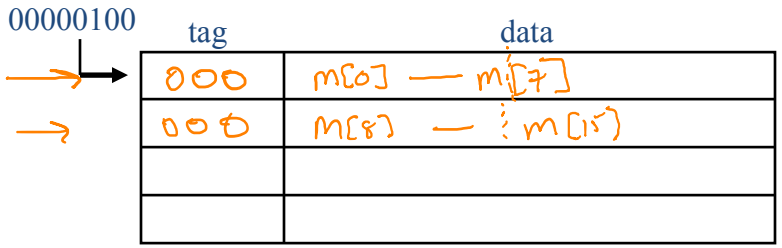
Announcements

- Final Exam
  - Reminder: Will come up quick! The first day after the quarter
  - Logistics:
    - ~50% longer than the midterm
    - Open from 8am-8pm US/Pacific
    - Going to allow 3 hours (less time pressure), still single block of time, forward only
  - Option to re-weight: See Canvas Announcement
- Bonus Participation Quiz "Practice Final"; will release over the weekend
  - Optional…, but many questions will look *a lot* like the final….

Longer Cache Blocks

address	string:
4	00000100
8	00001000
12	00001100
4	00000100
8	00001000
20	00010100
4	00000100
8	00001000
20	00010100
24	00011000
12	00001100
8	00001000
4	00000100

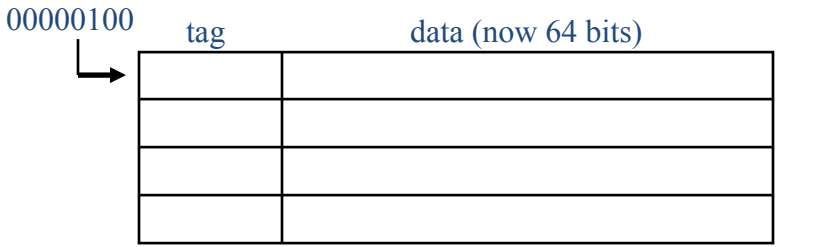


4 entries, each block holds two words, each word in memory maps to exactly one cache location (this cache is twice the total size of the prior caches).

- Large cache blocks take advantage of *spatial locality*.
- Too large of a block size can waste cache space.
- Longer cache blocks require less tag space

Longer Cache Blocks

address string:	
4	00000100
8	00001000
12	00001100
4	00000100
8	00001000
20	00010100
4	00000100
8	00001000
20	00010100
24	00011000
12	00001100
8	00001000
4	00000100



4 entries, each block holds two words, each word in memory maps to exactly one cache location (this cache is twice the total size of the prior caches).

- Large cache blocks take advantage of ***spatial locality***.
- Too large of a block size can waste cache space.
- Longer cache blocks require less tag space

Q: Describing Cache Type Tradeoffs?

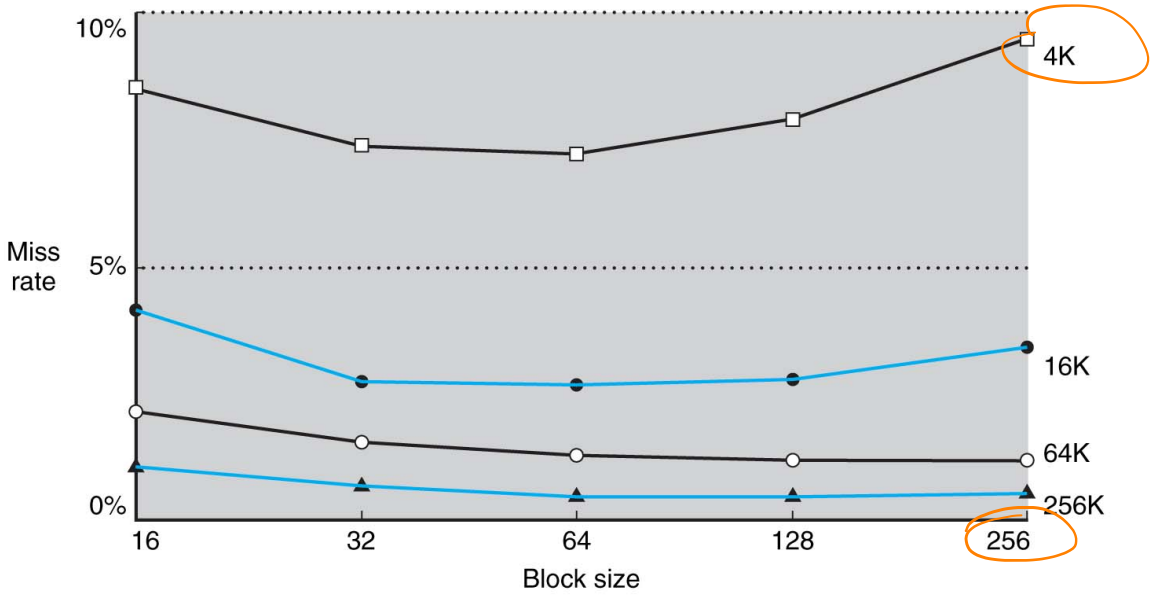
1. Exceptional usage of the cache space in exchange for a slow hit time
2. Poor usage of the cache space in exchange for an excellent hit time
3. Reasonable usage of cache space in exchange for a reasonable hit time

Selection	Fully-Associative	4-way Set Associative	Direct Mapped
A	3	2	1
B	1	3	2
C	1	2	3
D	3	2	1
E	None of the above		

Back to Block Size

- If block size increases spatial locality, should we just make the cache block size really, really big????

