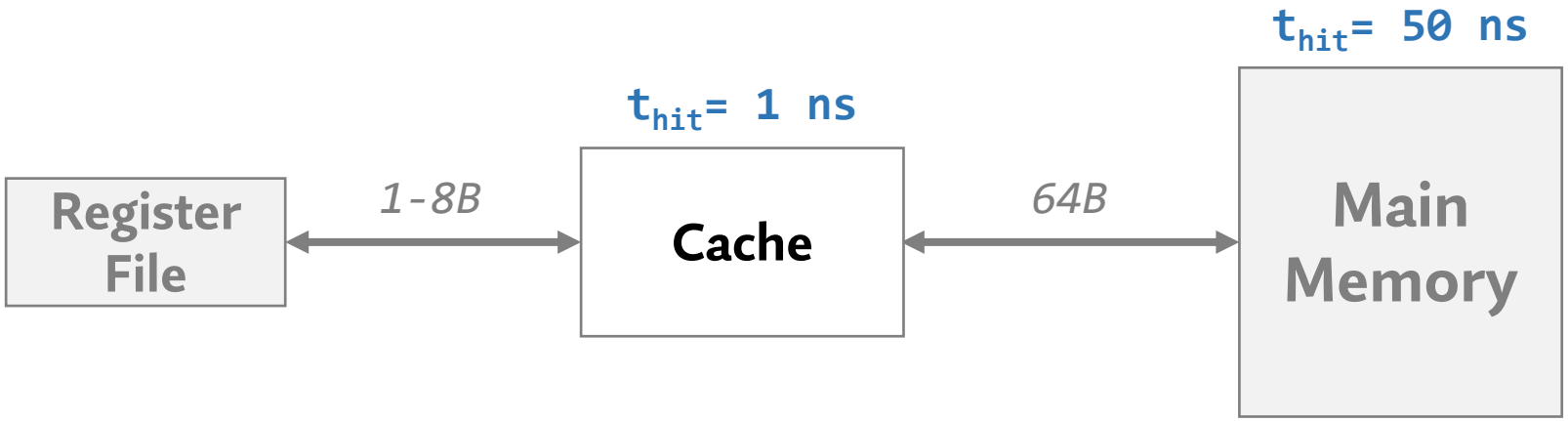


Memory Access Trace

`lb @ 0xf`	**Miss (cold)**
`sb Q @ 0x1`	**Cold miss**
`sb R @ 0xf`	**Hit**
`lb @ 0x9`	**Conflict miss**
`sb S @ 0x7`	**Conflict miss**
`sb T @ 0x2`	**Cold miss**
`lb @ 0x7`	**Hit**

