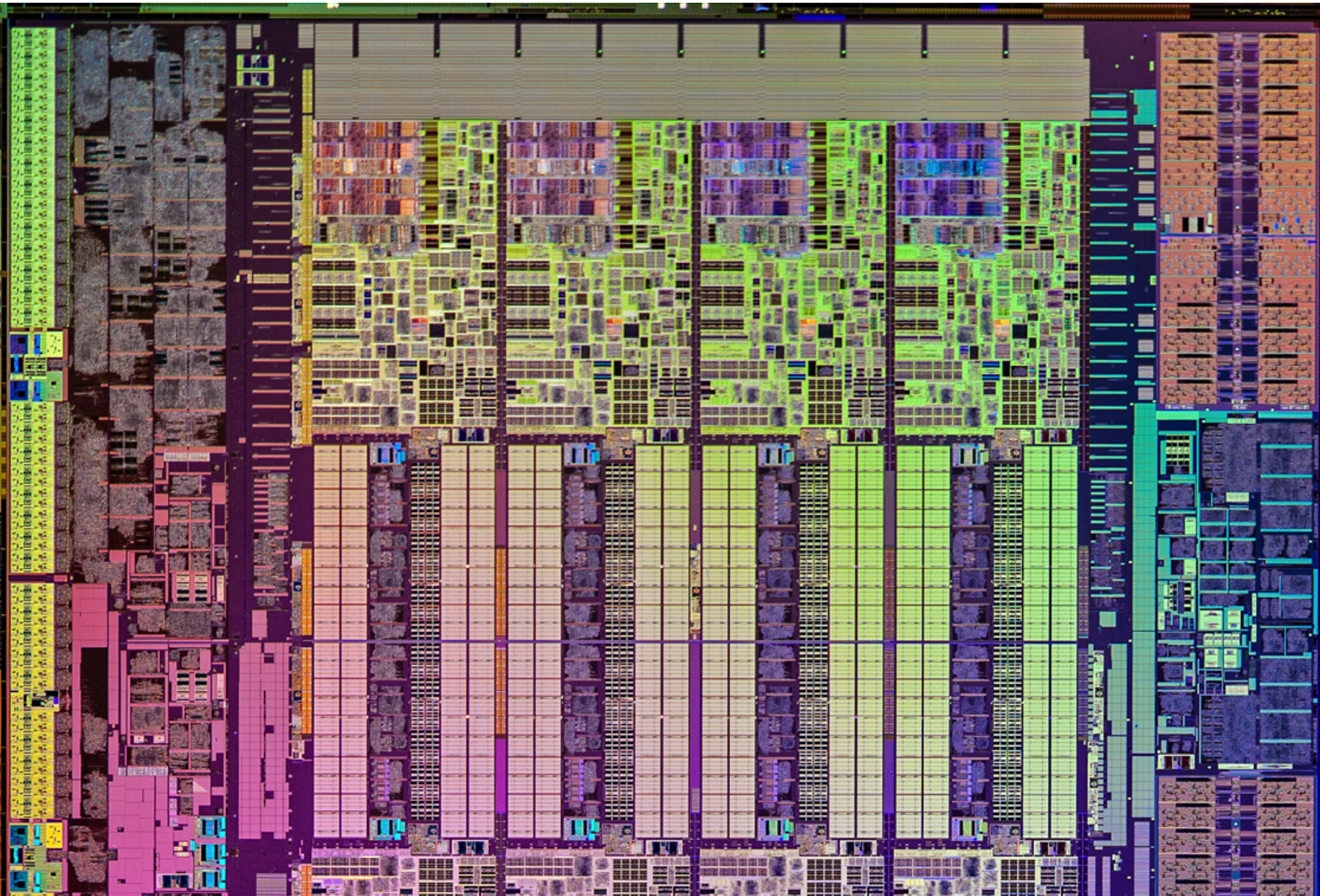
Cache memory



Caches

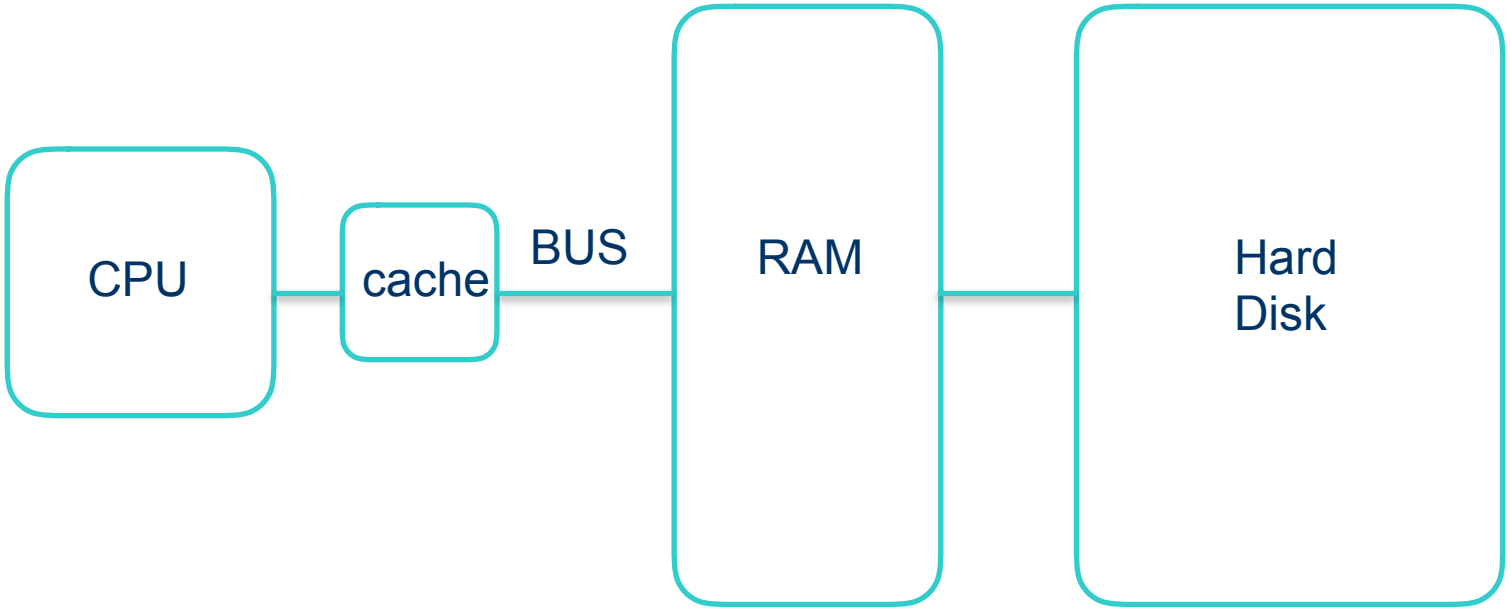
On completion of this lecture you will be able to:

- Explain the role of the tag, index and offset in Cache mapping
- Discuss the differences between Direct and Associative mapping
- Explain the differences between the different types of cache miss
- Explain what can happen to cached data on a write
- Discuss strategies to improve cache performance

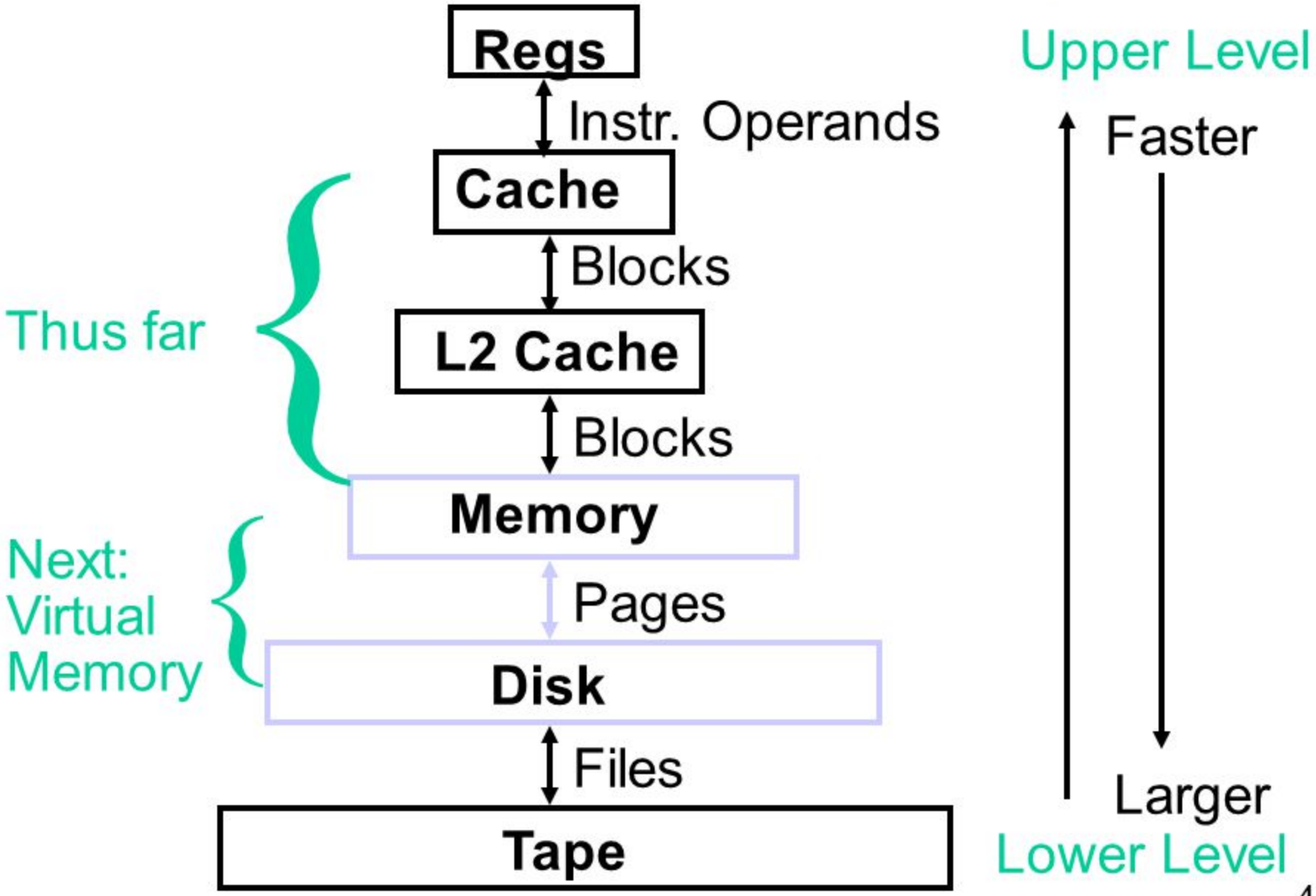
Cache Recap

A small piece of memory in-between the CPU and main memory

Why?



Another View of the Memory Hierarchy



Cache Optimization Questions

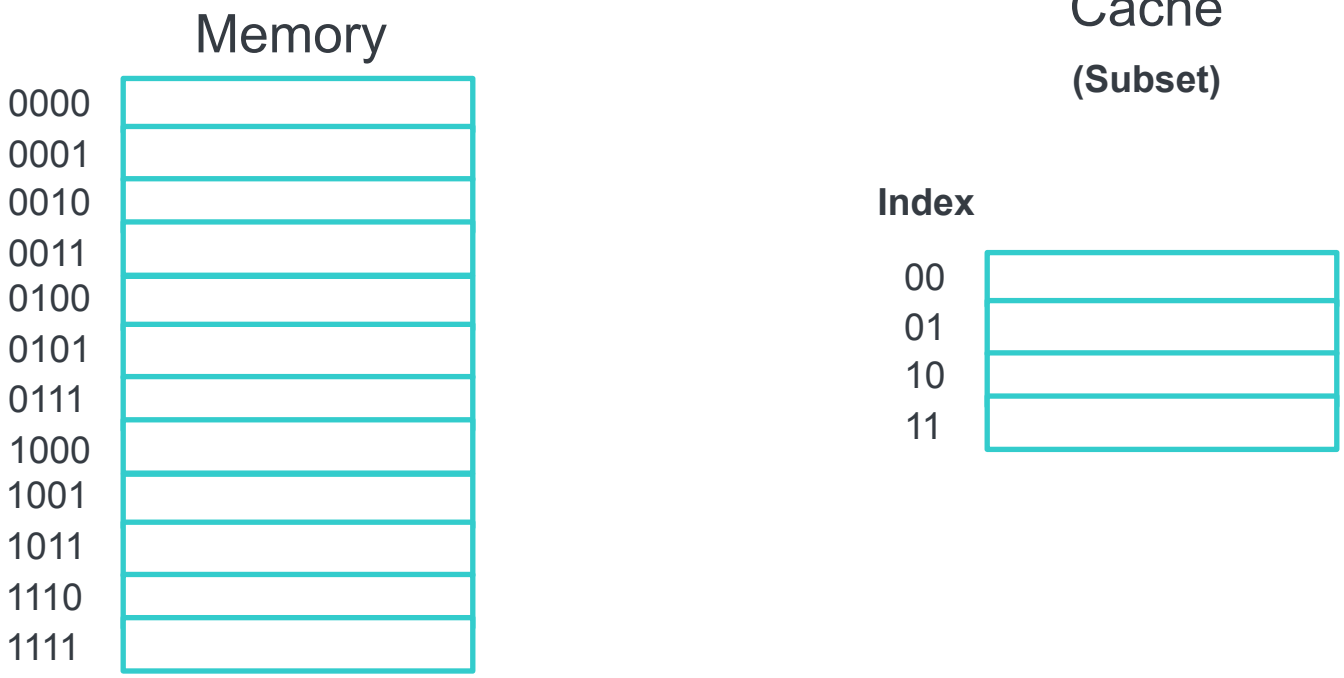
1. Where in cache to put a new block?

