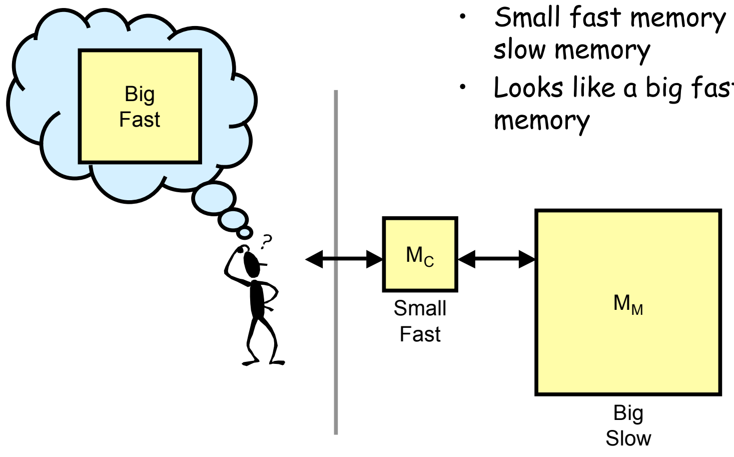
Lecture 15: Improving Cache Performance

- Last Time:
  - Cache introduction
    - Average Memory Access Time (AMAT)
    - Set associativity
- Today
  - Replacement and write policies
  - Cache performance optimizations

Cache Memory

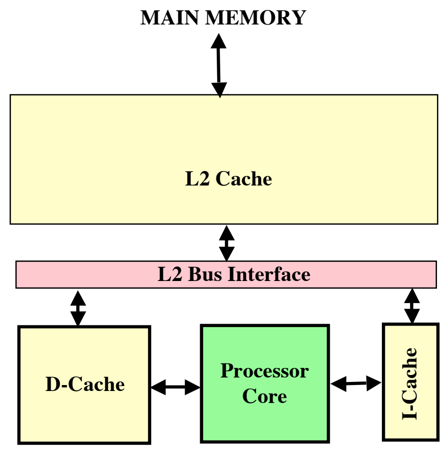


- Small fast memory + big slow memory
- Looks like a big fast memory

Bookshelf analogy

- Lots of books on shelves
- A few books on my desk
- One book that I'm reading

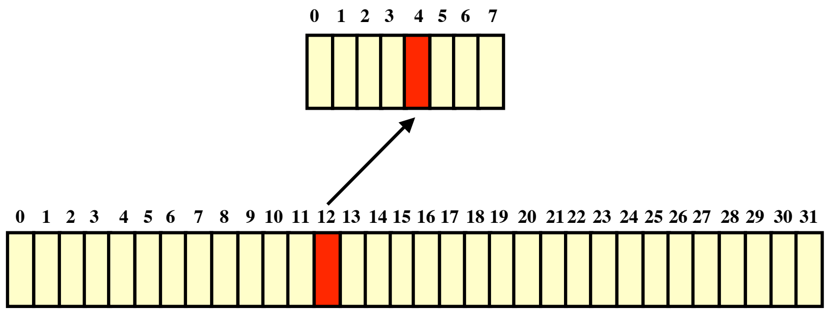
Typical Cache Organization



Direct Mapped

- Each block mapped to exactly 1 cache location

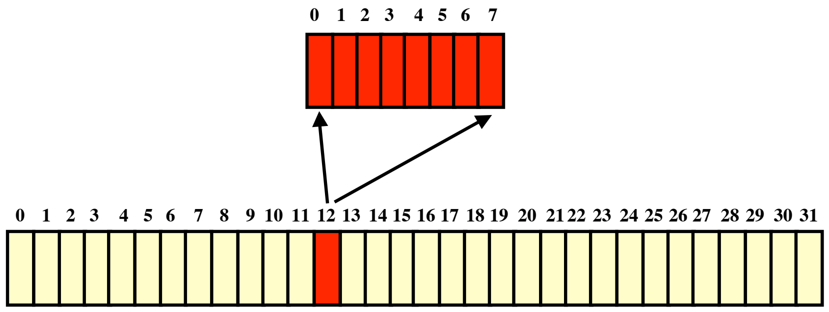
Cache location = (block address) MOD (# blocks in cache)