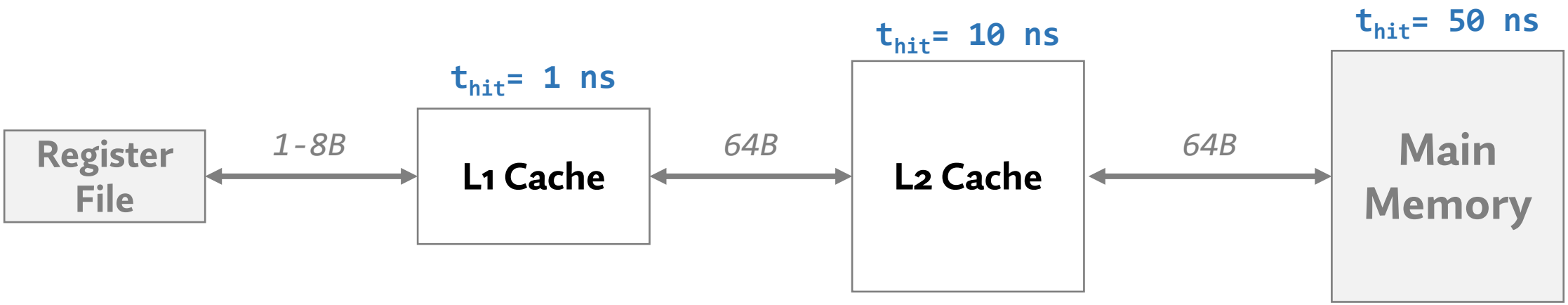
71.4% miss rate

Example Performance



$$t_{avg} = t_{hit} + \%_{miss} \times t_{miss}$$

Example Performance



$$t_{avg} = t_{hit} + \%_{miss} \times t_{miss}$$

Memory Hierarchy Performance

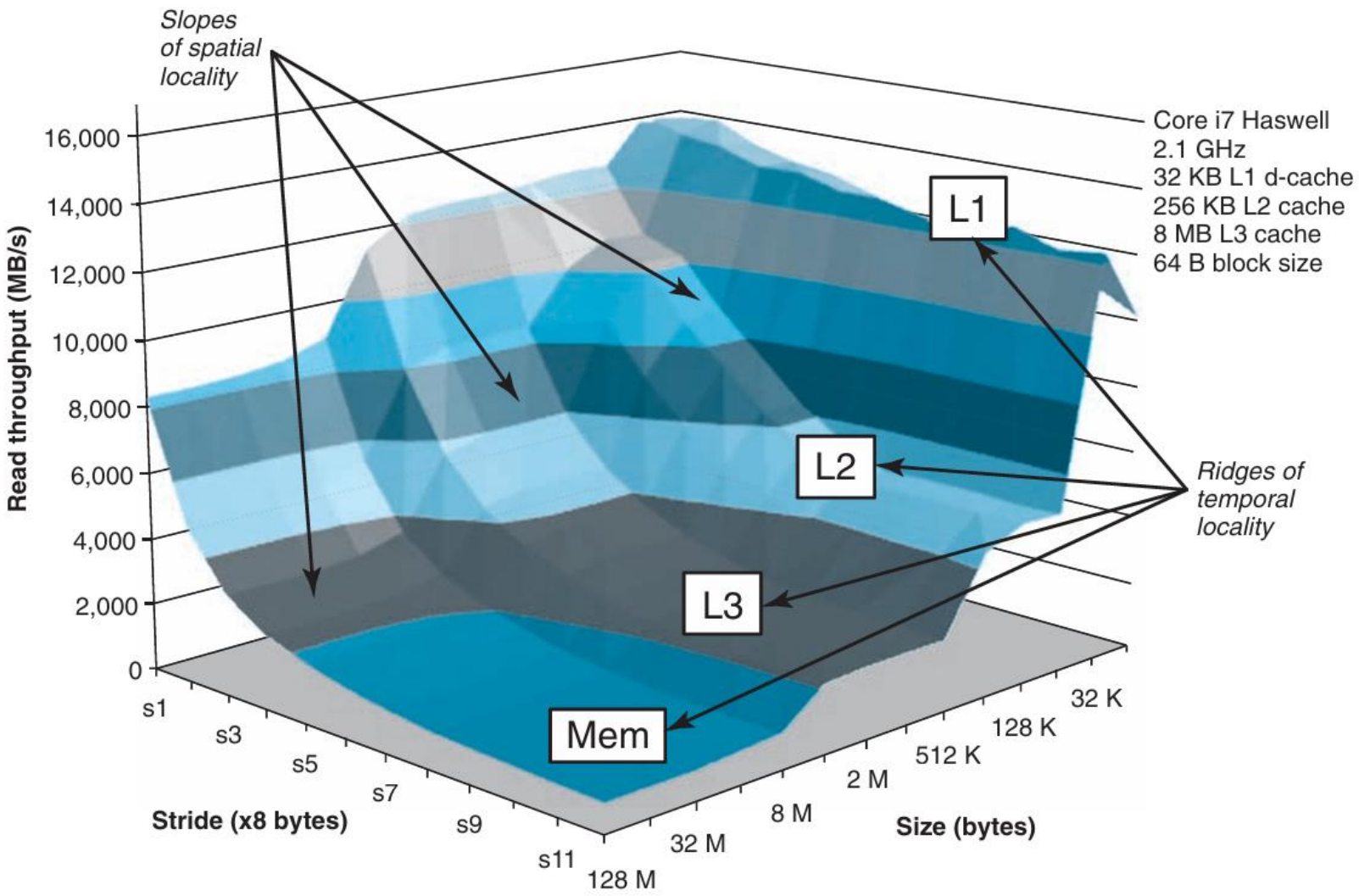
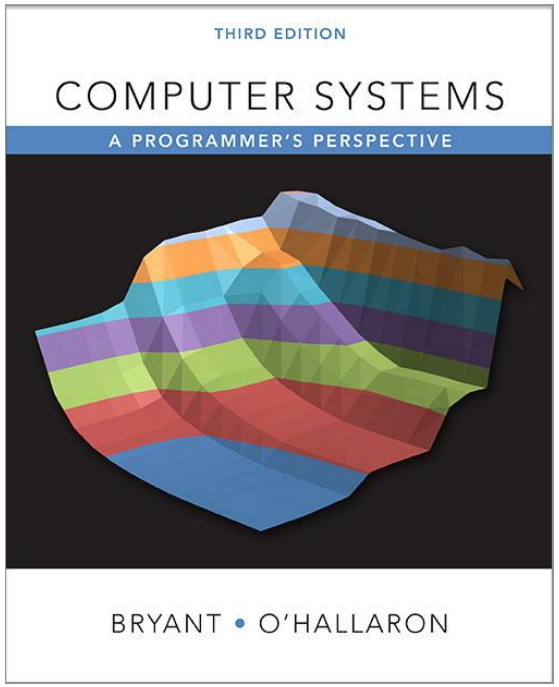


Figure 6.41 **A memory mountain.** Shows read throughput as a function of temporal and spatial locality.



https://csapp.cs.cmu.edu/3e/home.html

CS:APP 3/E, "Section 6.6"

A Word of Caution: Amdahl's Law

- For **any feature** (e.g,. faster caches, faster CPUs):
  - End-to-end benefit is **limited** by how often you're using that feature
  - Fast memory accesses only help during memory accesses ☺

Example:

- Assume **50%** of execution time is spent on memory
- You speed up memory by **10x**
- How much faster is your system?

$$t_{new} = t_{old} \times 0.5 + t_{old} \times 0.5 \times 0.1$$
$$= t_{old} \times 0.55$$
$$= t_{old} \times 0.55 \Rightarrow \textbf{1.8x total speedup}$$

Agenda

- Simulating Cache Behavior
- Types of Cache Misses
  - Cold Misses
  - **Set Associative Caches**
- Course Wrap-Up

Three Types of Cache Misses

1. **Cold (compulsory) Miss:** never seen this block before
   - Unavoidable without *prediction*

