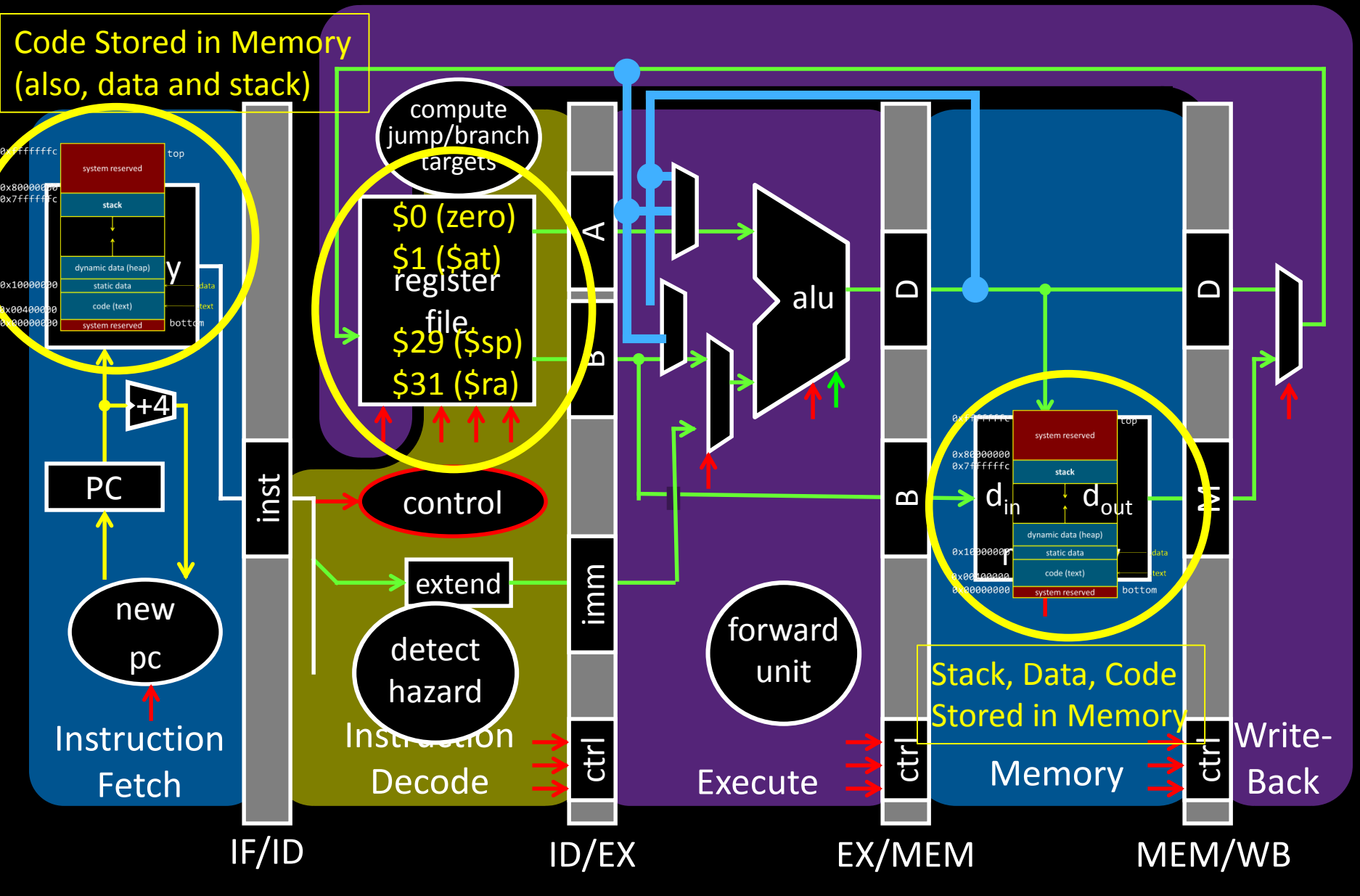
Caches 2

Hakim Weatherspoon
CS 3410, Spring 2013
Computer Science
Cornell University

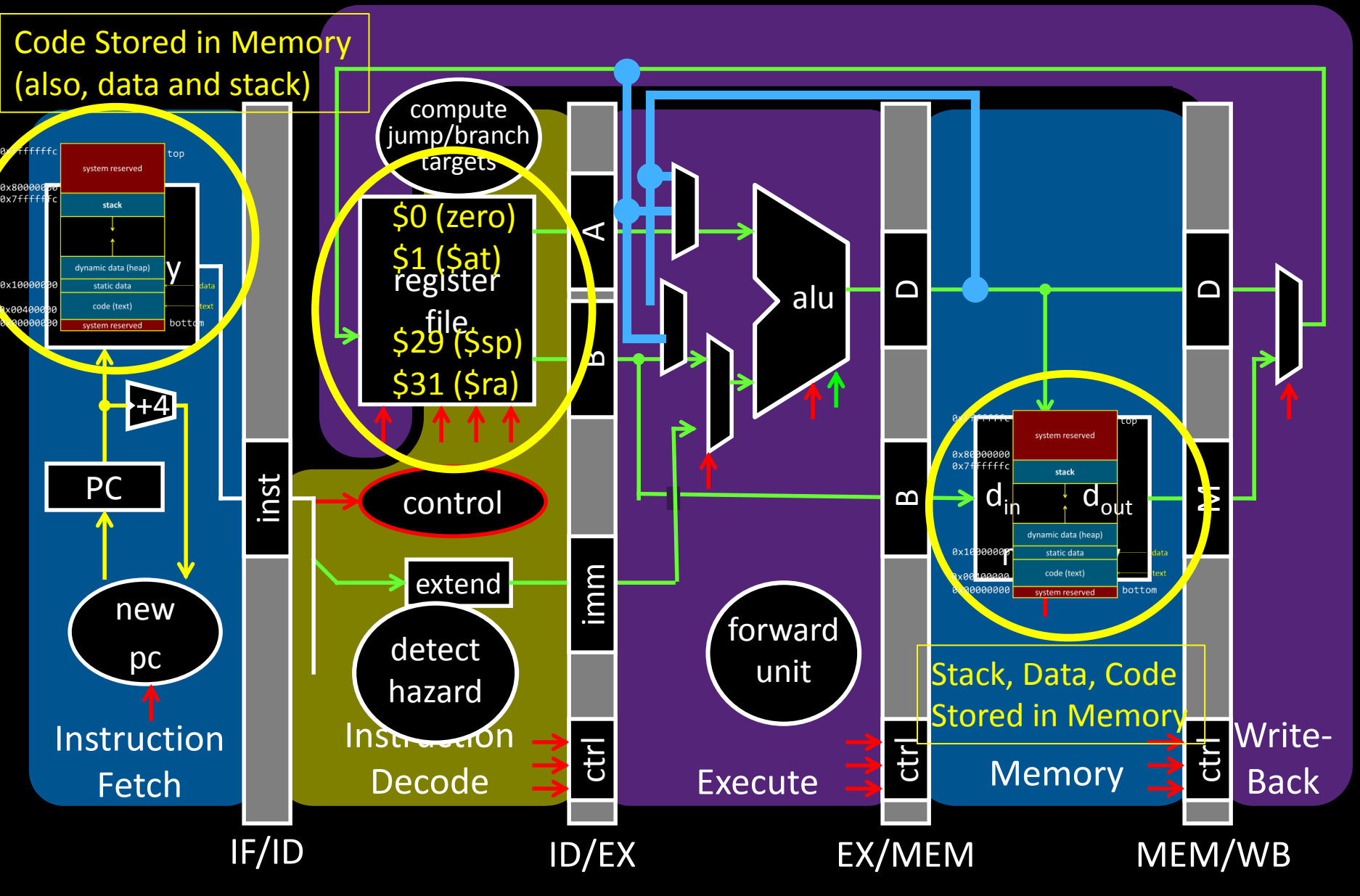
P & H Chapter 5.1-5.3, 5.3-5.4, 5.8, Also, 5.13 & 5.17