Fully Associative

- Each block mapped to any cache location

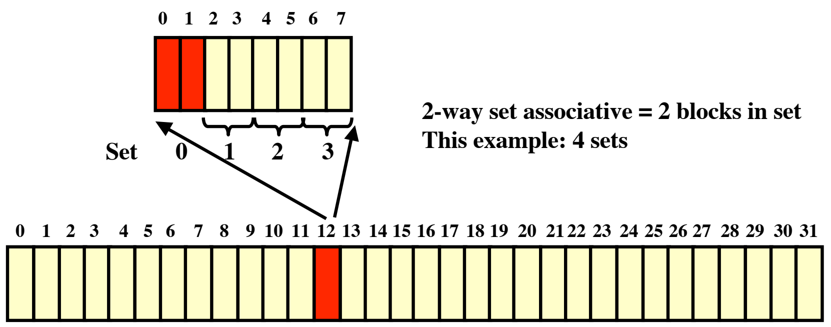
Cache location = any



Set Associative

- **Each block mapped to subset of cache locations**

Set selection = (block address) MOD (# sets in cache)

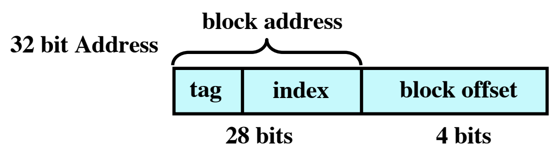


2-way set associative = 2 blocks in set
This example: 4 sets

How do we use memory address to find block in the cache?

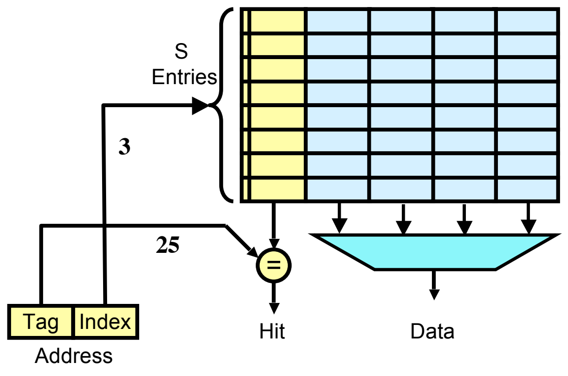
How Do We Find a Block in The Cache?

- Our Example:
  - Main memory address space = 32 bits (= 4GBytes)
  - Block size = 4 words = 16 bytes
  - Cache capacity = 8 blocks = 128 bytes
-



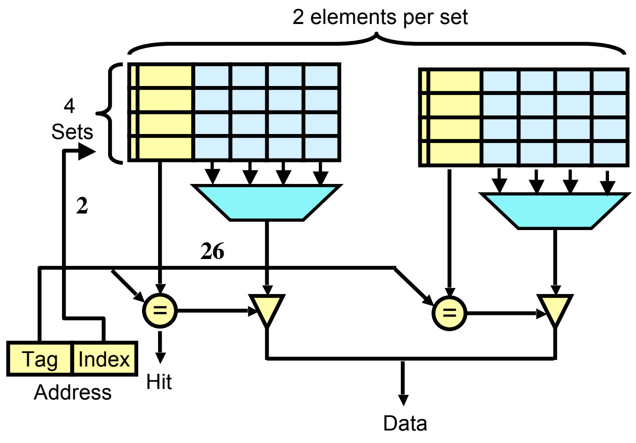
- index ⇒ which set
- tag ⇒ which data/instruction in block
- block offset ⇒ which word in block
- # tag/index bits determine the associativity
- tag/index bits can come from anywhere in block address

Finding a Block: Direct-Mapped



With cache capacity = 8 blocks

Finding A Block: 2-Way Set-Associative



UTCS
CS352, S07

Lecture 15

11

Set Associative Cache

- S - sets
- A - elements in each set
  - A-way associative
- In the example, S=4, A=2
  - 2-way associative 8-entry cache