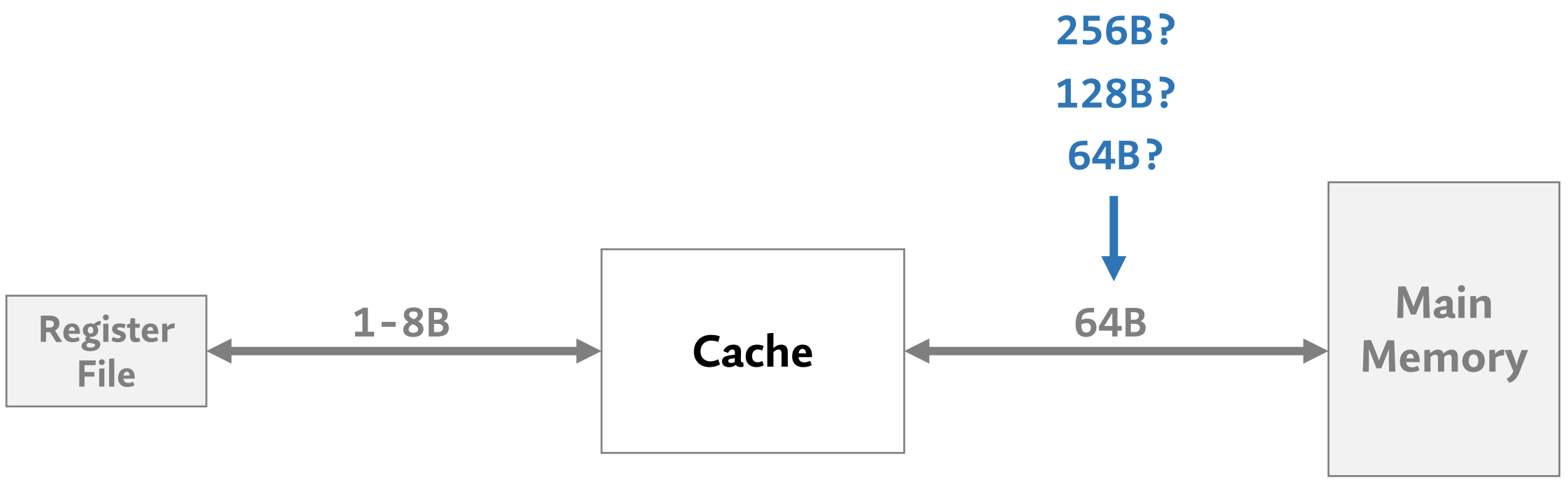
2. **Conflict Miss**: blocks compete for a set
   - Can make the cache bigger
   - Can change the set index hash function
   - Can allow *(set associative caches)*