2. How to identify/find it once it's there?

3. Which block should be removed to make space for a new one after a miss?

4. What happens when information in a block is modified?

Memory to Cache



- Cache controller: hardware that checks tags to determine if in cache

Memory to Cache

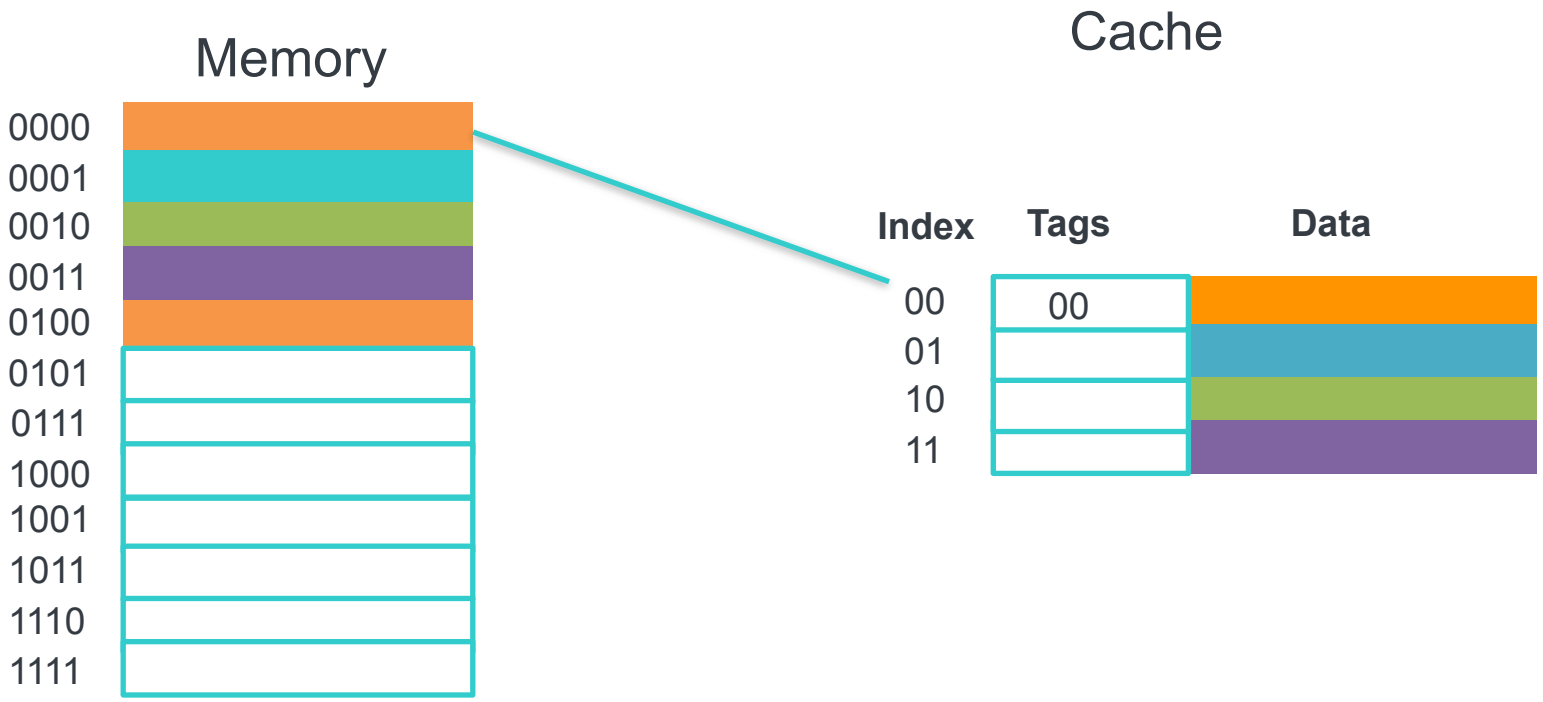
We can compute which memory location maps to which cache index by:

- taking mod 4 OR
- taking the last 2 least significant bits

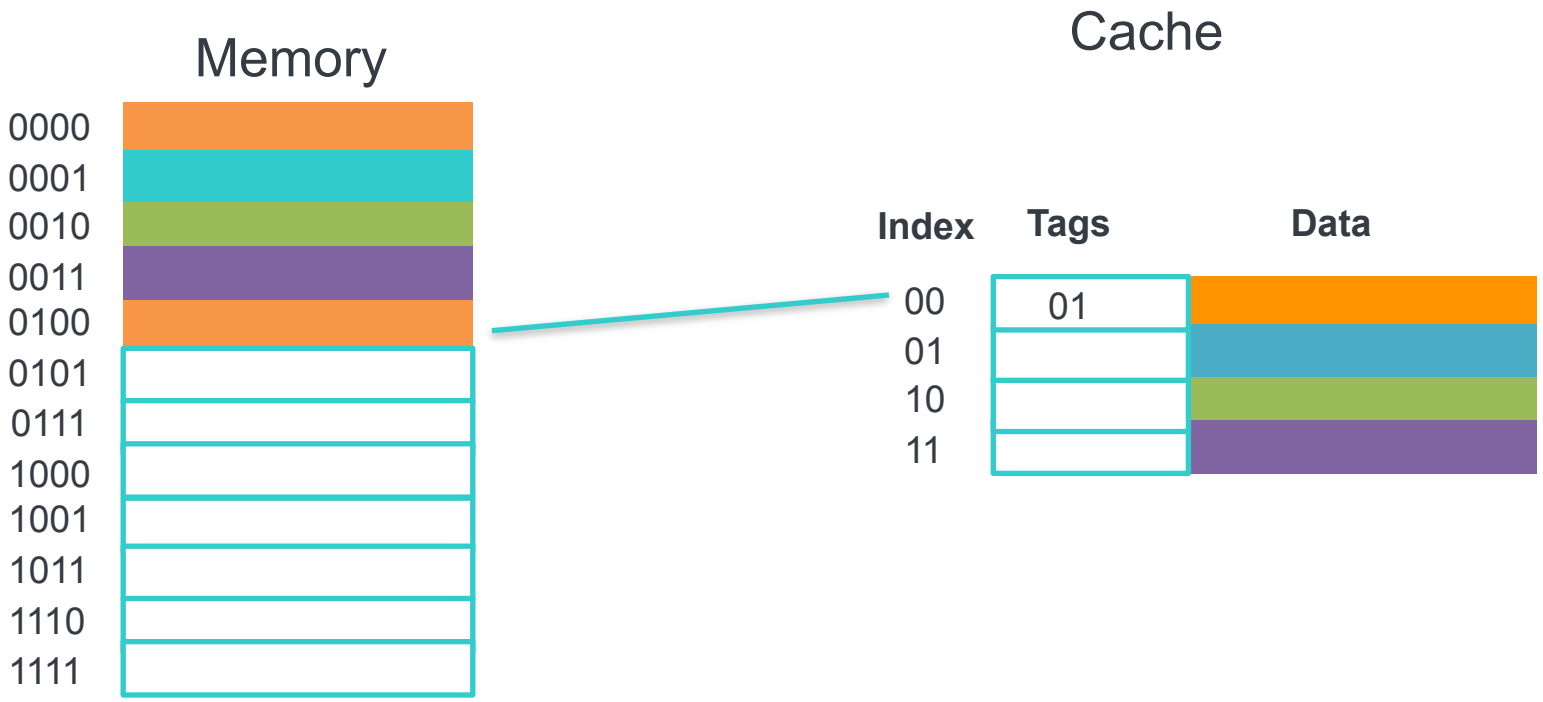
Memory to Cache

Both memory locations map to the same cache location.
How does the cache know which memory location it is referencing?