Big Picture: Memory



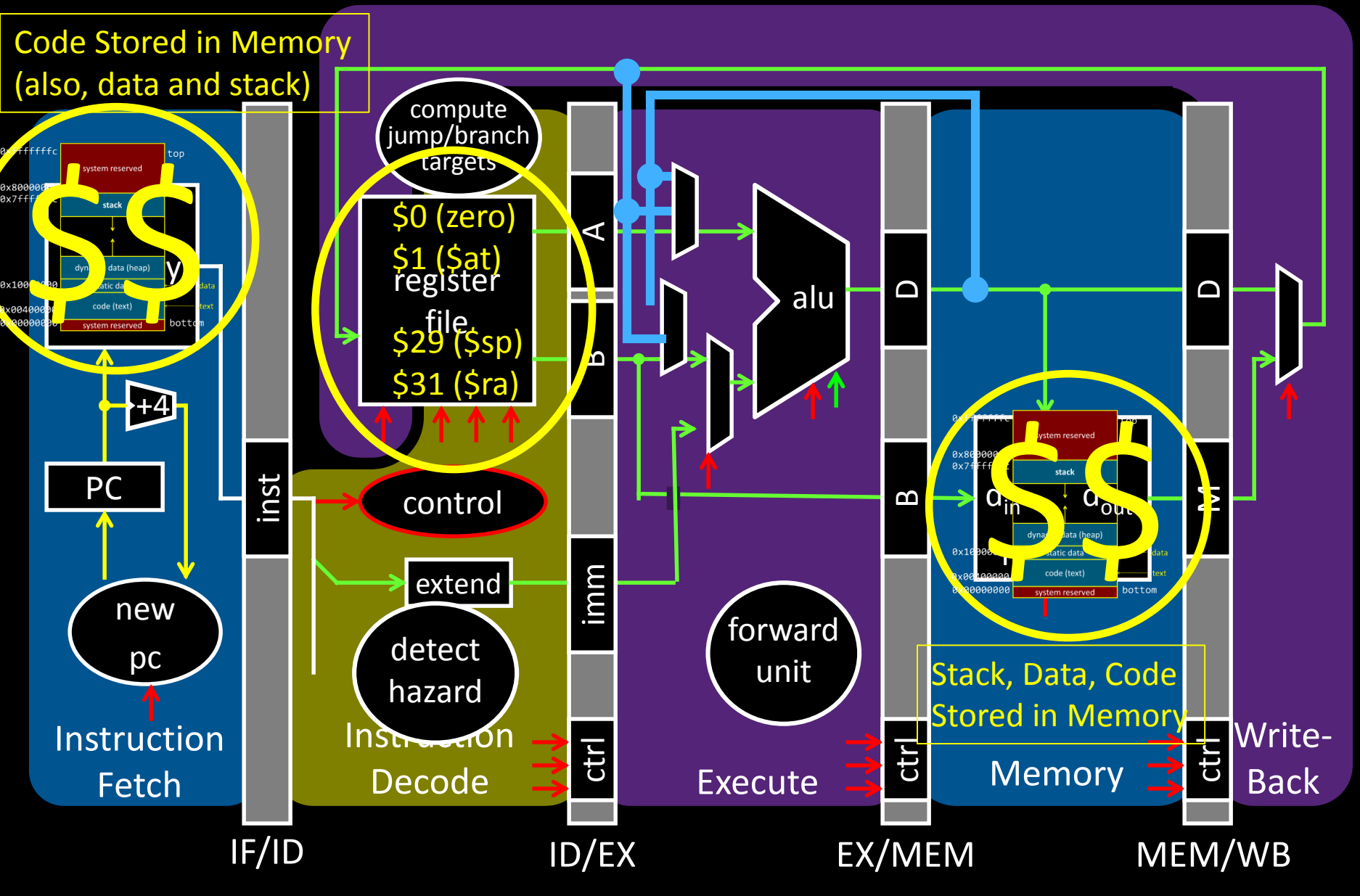
Big Picture: Memory

Memory: big & slow vs Caches: small & fast



Big Picture: Memory

Memory: big & slow vs Caches: small & fast



Big Picture

How do we make the processor fast,
Given that memory is VEEERRRYYYY SLLOOOWWW!!

Big Picture

How do we make the processor fast,
Given that memory is VEEERRRYYYY SLLOOOWWW!!