3. **Capacity Miss**: cache is too small

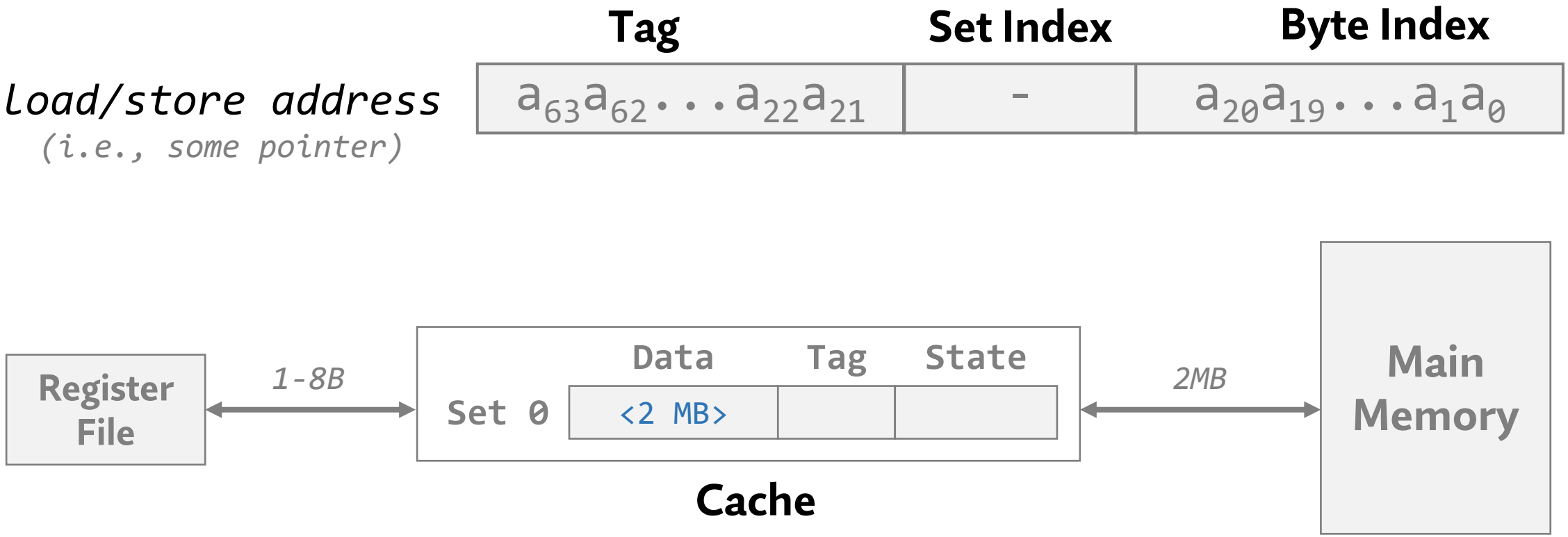
Cache Miss Penalty

- For every miss, we fetch an **entire cache block**

Bigger block = higher miss penalty

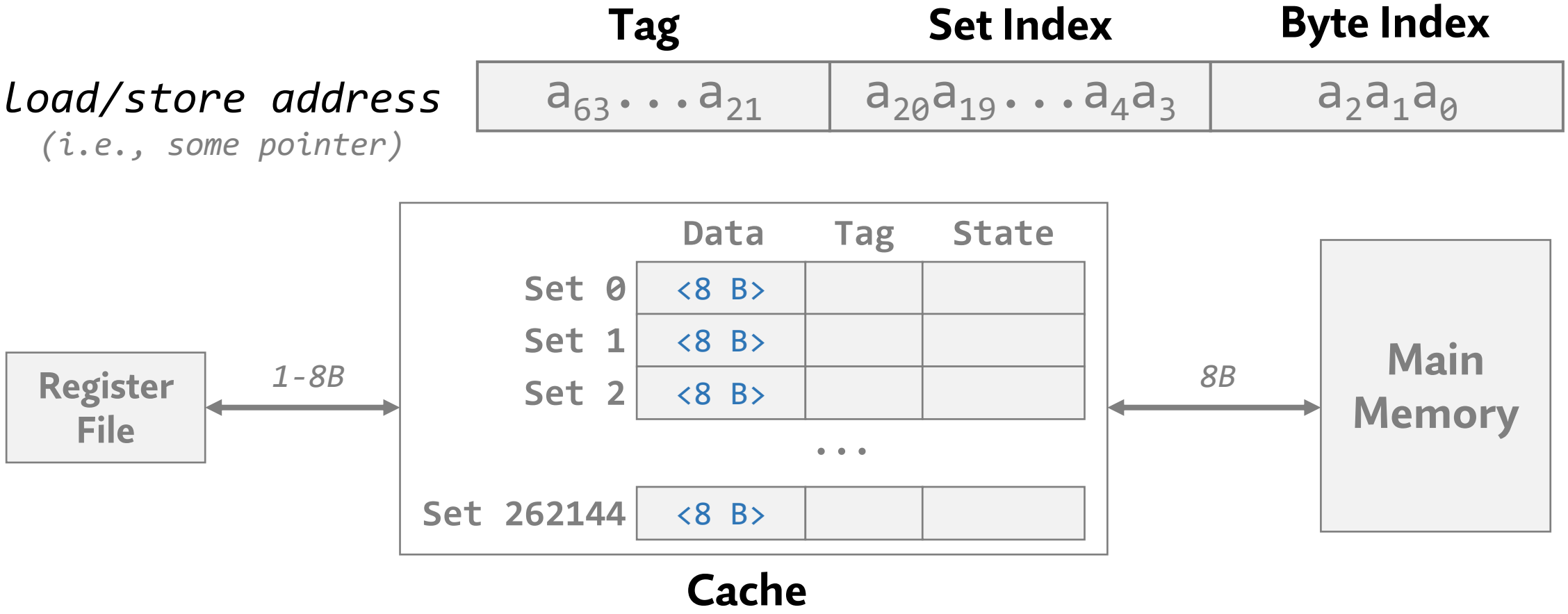


Extreme Example: One Giant Block



- We prefetch ~2MB of data on **every cold miss**
- **Q: Good or bad idea?**

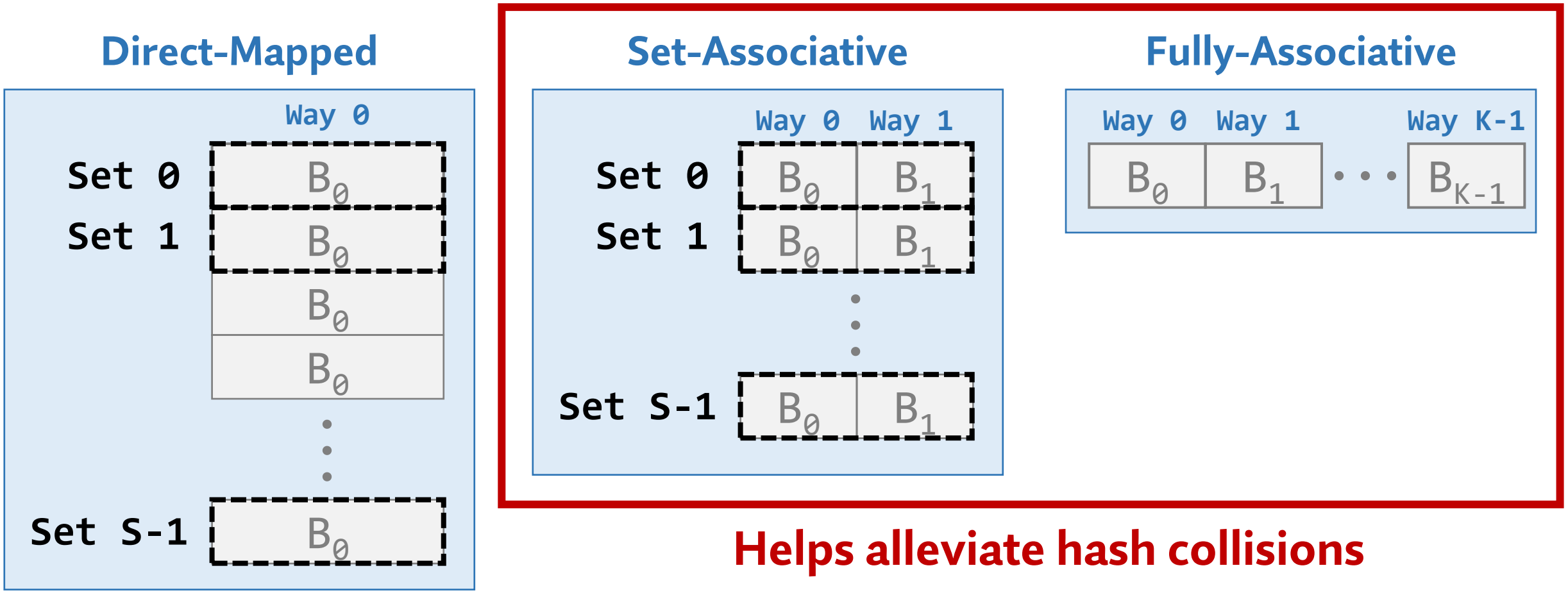