UTCS
CS352, S07

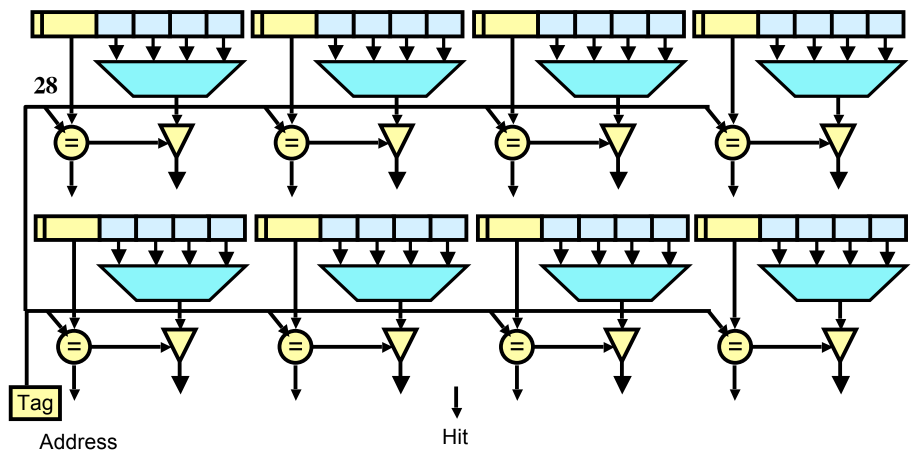
Lecture 15

12

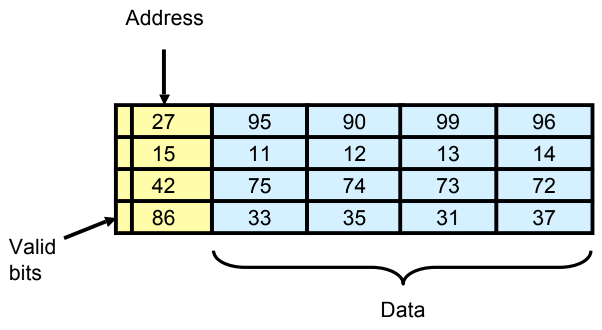
Set Associative Cache - cont'd

- All of main memory is divided into S sets
  - All addresses in set N map to same set of the cache
    - Addr = N mod S
    - A locations available
- Shares costly comparators across sets
- Low address bits select set
  - 2 in example
- High address bits are *tag*, used to associatively search the selected set
- Extreme cases
  - A=1: Direct mapped cache
  - S=1: Fully associative
- A need not be a power of 2

Finding A Block: Fully Associative



What is the purpose of the valid bit?



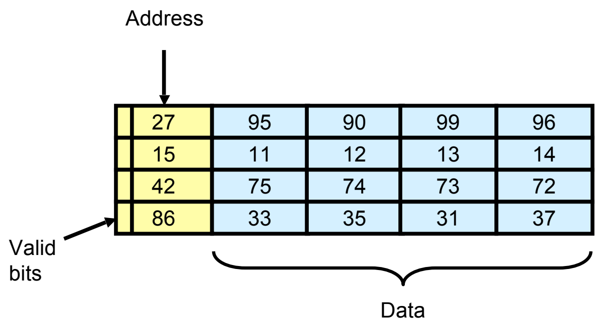
	27	95	90	99	96
	15	11	12	13	14
	42	75	74	73	72
	86	33	35	31	37

- **Hint: what happens when the machine boots up?**

Questions to think about

- As the block size goes up, what happens to the miss rate?
- … what happens to the miss penalty?
- … what happens to hit time?
- As the associativity goes up, what happens to the miss rate?
- … what happens to the hit time?

Cache Organization



	27	95	90	99	96
	15	11	12	13	14
	42	75	74	73	72
	86	33	35	31	37

- **Where does a block get placed? - DONE**
- **How do we find it? - DONE**
- **Which one do we replace when a new one is brought in?**
- **What happens on a write?**

Which Block Should Be Replaced on Miss?