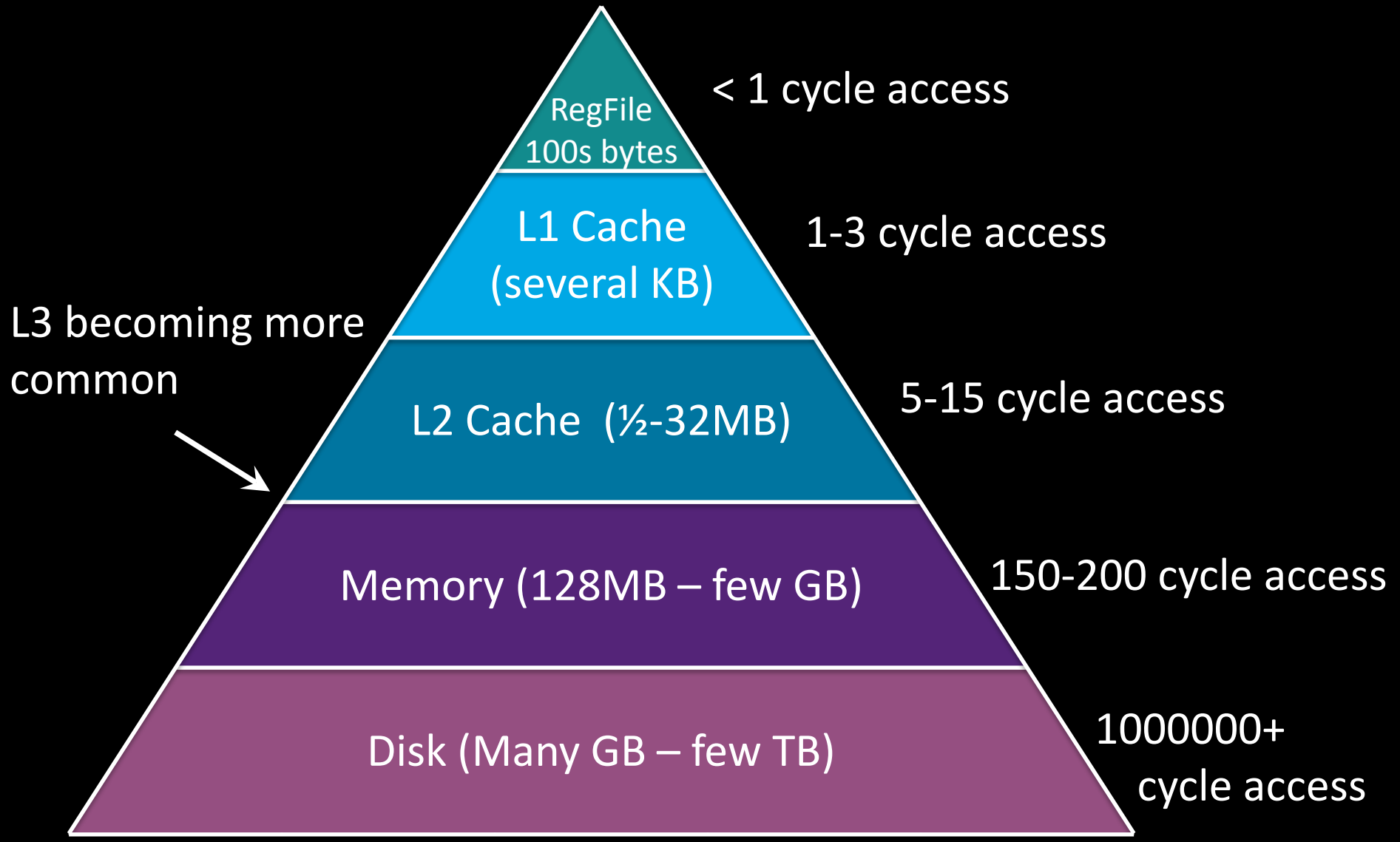
Insight for Caches

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

Benefits

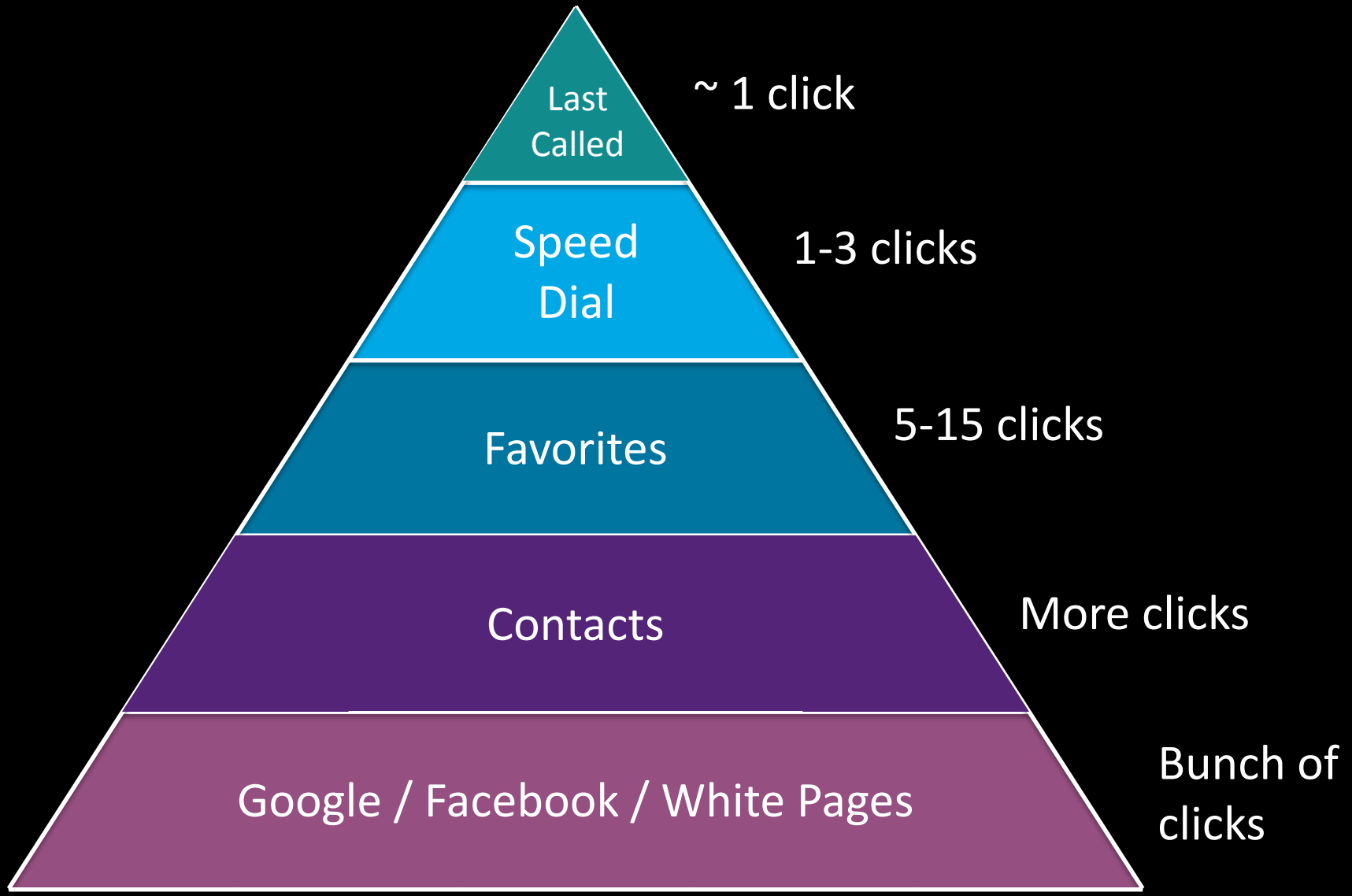
- Abstraction: big & fast memory
  - built from (big & slow memory; DRAM) + (small & fast cache; SRAM)

Memory Hierarchy



*These are rough numbers, mileage may vary.

Just like your Contacts!



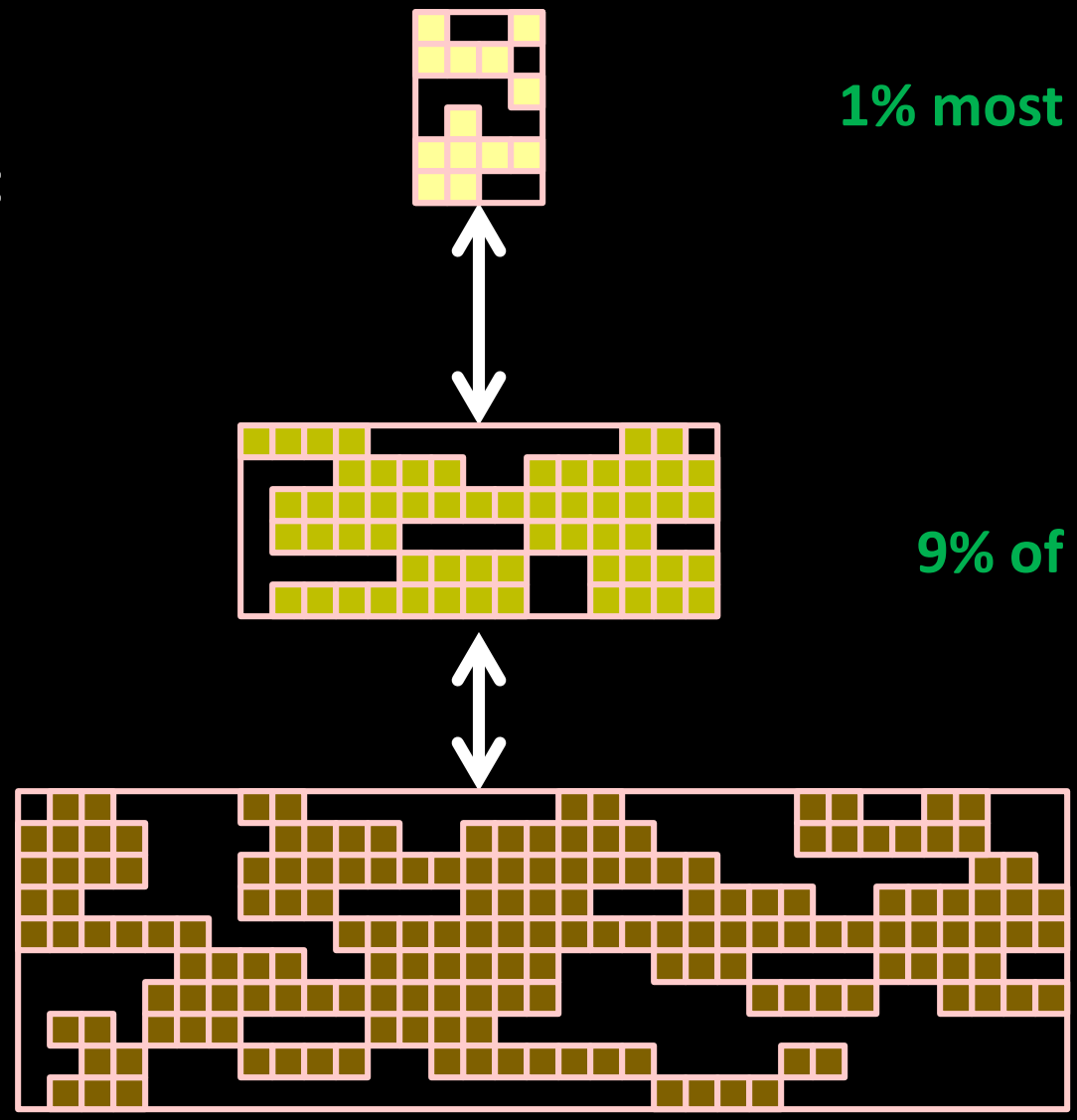
*** Will refer to this analogy using GREEN in the slides.**