Extreme Example: Many Tiny Blocks



- We **never prefetch** based on cold misses
- **Q: Good or bad idea?**

Set Associative Caches

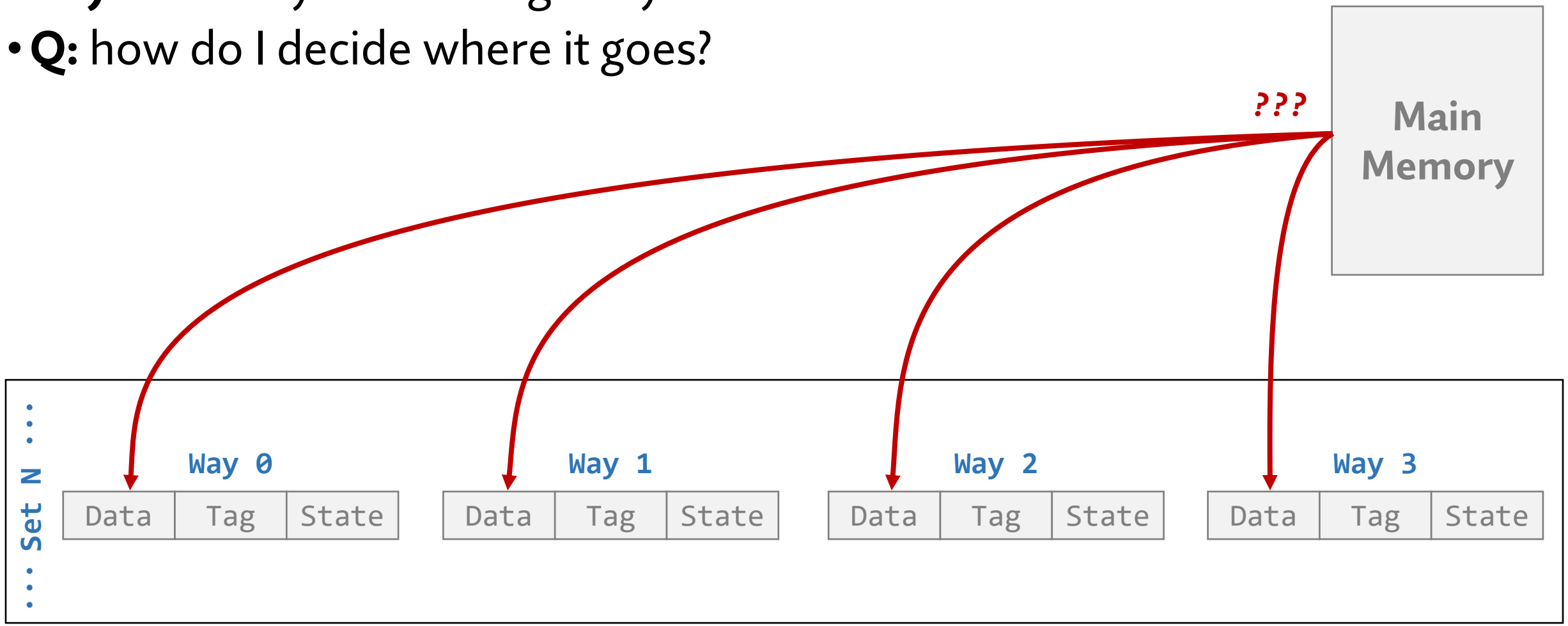
- What if we kept **more than one block** per set?



Helps alleviate hash collisions

Set Associativity

- **Key Idea:** Any block can go anywhere in the set
- **Q:** how do I decide where it goes?