- Direct Mapped
  - Choice is easy - only one option
- Associative
  - Randomly select block in set to replace
  - Least-Recently used (LRU)
- Implementing LRU
  - 2-way set-associative
  - >2 way set-associative

What Happens on a Store?

- Need to keep cache consistent with main memory
  - Reads are easy - no modifications
  - Writes are harder - when do we update main memory?
- Write-Through
  - On cache write - always update main memory as well
  - Use a write buffer to stockpile writes to main memory for speed
- Write-Back
  - On cache write - remember that block is modified (dirty bit)
  - Update main memory when dirty block is replaced
  - Sometimes need to flush cache (I/O, multiprocessing)

BUT: What if Store Causes Miss!

- Write-Allocate
  - Bring written block into cache
  - Update word in block
  - Anticipate further use of block
- No-write Allocate
  - Main memory is updated
  - Cache contents unmodified

Improving cache performance

Three kinds of cache misses

- Compulsory misses
  - First time data is accessed
- Capacity misses
  - Working set larger than cache size
- Conflict misses
  - One set fills up, but room in other sets

How Do We Improve Cache Performance?

$$AMAT = t_{hit} + p_{miss} \bullet penalty_{miss}$$

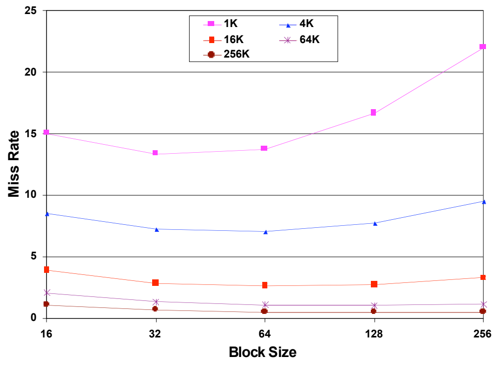
How Do We Improve Cache Performance?

$$AMAT = t_{hit} + p_{miss} \bullet penalty_{miss}$$

- Reduce miss rate
- Reduce miss penalty
- Reduce hit time

Reducing Miss Rate: Increase Block Size

- Fetch more data with each cache miss
  - 16 bytes ⇒ 64, 128, 256 bytes!
  - Works because of Locality (spatial)

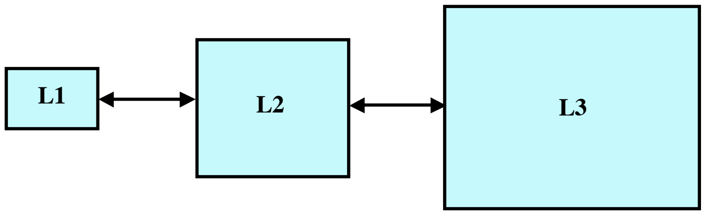


Reducing Miss Rate: Increase Associativity

- Reduce conflict misses
- Rules of thumb
  - 8-way = fully associative
  - Direct mapped size N = 2-way set associative size N/2
- But!
  - Size N associative is larger than Size N direct mapped
  - Associative typically slower that direct mapped (t_{hit} larger)

Reduce Miss Penalty: More Cache Levels

- Average access time =
  $HitTime_{L1} + MissRate_{L1} * MissPenalty_{L1}$
- $MissPenalty_{L1}$ =
  $HitTime_{L2} + MissRate_{L2} * MissPenalty_{L2}$
- etc.
- Size/Associativity of higher level caches?



Reduce Miss Penalty: Transfer Time

- Wider path to memory
  - Transfer more bytes/cycle
  - Reduces total time to transfer block
- Two ways to do this:
  - Wider path to each memory
  - Separate paths to multiple memories ("multiple memory banks")