Memory Hierarchy

L1 Cache
SRAM-on-chip
1% of data most accessed

L2/L3 Cache
SRAM
9% of data is "active"

Memory DRAM
90% of data inactive (not accessed)



Speed Dial
1% most called people

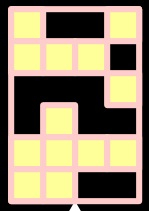
Favorites
9% of people called

Contacts
90% people rarely called

Memory Hierarchy

Memory closer to processor

- small & fast
- stores active data

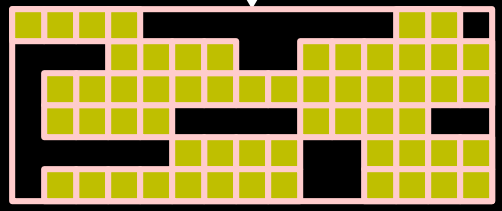


L1 Cache
SRAM-on-chip

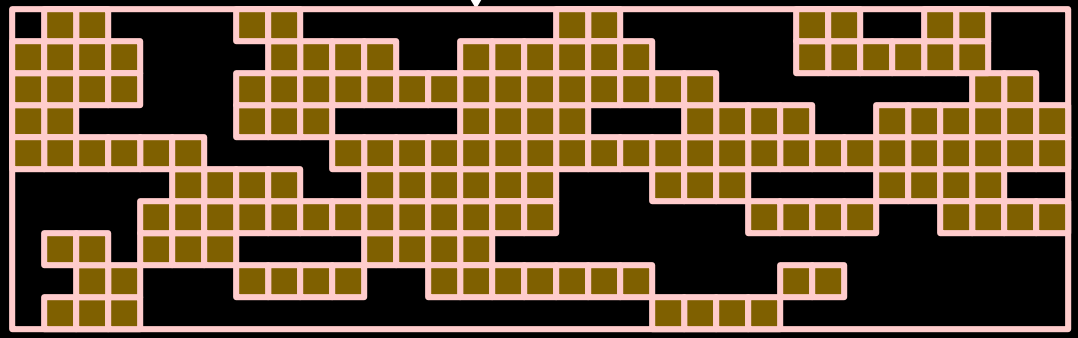
Speed Dial
small & fast

Memory farther from processor

- big & slow
- stores inactive data



L2/L3 Cache
SRAM



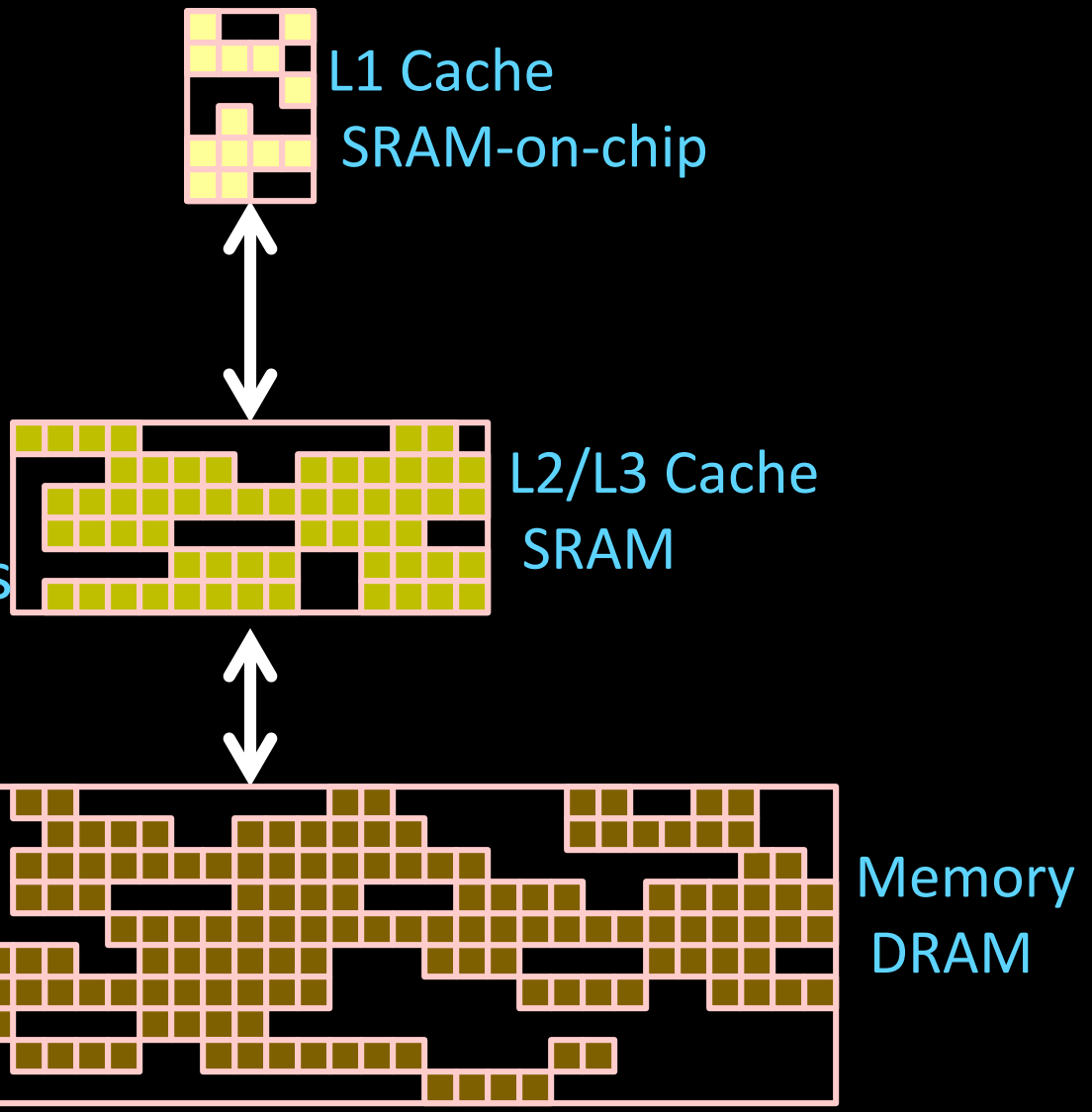
Contact List
big & slow
Memory
DRAM

Memory Hierarchy

Memory closer to processor is fast but small

usually stores subset of memory farther

- "strictly inclusive"



Transfer whole blocks (cache lines):

4kB: disk ↔ RAM

256B: RAM↔L2

64B: L2 ↔ L1

Goals for Today: caches

Comparison of cache architectures:

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

Caching Questions

- How does a cache work?
- How effective is the cache (hit rate/miss rate)?
- How large is the cache?
- How fast is the cache (AMAT=average memory access time)

Next time: Writing to the Cache

- Write-through vs Write-back

Next Goal

How do the different cache architectures compare?

- Cache Architecture Tradeoffs?
- Cache Size?
- Cache Hit rate/Performance?

Cache Tradeoffs

A given data block can be placed…

- … in any cache line → Fully Associative

  (a contact maps to *any* speed dial number)

- … in exactly one cache line → Direct Mapped

  (a contact maps to exactly one speed dial number)

- … in a small set of cache lines → Set Associative

  (like direct mapped, a contact maps to exactly one speed dial number, but many different contacts can associate with the same speed dial number at the same time)

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?	?