Set-Associative Cache

Associativity Improves Miss Rates

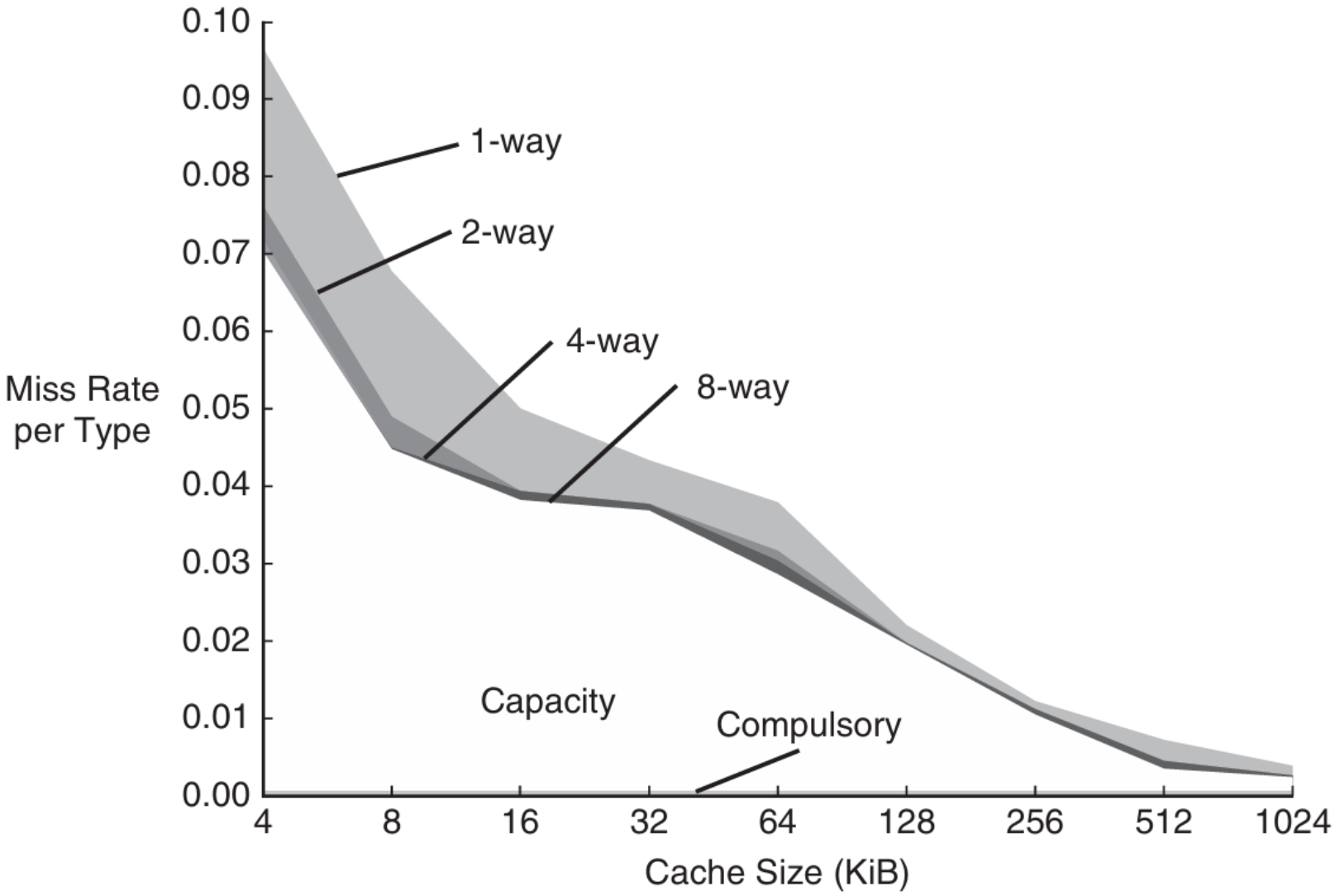


Figure 8.17 Miss rate versus cache size and associativity on SPEC2000 benchmark
Adapted with permission from Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed., Morgan Kaufmann, 2012

Three Types of Cache Misses

1. **Cold (compulsory) Miss:** never seen this block before
   - Unavoidable without *prediction*

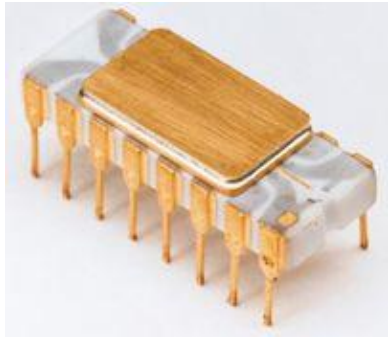
2. **Conflict Miss**: blocks compete for a set
   - Can make the cache bigger
   - Can change the set index hash function
   - Can allow *(set associative caches)*

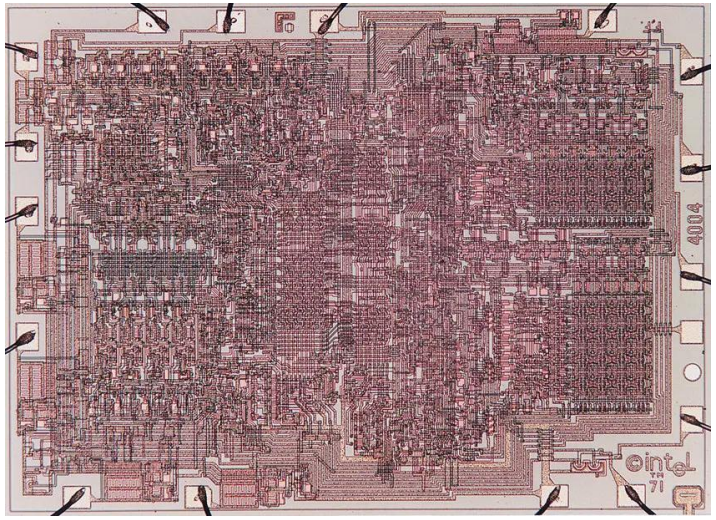
3. **Capacity Miss**: cache is too small

Larger and Larger Caches: $$$

Intel 4004 (1971)