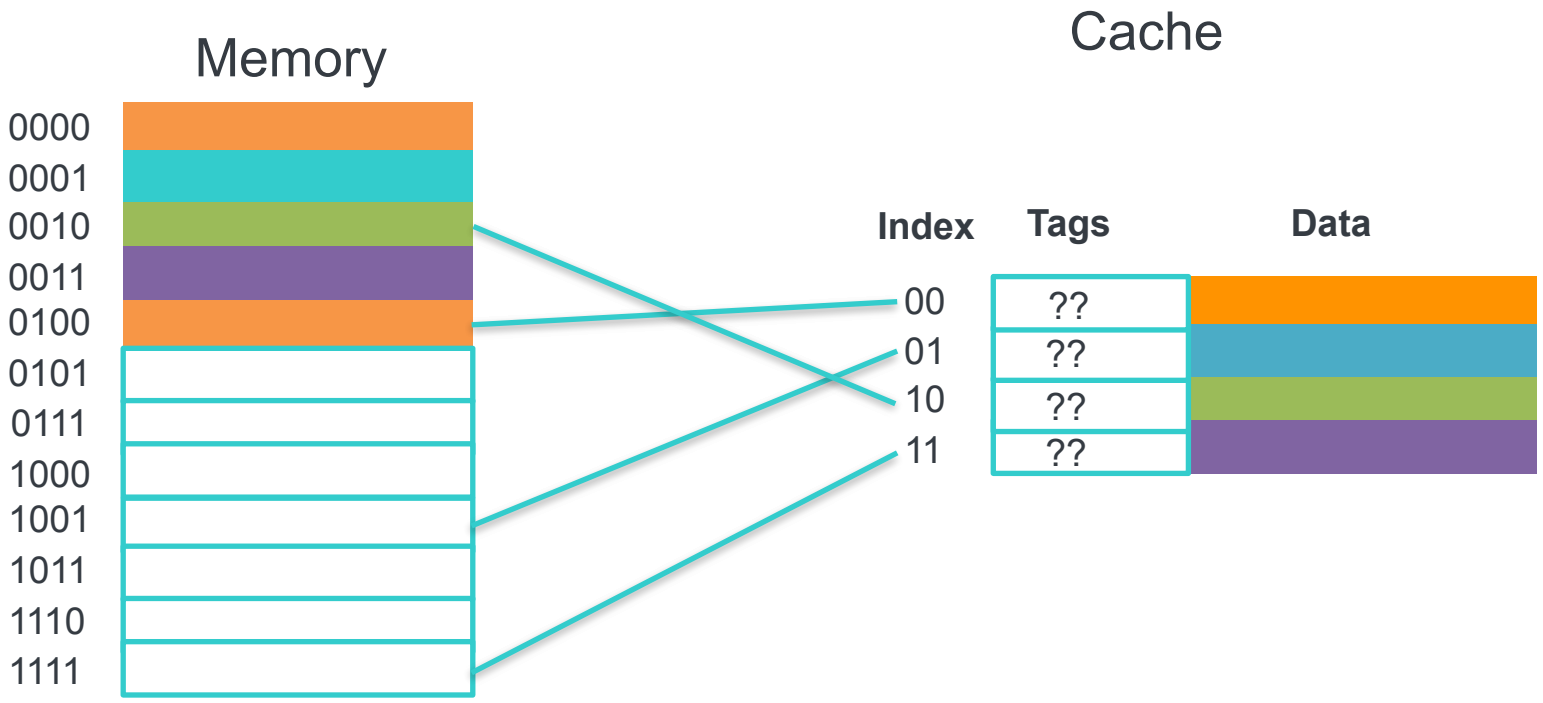
Memory to Cache



Memory to Cache



Memory to Cache

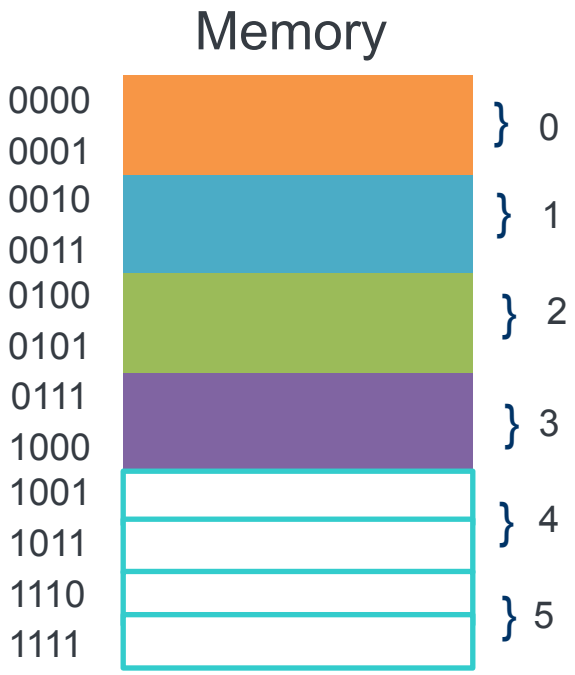


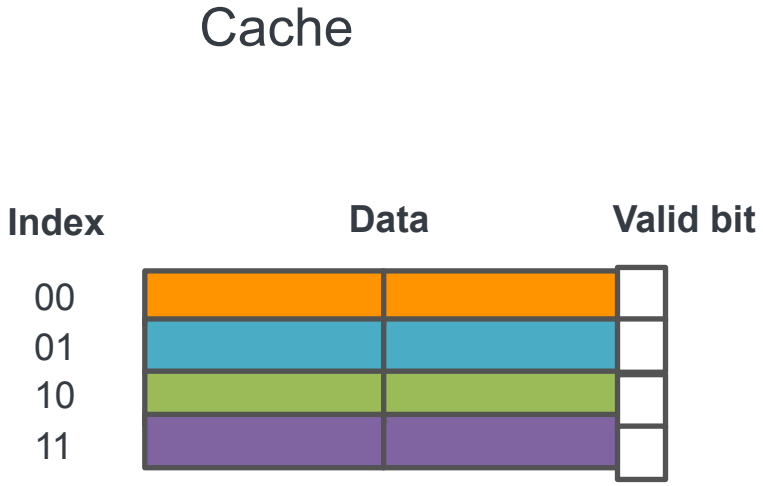
What would happen if the CPU requests memory location 0001 then 1001, then 0001, then 1001?

What is a cache block / cache line?

- Cache Block / Line: unit of transfer between storage and cache memory
- Blocks: between 8 and 64 bytes

Other bits and pieces





Is the data in the cache location an accurate representation of the memory location?

Direct Mapping Exercise

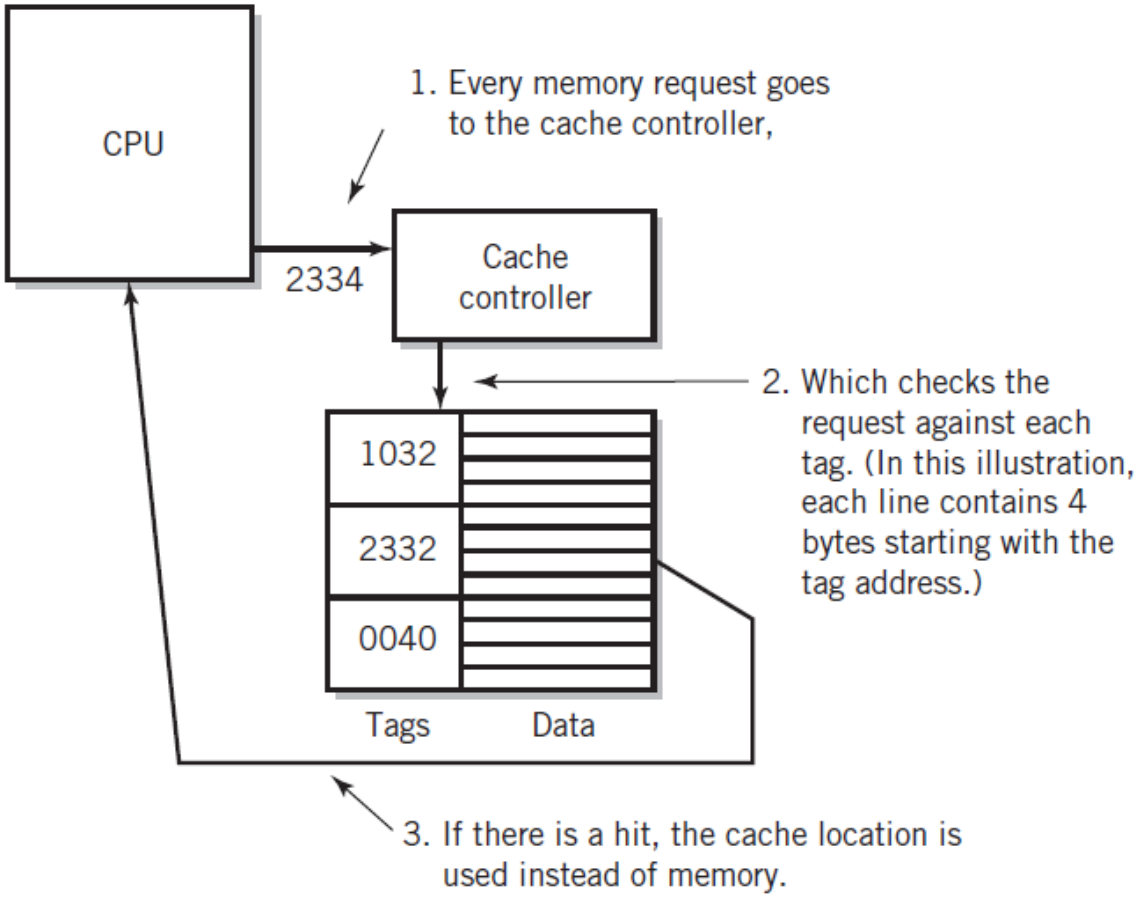
16-bit address space (e.g., 0000-0000-0000-0000)