Cache Tradeoffs

Compromise: Set-associative cache

Like a direct-mapped cache

- Index into a location
- Fast

Like a fully-associative cache

- Can store multiple entries
  - decreases thrashing in cache
- Search in each element

Direct Mapped Cache

- Use the first letter to index!

Speed Dial

2	ABC	Baker, J.
3	DEF	Dunn, A.
4	GHI	Gill, S.
5	JKL	Jones, N.
6	MNO	Mann, B.
7	PQRS	Powell, J.
8	TUV	Taylor, B.
9	WXYZ	Wright, T.

Contacts

Baker, J.	111-111-1111
Baker, S.	222-222-2222
Dunn, A.	333-333-3333
Foster, D.	444-444-4444
Gill, D.	555-555-5555
Harris, F.	666-666-6666
Jones, N.	777-777-7777
Lee, V.	888-888-8888
Mann, B.	111-111-1119
Moore, F.	222-222-2229
Powell, C.	333-333-3339
Sam, J.	444-444-4449
Taylor, B.	555-555-5559
Taylor, O.	666-666-6669
Wright, T.	777-777-7779
Zhang, W.	888-888-8889

Block Size: 1 Contact Number

Fully Associative Cache

- No index!

Speed Dial

2	Baker, J.
3	Dunn, A.
4	Gill, S.
5	Jones, N.
6	Mann, B.
7	Powell, J.
8	Taylor, B.
9	Wright, T.

Contacts

Baker, J.	111-111-1111
Baker, S.	222-222-2222
Dunn, A.	333-333-3333
Foster, D.	444-444-4444
Gill, D.	555-555-5555
Harris, F.	666-666-6666
Jones, N.	777-777-7777
Lee, V.	888-888-8888
Mann, B.	111-111-1119
Moore, F.	222-222-2229
Powell, C.	333-333-3339
Sam, J.	444-444-4449
Taylor, B.	555-555-5559
Taylor, O.	666-666-6669
Wright, T.	777-777-7779
Zhang, W.	888-888-8889

Block Size: 1 Contact Number

Fully Associative Cache

- No index!

Speed Dial

2	Mann, B.
3	Dunn, A.
4	Taylor, B.
5	Wright, T.
6	Baker, J.
7	Powell, J.
8	Gill, S.
9	Jones, N.

Contacts

Baker, J.	111-111-1111
Baker, S.	222-222-2222
Dunn, A.	333-333-3333
Foster, D.	444-444-4444
Gill, D.	555-555-5555
Harris, F.	666-666-6666
Jones, N.	777-777-7777
Lee, V.	888-888-8888
Mann, B.	111-111-1119
Moore, F.	222-222-2229
Powell, C.	333-333-3339
Sam, J.	444-444-4449
Taylor, B.	555-555-5559
Taylor, O.	666-666-6669
Wright, T.	777-777-7779
Zhang, W.	888-888-8889

Block Size: 1 Contact Number

Fully Associative Cache

- No index!
- Use the initial to offset!

Speed Dial

2	Baker, J.	Baker, S.
3	Dunn, A.	Foster, D.
4	Gill, D.	Harris, F.
5	Jones, N.	Lee, V.
6	Mann, B.	Moore, F.
7	Powell, C.	Sam, J.
8	Taylor, B.	Taylor, O.
9	Wright, T.	Zhang, W.

Block Size: 2 Contact Numbers

Contacts

Baker, J.	111-111-1111
Baker, S.	222-222-2222
Dunn, A.	333-333-3333
Foster, D.	444-444-4444
Gill, D.	555-555-5555
Harris, F.	666-666-6666
Jones, N.	777-777-7777
Lee, V.	888-888-8888
Mann, B.	111-111-1119
Moore, F.	222-222-2229
Powell, C.	333-333-3339
Sam, J.	444-444-4449
Taylor, B.	555-555-5559
Taylor, O.	666-666-6669
Wright, T.	777-777-7779
Zhang, W.	888-888-8889

Fully Associative Cache

- No index!
- Use the initial to offset!

Speed Dial

2	Mann, B.	Moore, F.
3	Powell, C.	Sam, J.
4	Gill, D.	Harris, F.
5	Wright, T.	Zhang, W.
6	Baker, J.	Baker, S.
7	Dunn, A.	Foster, D.
8	Taylor, B.	Taylor, O.
9	Jones, N.	Lee, V.

Block Size: 2 Contact Numbers

Contacts

Baker, J.	111-111-1111
Baker, S.	222-222-2222
Dunn, A.	333-333-3333
Foster, D.	444-444-4444
Gill, D.	555-555-5555
Harris, F.	666-666-6666
Jones, N.	777-777-7777
Lee, V.	888-888-8888
Mann, B.	111-111-1119
Moore, F.	222-222-2229
Powell, C.	333-333-3339
Sam, J.	444-444-4449
Taylor, B.	555-555-5559
Taylor, O.	666-666-6669
Wright, T.	777-777-7779
Zhang, W.	888-888-8889

N-way Set Associative Cache

- 2-way set associative cache
- 8 sets, use first letter to index set
- Use the initial to offset!

Speed Dial

2	ABC	Baker, J.	Baker, S.		
3	DEF	Dunn, A.	Foster, D.		
4	GHI	Gill, D.	Harris, F.		
5	JKL	Jones, N.	Lee, V.		
6	MNO	Mann, B.	Moore, F.		
7	PQRS	Powell, C.	Sam, J.		
8	TUV	Taylor, B.	Taylor, O.		
9	WXYZ	Wright, T.	Zhang, W.		

Block Size: 2 Contact Numbers

Contacts

Baker, J.	111-111-1111
Baker, S.	222-222-2222
Dunn, A.	333-333-3333
Foster, D.	444-444-4444
Gill, D.	555-555-5555
Harris, F.	666-666-6666
Jones, N.	777-777-7777
Lee, V.	888-888-8888
Mann, B.	111-111-1119
Moore, F.	222-222-2229
Powell, C.	333-333-3339
Sam, J.	444-444-4449
Taylor, B.	555-555-5559
Taylor, O.	666-666-6669
Wright, T.	777-777-7779
Zhang, W.	888-888-8889

N-way Set Associative Cache

- 2-way set associative cache
- 8 sets, use first letter to index set
- Use the initial to offset!