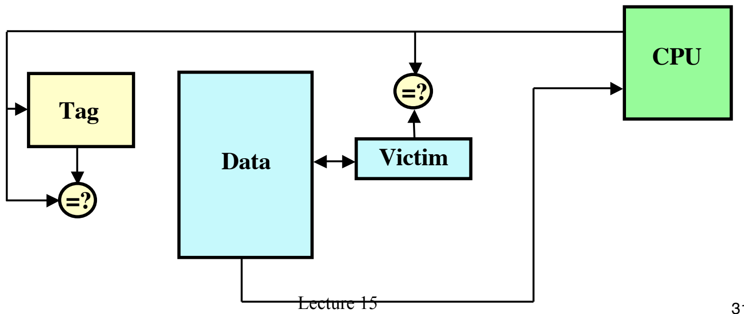
Reducing Hit Time

- Make Caches small and simple
  - Hit Time = 1 cycle is good (3.3ns!)
  - L1 - low associativity, relatively small
- Even L2 caches can be broken into sub-banks
  - Can exploit this for faster hit time in L2

Reducing Miss Rate: Use a "Victim" Cache

- Small cache (< 8 entries)
  - Jouppi 1990
  - Accessed in parallel with main cache
  - Captures conflict misses

Set	1W	2W	3W	4W	
0	X				
1	X	X			
2	X				
3	X	X	X	X	X
4	X				
5	X				
6					
7	X	X			

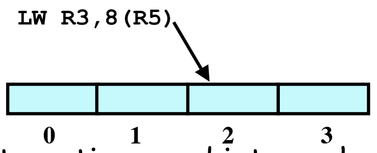


Reducing Miss Rate: Prefetching

- Fetching Data that you will probably need
- Instructions
  - Alpha 21064 on cache miss
    - Fetches requested block intro instruction stream buffer
    - Fetches next sequential block into cache
- Data
  - Automatically fetch data into cache (spatial locality)
  - Issues?
- Compiler controlled prefetching
  - Inserts prefetching instructions to fetch data for later use
  - Registers or cache

Reduce Miss Penalty: Deliver Critical word first

- Only need one word from block immediately

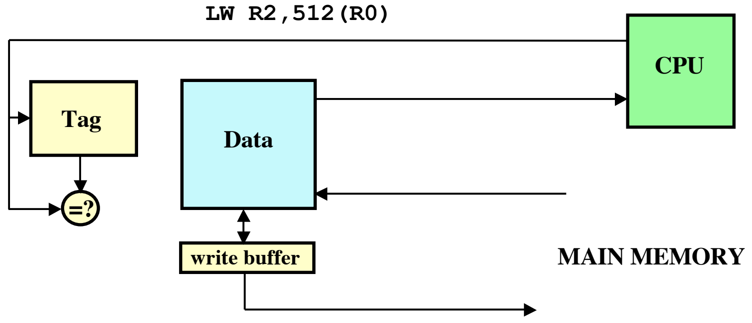


- Don't write entire word into cache first
  - Fetch word 2 first (deliver to CPU)
  - Fetch order: 2 3 0 1

Reduce Miss Penalty: Read Misses First

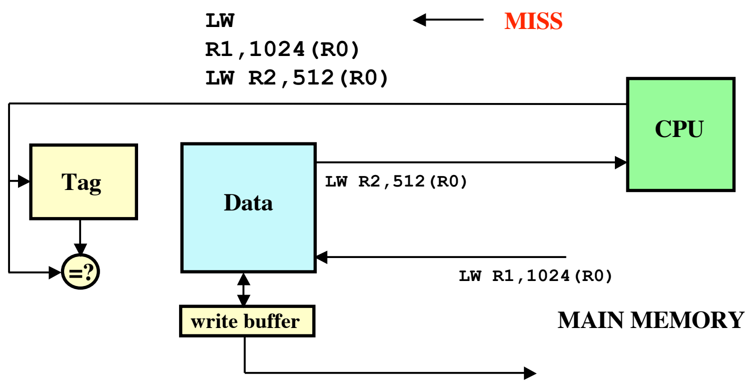
- Let reads pass writes in Write buffer

```
SW 512(R0),R3
LW R1,1024(R0)
LW R2,512(R0)
```



Reduce Miss Penalty: Lockup Free Cache

- Let cache continue to function while miss is being serviced



Summary

- Recap
  - Using a memory address to find location in cache
  - Deciding what to evict from the cache
  - Improving cache performance
- Next time:
  - Memory system and Virtual Memory
  - Read P&H 7.4 - 7.9