No caches

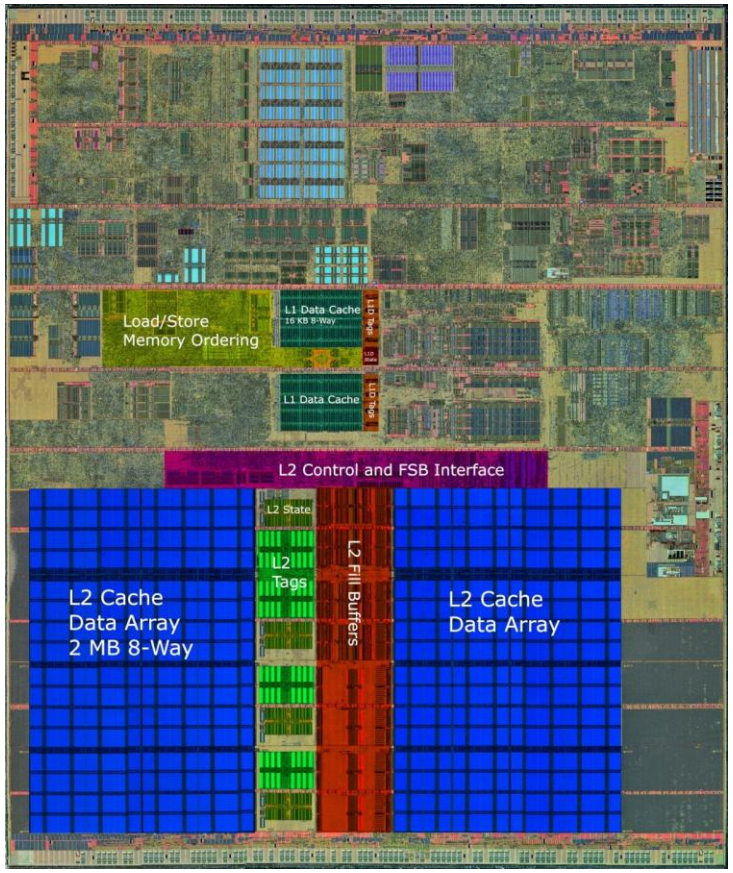




https://en.wikichip.org/wiki/intel/mcs-4/4004

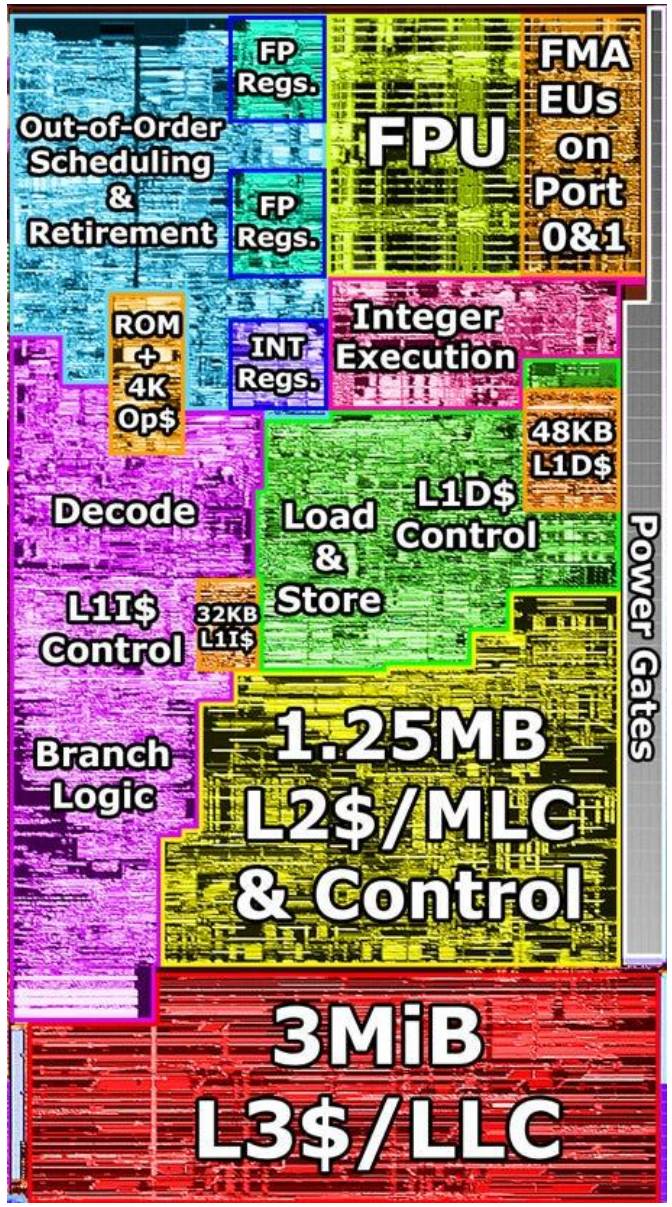
Intel NetBurst (2000)

L1 cache	8-16 KB
L2 cache	128-4096 KB
L3 cache	4-16 MB



https://chipsandcheese.com/p/intels-netburst-failure-is-a-foundation-for-success

Intel Golden Cove (2021)



x8

https://locuza.substack.com/p/die-walkthrough-alder-lake-sp-and

Agenda

- Simulating Cache Behavior
- Types of Cache Misses
  - Cold Misses
  - Set Associative Caches
- **Course Wrap-Up**

CS 211: Intro to Computer Architecture

- Unfortunately, CS 211 won't teach you how to build a computer
  - Need a few more classes for that ☺
- Instead, we'll scratch the surface to prepare you for more

Where To Go Next

Official course title:

Intro to Computer Architecture

A more appropriate title:

Intro to Computing Systems

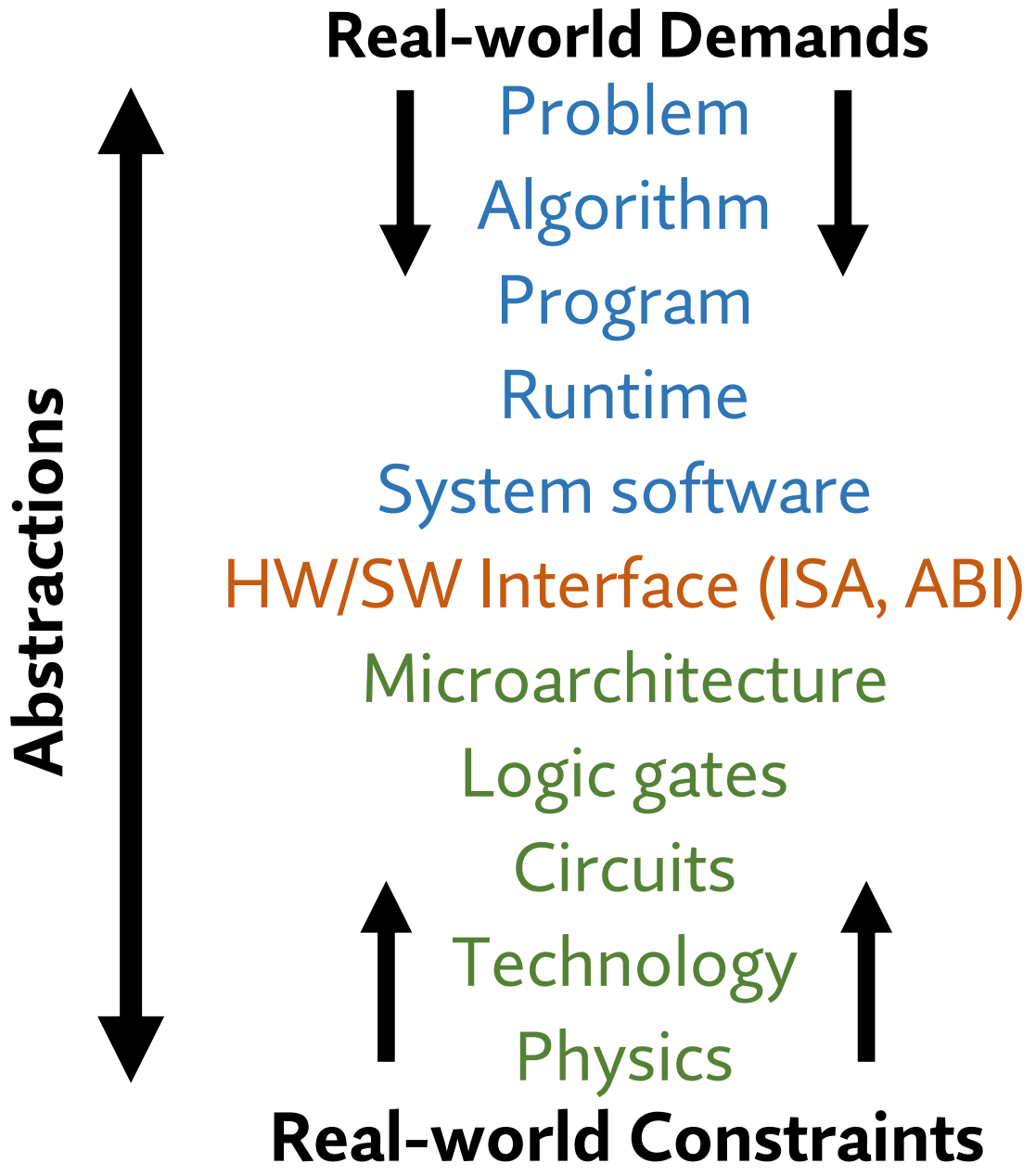
Goals of CS 211

#1: Know your tools
(and their tradeoffs)

#2: Understand computing abstractions
(and know when to break them)

#3: Gain a holistic view of the system

Where To Go Next?

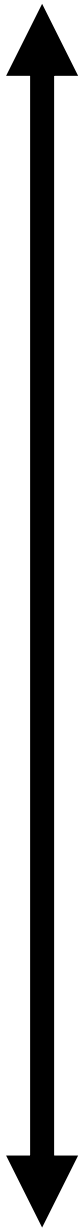


Computer science and engineering is **extremely** broad

Ask yourself:
Which topics did you really enjoy?

Where To Go Next? Anywhere.

Problem
Algorithm
Program
Runtime
System software
HW/SW Interface (ISA, ABI)
Microarchitecture
Logic gates
Circuits
Technology
Physics



C++ / Rust / other C-derived languages
Programming Languages / Runtimes
Systems Programming
Parallel Programming
Distributed Systems
Networking
Compiler Design
Operating Systems
Embedded Systems
Computer / Hardware Security
Hardware Accelerators
Advanced Computer Architecture
Digital Logic Design
Chip Design (e.g., VLSI)

And many more…

Acknowledgements

1. Other CS 211 instructions
2. Other instructors across the world (who inspired these materials)

Our instructional staff this semester

Ramesh Balaji

Neha Jeyaram

Nate Blum

Jerlin Yuen

SIRS Survey

- Very helpful for next year's version of CS 211 ☺

https://sirs.ctaar.rutgers.edu/blue

CS 211: Intro to Computer Architecture

14.2: Caching 3 & Course Wrap-Up

Minesh Patel

Spring 2025 – Thursday 1 May