- Each block is 1 byte
- Cache index is 4 bits
- Direct-mapped

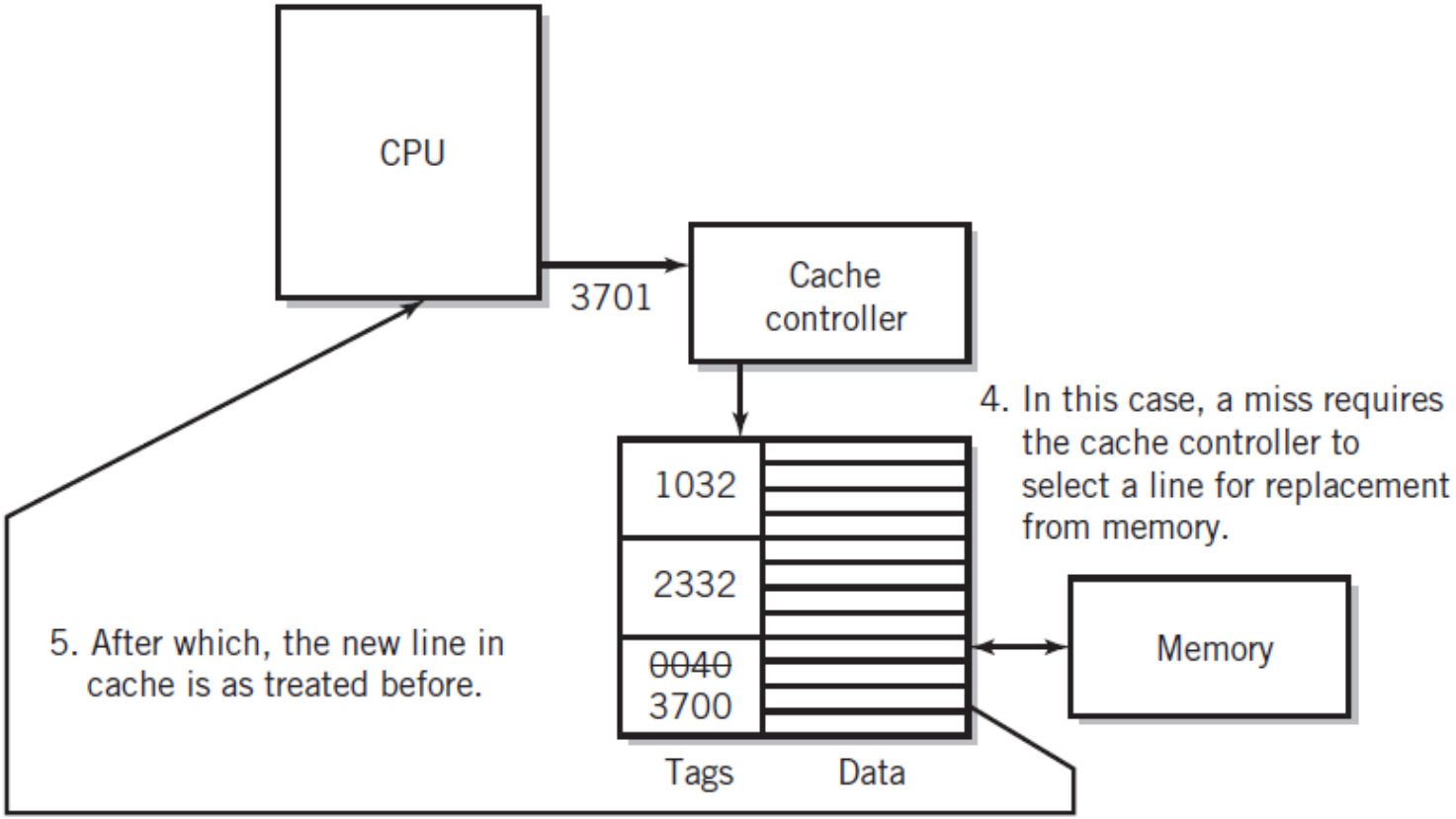
1. How many blocks in cache?

2. How many bits of storage are required to build the cache (e.g., for the data array, tags & valid bit)?

How the cache actually functions?



How the cache actually functions?



Cache design – types of misses

Compulsory

Very first access to a block cannot be in cache

Capacity

If cache not large enough to hold all blocks needed during execution of a program

Conflict

If multiple blocks map to same cache location (if cache not fully associative)

Coherence

Keeping multiple caches coherent in multiprocessor

Cache Simulation

Decimal System

- 100 blocks of RAM
- 10 Blocks in Cache

Spreadsheet on Moodle

- Simulates Memory Address (or Memory Block) requests
- Different levels of locality
  - Uniform
  - Normal (Gaussian) std dev = 25
  - Normal std dev = 150 (poor locality)

So what?

- impact of locality
- problems with Direct Mapping

Block Sequences

Uniform		Good Locality				Poor Locality			
Address	Block	Address	Block	Tag	Cache Bl.	Address	Block	Tag	Cache Bl.
146	14	561	56	5	6	621	62	6	2
139	13	530	53	5	3	440	44	4	4
7	0	624	62	6	2	595	59	5	9
716	71	600	60	6	0	607	60	6	0
826	82	572	57	5	7	270	27	2	7
816	81	556	55	5	5	650	65	6	5
645	64	531	53	5	3	626	62	6	2
311	31	574	57	5	7	831	83	8	3
854	85	543	54	5	4	551	55	5	5
445	44	561	56	5	6	518	51	5	1
172	17	556	55	5	5	633	63	6	3
542	54	562	56	5	6	539	53	5	3
53	5	581	58	5	8	845	84	8	4
605	60	614	61	6	1	649	64	6	4
664	66	596	59	5	9	761	76	7	6
465	46	579	57	5	7	571	57	5	7
35	3	599	59	5	9	710	71	7	1
497	49	585	58	5	8	594	59	5	9
842	84	581	58	5	8	600	60	6	0
970	97	562	56	5	6	555	55	5	5
836	83	575	57	5	7	504	50	5	0

Exercise Identify the Hits (H)

& Misses here

Classify Misses as:

Compulsory (C)

Conflict (F)

	Good Locality			Poor Locality		
	Block	Tag	Cache Bl.	Block	Tag	Cache Bl.
1	58	5	8	70	7	0
2	60	6	0	65	6	5
3	67	6	7	64	6	4
4	59	5	9	54	5	4
5	61	6	1	55	5	5
6	56	5	6	76	7	6
7	51	5	1	67	6	7
8	57	5	7	60	6	0
9	48	4	8	50	5	0
10	63	6	3	58	5	8
11	57	5	7	61	6	1
12	55	5	5	76	7	6
13	64	6	4	38	3	8
14	57	5	7	43	4	3
15	60	6	0	50	5	0
16	60	6	0	67	6	7

Exercise Identify the Hits (H)

& Misses here

Hits (H). 4 &. 3

Conflict (F). 3. &. 5

Compulsory (C) 9 & 8

	Good Locality			Poor Locality		
	Block	Tag	Cache Bl.	Block	Tag	Cache Bl.
1	58	5	8	70	7	0
2	60	6	0	65	6	5
3	67	6	7	64	6	4
4	59	5	9	54	5	4
5	61	6	1	55	5	5
6	56	5	6	76	7	6
7	51	5	1	67	6	7
8	57	5	7	60	6	0
9	48	4	8	50	5	0
10	63	6	3	58	5	8
11	57	5	7	61	6	1
12	55	5	5	76	7	6
13	64	6	4	38	3	8
14	57	5	7	43	4	3
15	60	6	0	50	5	0
16	60	6	0	67	6	7

Where can a block be placed?