Block Size and Miss Rate



Cache Parameters

Cache size = Number of sets * block size * associativity

Warning / Notice—Things that count towards "cache size": cache **data**

Things that **do not count** towards "cache size": tags, valid bits, etc…

Cache Parameters

Cache size = Number of sets * block size * associativity

- 128 blocks, 32-byte block size, direct mapped, size = ? 4 KB

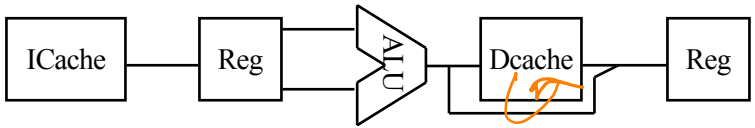
- 128 KB cache, 64-byte blocks, 512 sets, associativity = ?

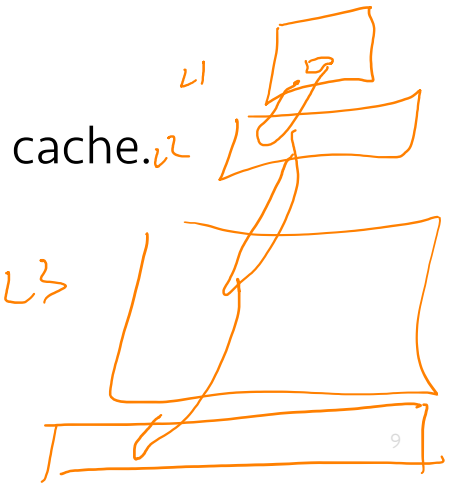
$2^{17} / 2^6 \quad 2^9 \quad 2^2 = 4$-way

(always keep in mind "cache size" only counts the data storage)

Handling a Cache Access

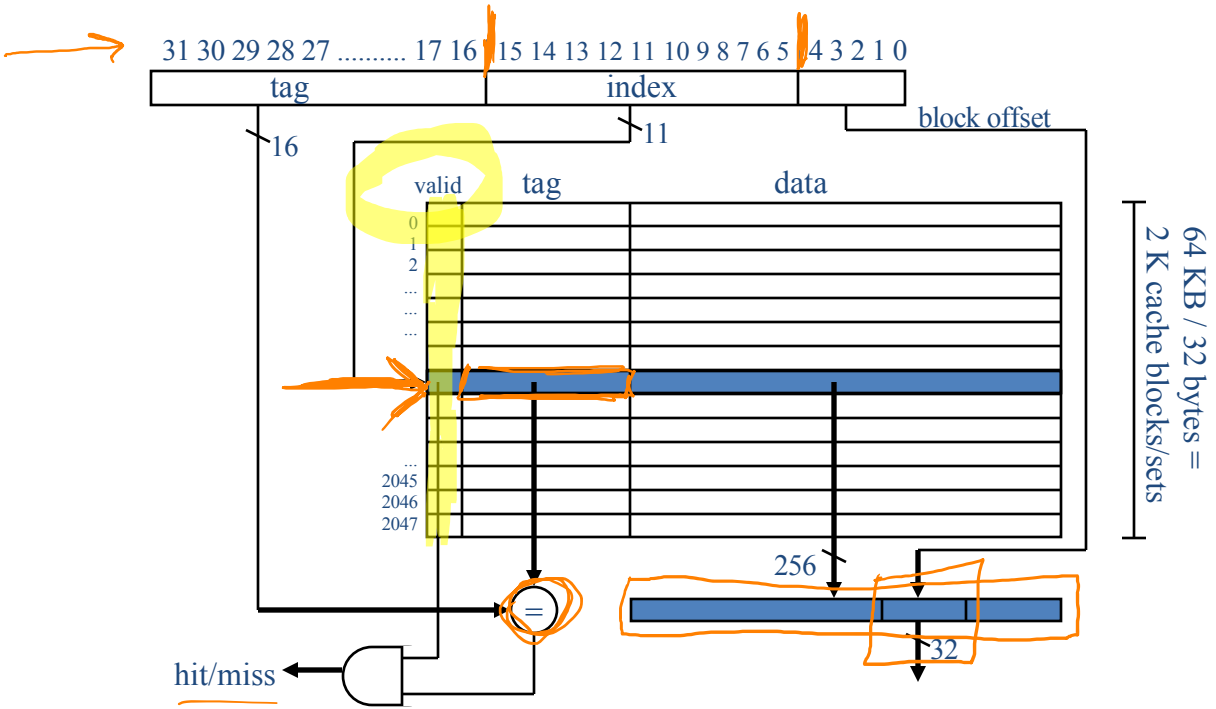
- 1. Use index and tag to access cache and determine hit/miss.
- 2. If hit, return requested data.
- 3. If miss, select a cache block to be replaced, and access memory or next lower cache (possibly stalling the processor).
  - load entire missed cache line into cache
  - return requested data to CPU (or higher cache)
- 4. If next lower memory is a cache, goto step 1 for that cache.