Speed Dial

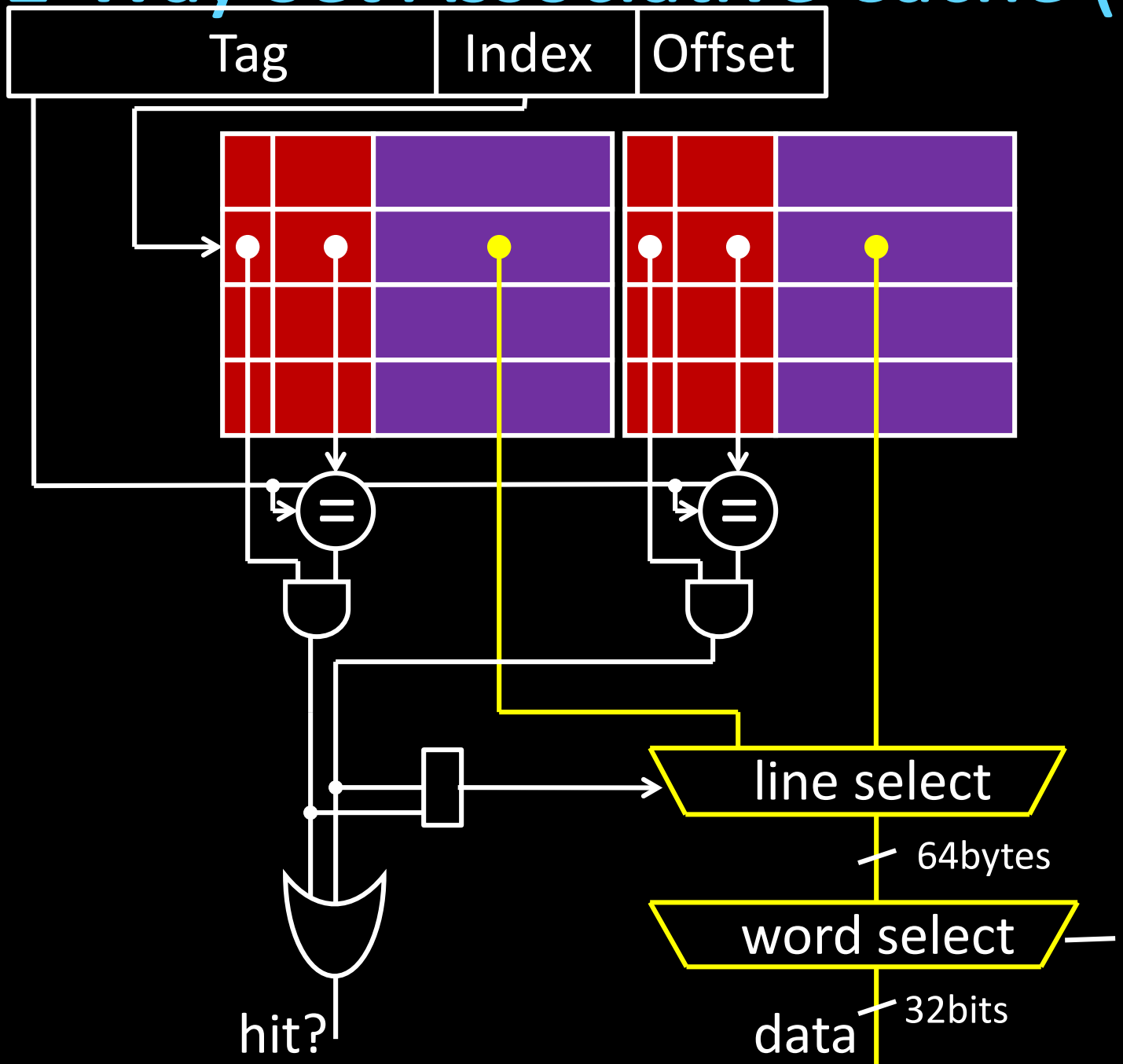
2	ABC	Baker, J.	Baker, S.		
3	DEF	Dunn, A.	Foster, D.		
4	GHI	Gill, D.	Harris, F.	Henry, J.	Isaacs, M.
5	JKL				
6	MNO	Mann, B.	Moore, F.		
7	PQRS	Powell, C.	Sam, J.		
8	TUV	Taylor, B.	Taylor, O.		
9	WXYZ	Wright, T.	Zhang, W.		

Block Size: 2 Contact Numbers

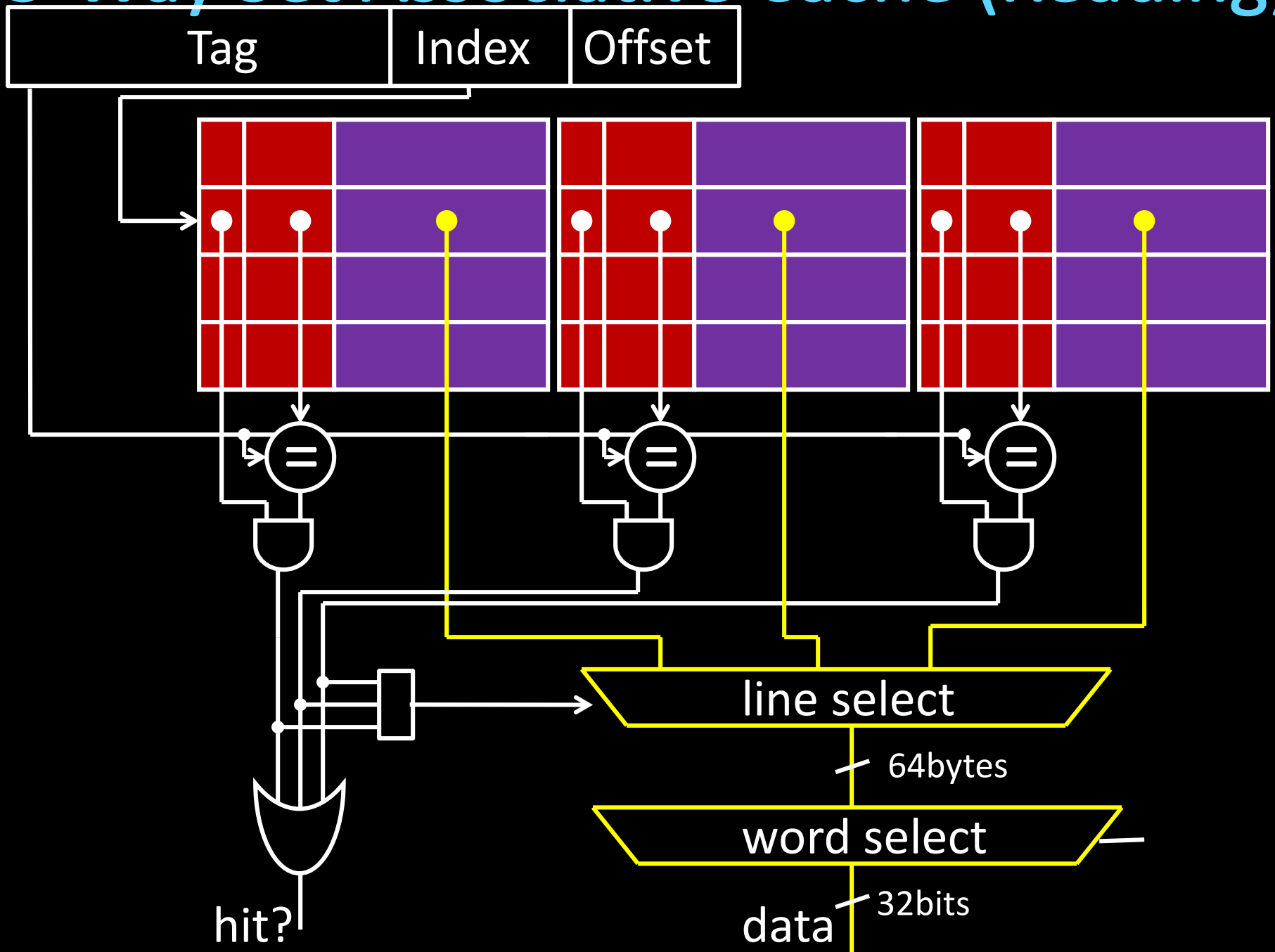
Contacts

Baker, J.	111-111-1111
Baker, S.	222-222-2222
Dunn, A.	333-333-3333
Foster, D.	444-444-4444
Gill, D.	555-555-5555
Harris, F.	666-666-6666
Henry, J.	777-777-7777
Isaacs, M.	888-888-8888
Mann, B.	111-111-1119
Moore, F.	222-222-2229
Powell, C.	333-333-3339
Sam, J.	444-444-4449
Taylor, B.	555-555-5559
Taylor, O.	666-666-6669
Wright, T.	777-777-7779
Zhang, W.	888-888-8889

2-Way Set Associative Cache (Reading)



3-Way Set Associative Cache (Reading)



3-Way Set Associative Cache (Reading)

n bit index, *m* bit offset, **N-way Set Associative**

Q: How big is cache (***data only***)?