Direct Mapped Cache

Each block has only one place where it can be

- if cache has N blocks, location k of memory goes into block (k mod N)

Conflict between memory locations with same (k mod N)

- reduces performance
- some locations overwritten while others can be empty

Fully Associative

Block can be placed anywhere

- Have to look for an empty slot
- Utilizes all slots
- Complex hardware

Set Associative

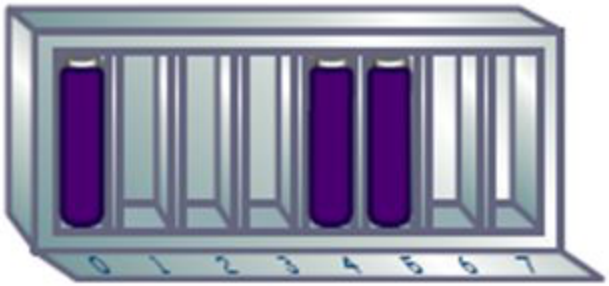
Block can be placed in a restricted set of places

- Have to look for an empty slot
- (block address) mod (number of sets in cache)

Some flexibility, less collisions than direct-mapped

N-way associative – each set contains N slots

Direct Mapped



Tag	Index	Offset

A cache block can only go in one spot in the cache. It makes a cache block very easy to find, but it's not very flexible about where to put the blocks.

2-Way Set Associative

Tag	Index	Offset

This cache is made up of sets that can fit two blocks each. The index is now used to find the set, and the tag helps find the block within the set.

4-Way Set Associative

Tag	Index	Offset

Each set here fits four blocks, so there are fewer sets. As such, fewer index bits are needed.

Fully Associative

Tag	Offset

No index is needed, since a cache block can go anywhere in the cache. Every tag must be compared when finding a block in the cache, but block placement is very flexible!

- Illustration from http://csillustrated.berkeley.edu/PDFs/cache-types.pdf

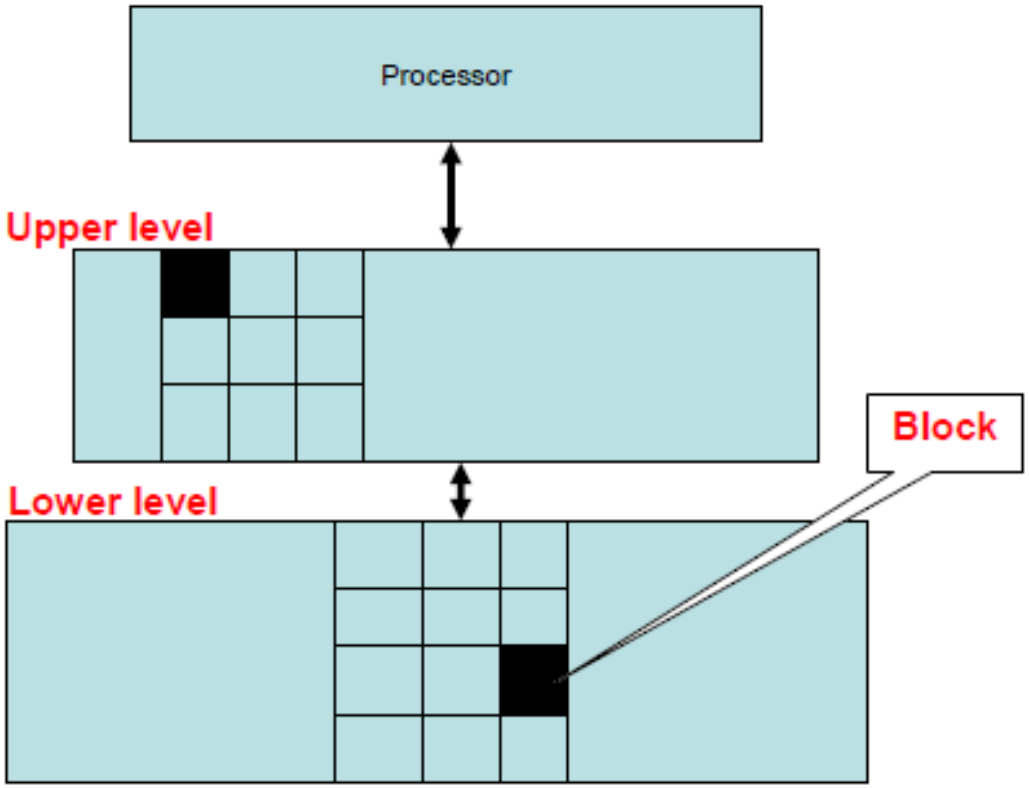
Multiple Levels

Program needs object d, which is stored in some block b

Cache hit - Program finds b in the cache at level k.

Cache miss - b is not at level k, so level k cache must fetch it from level k+1 and store in k

If level k cache is full, then some current block must be replaced.



Which block should be removed to make space for a new one after a miss?

Cache replacement policy

In direct-mapped cache – **no choice**, only one place where it can be placed

For associative

- Random
  - For larger cache size as good as LRU!
- Least recently used (LRU)
- First in first out (FIFO)

What happens on a write?

Most cache accesses are reads

- All instructions accesses are reads
- Most instructions don't write to memory
- Writes about 7% of overall memory traffic, and 28% of data cache traffic
- *Note: cache can be split into data cache and instruction cache)

Write hit and write miss

Write hit strategies

Write-through

Modify the cache, but then also immediately modify main memory

Keeps main memory always up-to-date

Creates a lot of bus traffic

Write-back

Write only to the cache, and mark the cache block “dirty”; main memory updated only when block is replaced

Write miss strategies

If a word being written is not in cache

Write allocate

Block allocated on a write miss
then treat it as a write hit

No-write allocate

Does not affect cache; block only modified in lower-level memory

Cache Optimisations

Reduce the Miss Rate

Larger block size (compulsory misses)
Larger cache size (compulsory misses)
Higher associativity (conflict misses)

Reduce the Miss Penalty

Multi-level cache memory

Reduce Hit Time

Prioritize Reads over Writes

Prefetching

Multiple levels of cache L1, L2, L3

If we know what data we'll need from level-N cache *a priori*, get data from level-(N+1) and place it in level-N cache before the data is accessed by the processor