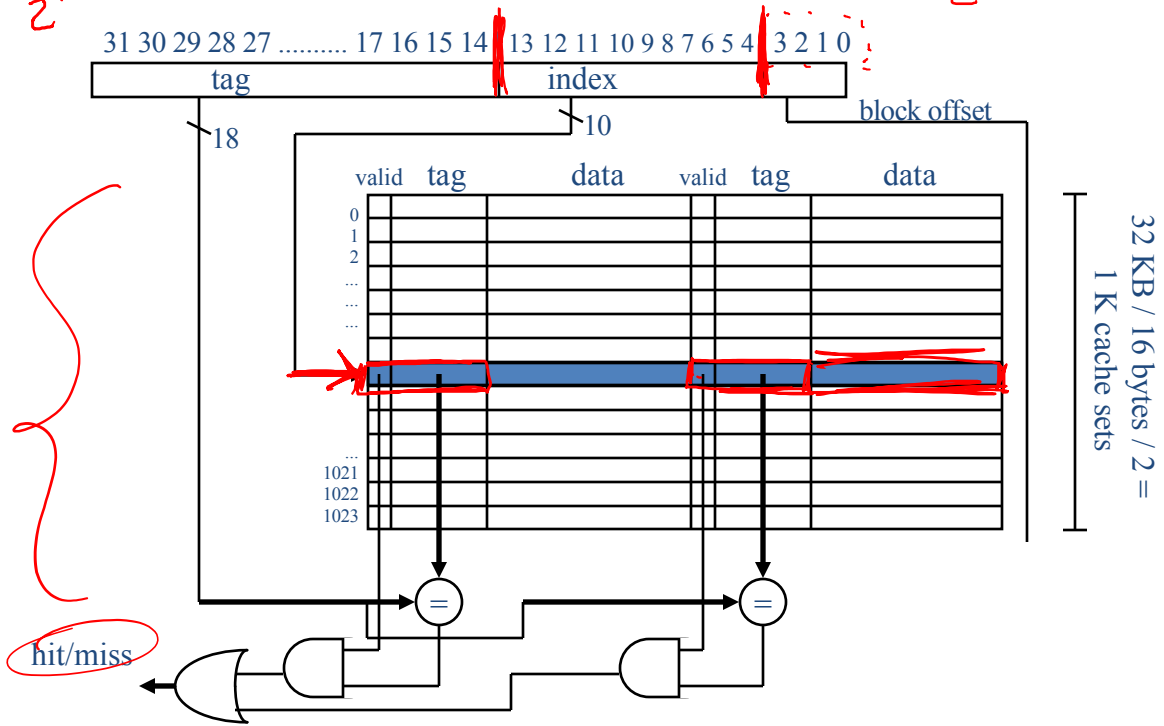
Accessing a Sample Cache

- 64 KB cache, direct-mapped, 32-byte cache block size



Accessing a Sample Cache

- 32 KB cache, 2-way set-associative, 16-byte block size

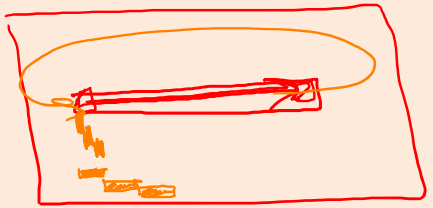


Associative Caches

- Higher hit rates, but…

- longer access time
  - (longer to determine hit/miss, more muxing of outputs)
- more space (longer tags)
  - 16 KB, 16-byte blocks, DM, tag = ?
  - 16 KB, 16-byte blocks, 4-way, tag = ?

```
for (int i = 0; i < 10,000,000; i++)
    sum += A[i];
```



Assume each element of A is 4 bytes and sum is kept in a register.
Assume a baseline direct-mapped 32KB L1 cache with 32 byte blocks.
Assume this loop is visited many times.
Which changes would help the hit rate of the above code?

Selection	Change
A	~~Increase to 2-way set associativity~~
B	Increase block size to 64 bytes
C	~~Increase cache size to 64 KB~~
D	A and C combined
E	A, B, and C combined

```
for (int i=0; i < 10,000,000; i++)
    for (int j = 0; j < 8192; j++)
        sum += A[j] - B[j];
```

Assume each element of A and B are 4 bytes.

Assume each array is at least 32KB in size.

Assume sum is kept in a register.

Assume a baseline direct-mapped 32KB L1 cache with 32 byte blocks.

Which changes would help the hit rate of the above code?

Selection	Change
A	Increase to 2-way set associativity
B	Increase block size to 64 bytes
C	Increase cache size to 64 KB
D	A and C combined
E	A, B, and C combined