Cache of size 2^n sets

Block size of 2^m bytes, N-way set associative

Cache Size: 2^m bytes-per-block x (2^n sets x N-way-per-set)

= N x 2^{n+m} bytes

3-Way Set Associative Cache (Reading)

n bit index, *m* bit offset, **N-way Set Associative**

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

Cache of size 2^n sets

Block size of 2^m bytes, N-way set associative

Tag field: 32 – (n + m), Valid bit: 1

SRAM Size: 2^n sets x N-way-per-set x (block size + tag size + valid bit size)

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

Comparison: Direct Mapped

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

4 cache lines

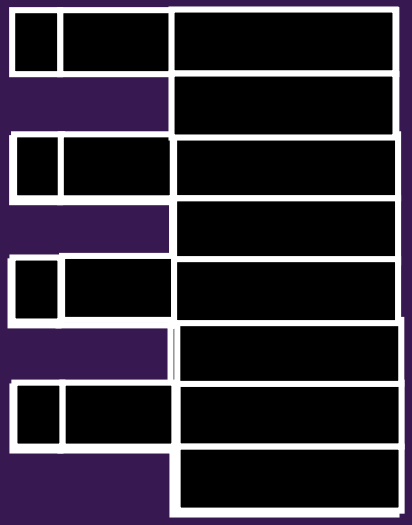
2 word block

2 bit tag field

2 bit index field

1 bit block offset field

tag data



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: Direct Mapped

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

Hits?

a) 3

b) 4

c) 5

d) 7

e) 8

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

4 cache lines

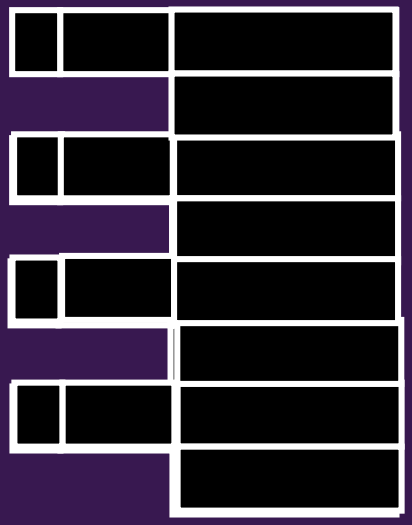
2 word block

2 bit tag field

2 bit index field

1 bit block offset field

tag data



Misses:

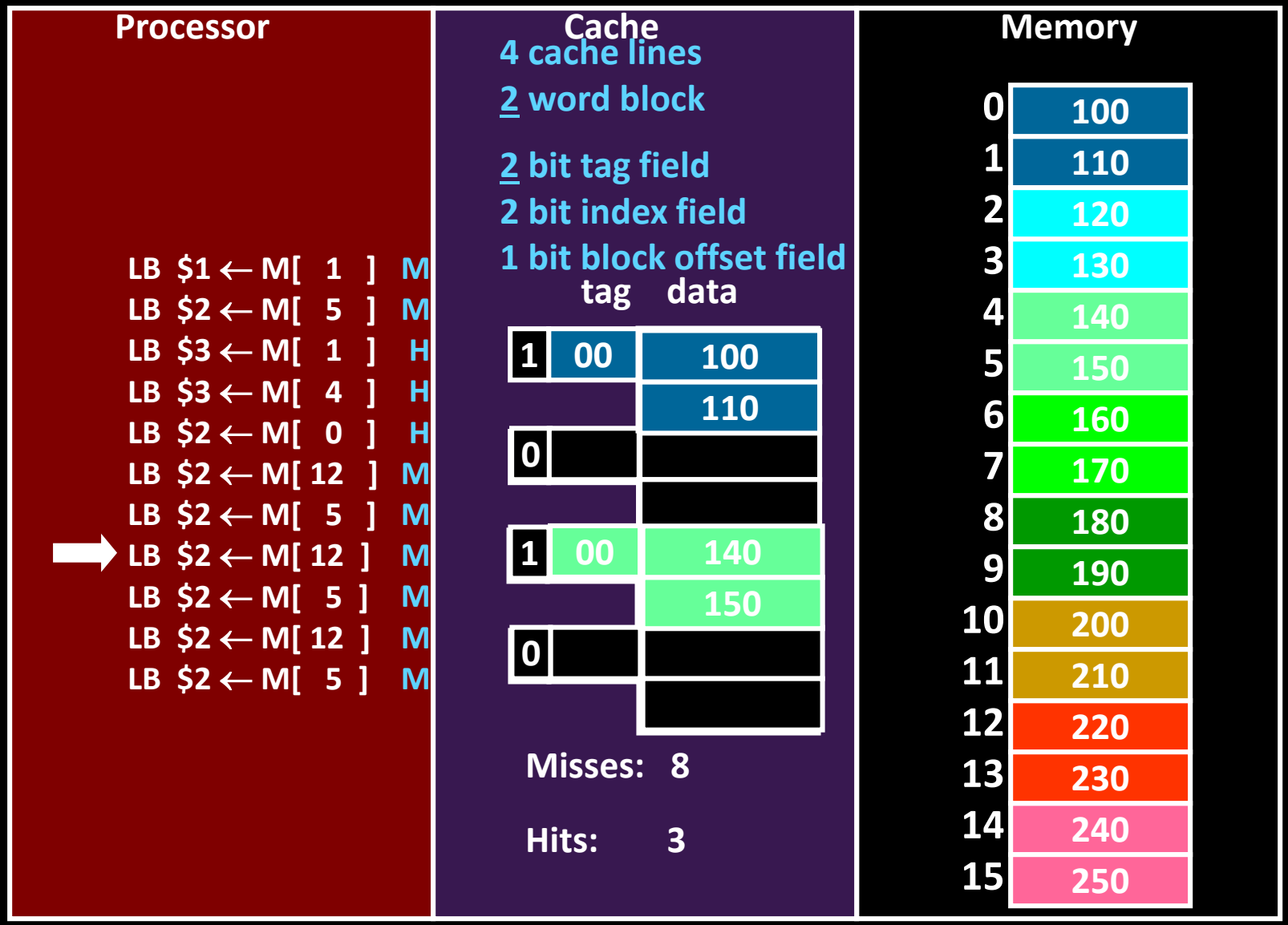
Hits:

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: 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

Hits?

a) 3

b) 4

c) 5

d) 7

e) 8

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

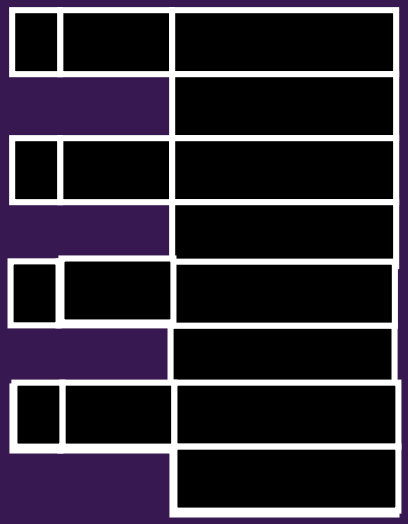
4 cache lines

2 word block

4 bit tag field

1 bit block offset field

tag data



Misses:

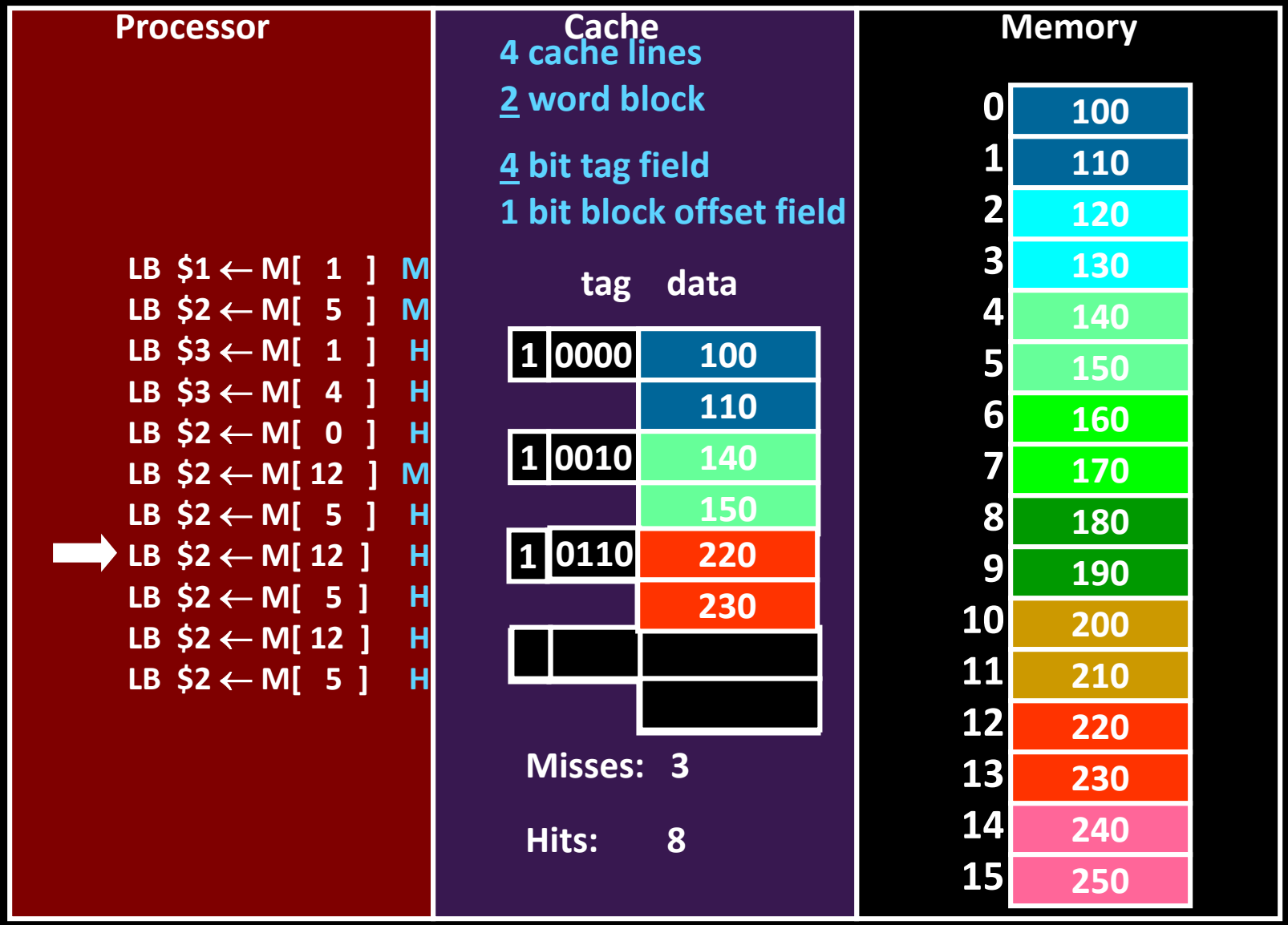
Hits:

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: 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

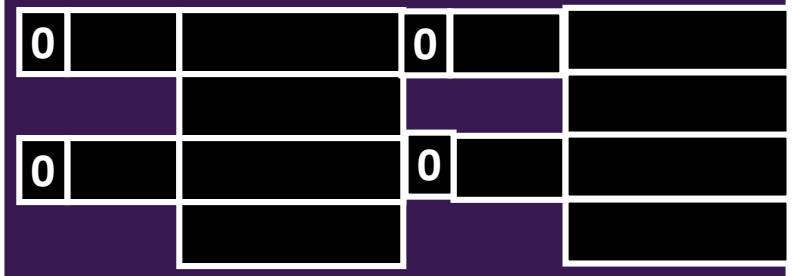
Processor

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

tag data



Hits?
a) 3
b) 4
c) 5
d) 6
e) 7

Misses:

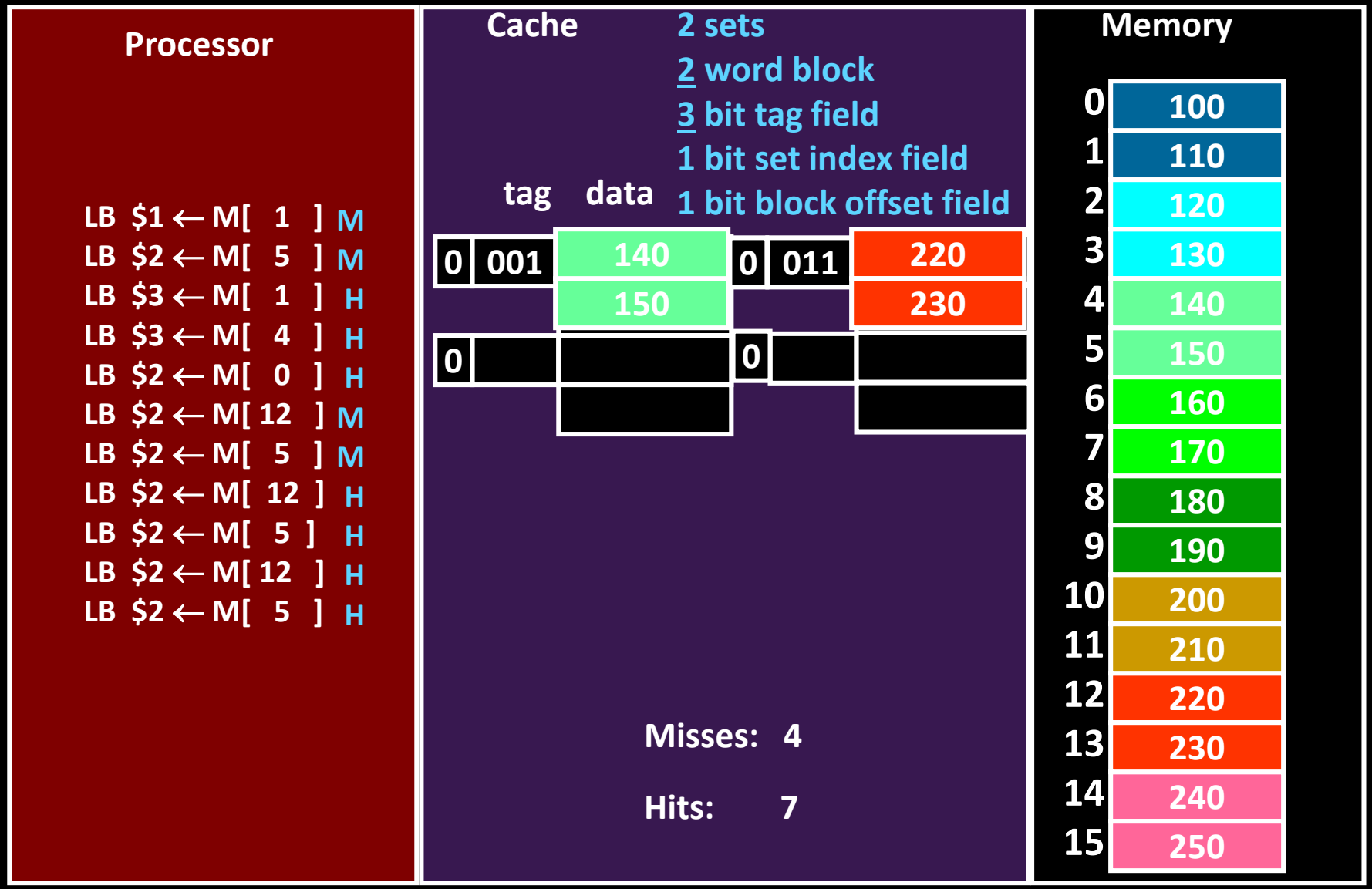
Hits:

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



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

Eviction

Which cache line should be evicted from the cache to make room for a new line?

- Direct-mapped
  - no choice, must evict line selected by index
- Associative caches
  - Random: select one of the lines at random
  - Round-Robin: similar to random
  - FIFO: replace oldest line
  - LRU: replace line that has not been used in the longest time

Takeaway

Direct Mapped → fast but low hit rate

Fully Associative → higher hit cost, but higher hit rate

N-way Set Associative → middleground

Summary

- Illusion: Large, fast, memory
- 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
  - Direct mapped → fast, low hit rate, low hit rate
  - Fully Associative → higher hit cost, higher hit rate
  - N-way set associative → middleground

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