Considerations

- Shall we prefetch a block?
- Instruction prefetch vs data prefetch

Multiple prefetching schemes

Cache Optimization

1. Where in cache to put new block?
   Cache organization

2. How to identify/find it once it's there?
   Address tag

3. Which block should be removed to make space for a new one after a miss?
   Cache replacement policy

4. What happens when information in a block is modified?
   Write hit and write miss

Cache types

On chip vs off-chip cache

Separate vs unified – instructions and data caches

Exclusive vs inclusive – in multi-level caches is data present in lower level caches or only ever in one

Examples

Intel® Core™ i7-5600U Processor – 2 cores

Level 1 cache: 2 x 32 KB instruction caches, 2 x 32 KB data caches

Level 2 cache: 2 x 256 KB caches

Level 3 cache: 4 MB shared cache

Apple A7 (iphone 5s, ipad air, ipad mini 2/3)

L1 - 64 KB data and 64 KB instructions per-core

L2 – 1 MB shared

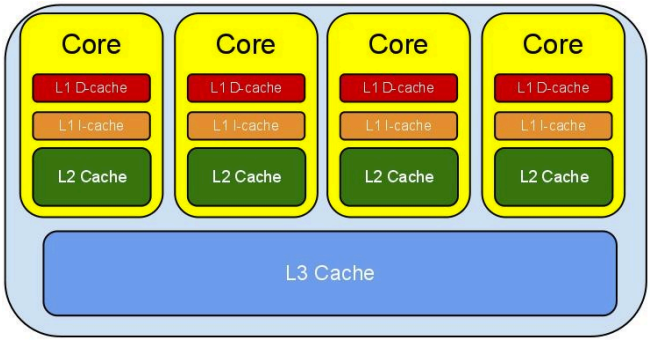
L3 – 4MB shared

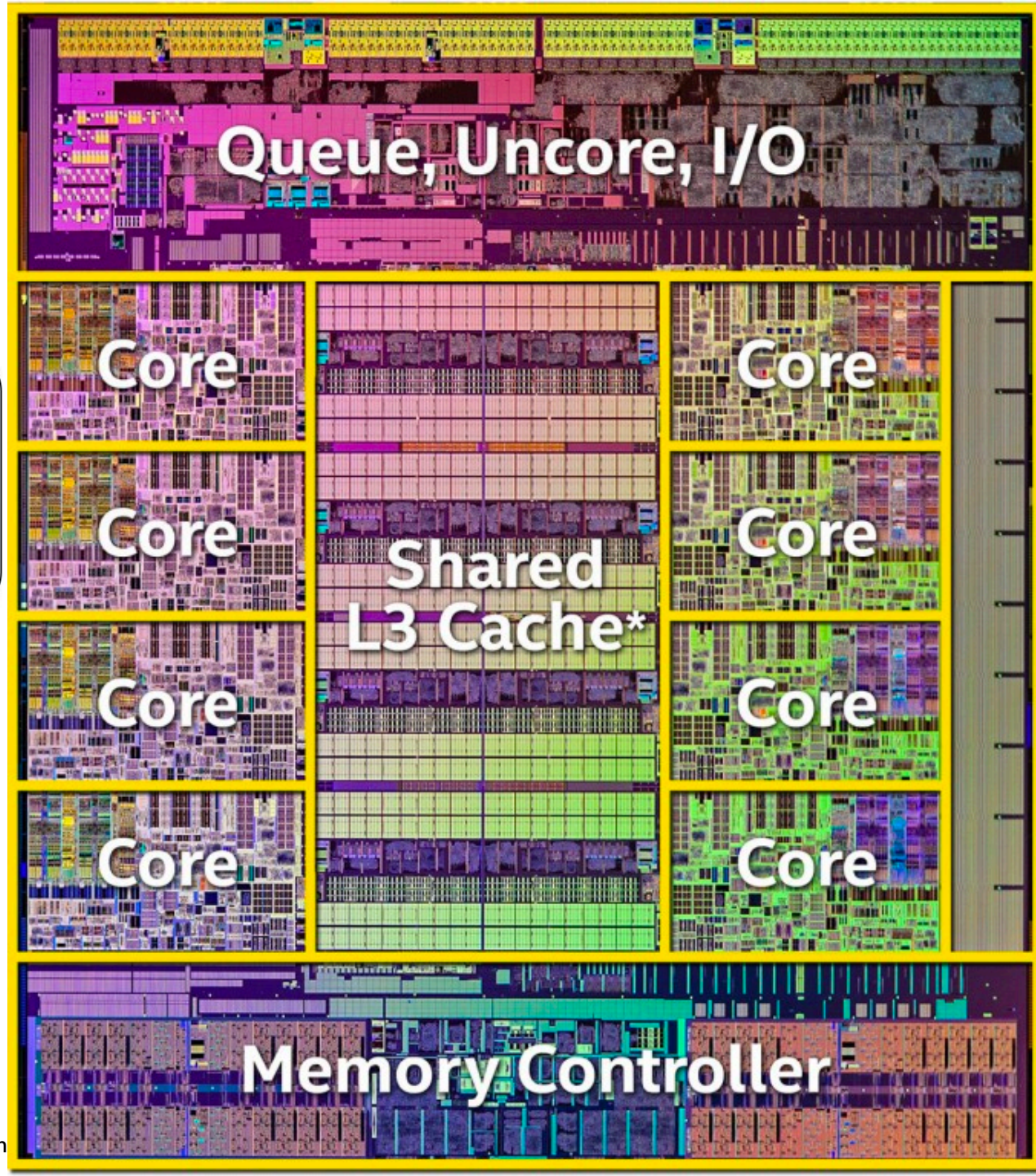
Apple A7 (iphone 5s, ipad air, ipad mini 2/3)

L1 - 64 KB data and 64 KB instructions per-core

L2 – 1 MB shared

L3 – 4MB shared


Core
L1 D-cache
L1 I-cache
L2 Cache
Core
L1 D-cache
L1 I-cache
L2 Cache
Core
L1 D-cache
L1 I-cache
L2 Cache
Core
L1 D-cache
L1 I-cache
L2 Cache
L3 Cache


Queue, Uncore, I/O
Core
Core
Core
Core
Shared
L3 Cache*
Core
Core
Core
Core
Memory Controller

Caches

On completion of this lecture you will be able to:

- Explain the role of the tag, index and offset in Cache mapping
- Discuss the differences between Direct and Associative mapping
- Explain the differences between the different types of cache miss
- Explain what can happen to cached data on a write
- Discuss strategies to improve cache performance