CS 354 - Machine Organization & Programming
Thursday, October 19, 2017

Midterm Exam - Thursday October 26th, 7:15 - 9:15 pm
- **Lec 1 (1 PM):** room 6210 of Social Sciences
- **Lec 2 (11 AM):** room B10 of Ingraham Hall
- UW ID required
- closed book, no notes, no calculators
- see "Exam Information and Policies" on course site

Project p3 (6%): DUE at 10 pm on Wednesday, November 1st

Homework hw4 (1.5%): DUE at 10 pm on Tuesday, October 24th

Last Time

Locality
Memory Hierarchy
Cache Basic Idea
Designing a Cache - Blocks
Designing a Cache - Sets and Tags
Designing a Cache - Lines
Basic Cache Operation

Today

Basic Cache Practice
Improving Direct Mapped Cache
Set Associative Cache
Replacement Policies
Fully Associative Cache

Next Time

Cache Performance and Coding Considerations
Read: B&O 6.4.3 - 6.4.7, 6.5 - 6.7

Basic Cache Practice

You are given the following 32-bit address layout used by a cache:

32-bit Address Layout

bit 31   24   16   8   0

t bits   s bits   b bits
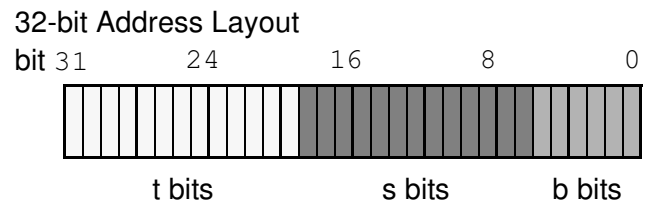
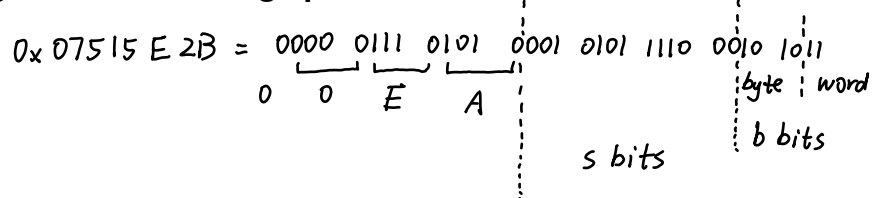→ How big are the blocks (bytes)?

$$B = 2^b = 64 \text{ bytes}$$

→ How many sets?

$$S = 2^s = 2^{13} = 2^3 K = 8k$$

→ If each set has one line, how big is the cache?

$$S \times B = 2^{13} \times 2^6 = 2^{19} = \tfrac{1}{2} M = 512\,k$$

You are given the following specific address used in this cache: `0x07515E2B`

$$0x\,07515\,E\,2B = 0000\ 0111\ 0101\ \underbrace{0001\ 0101\ 1110\ 0010\ 1011}$$

$$\underbrace{0}_{0}\ \underbrace{0}_{0}\ \underbrace{E}_{E}\ \underbrace{A}_{A}$$

s bits   byte | word   b bits

→ Which word is accessed in its block (base 10)? Which byte in this word (base 10)?

word 10

byte 3

→ What is its set number (base 10)?

$$1024 + 256 + 64 + 32 + 16 + 8 = 1400$$

→ For each cache line partially shown below, does the address produce a hit or miss?

1.) `1 0x0750` (block of bytes not shown) *Miss*
2.) `1 0x00EA` (block of bytes not shown) *Hit*
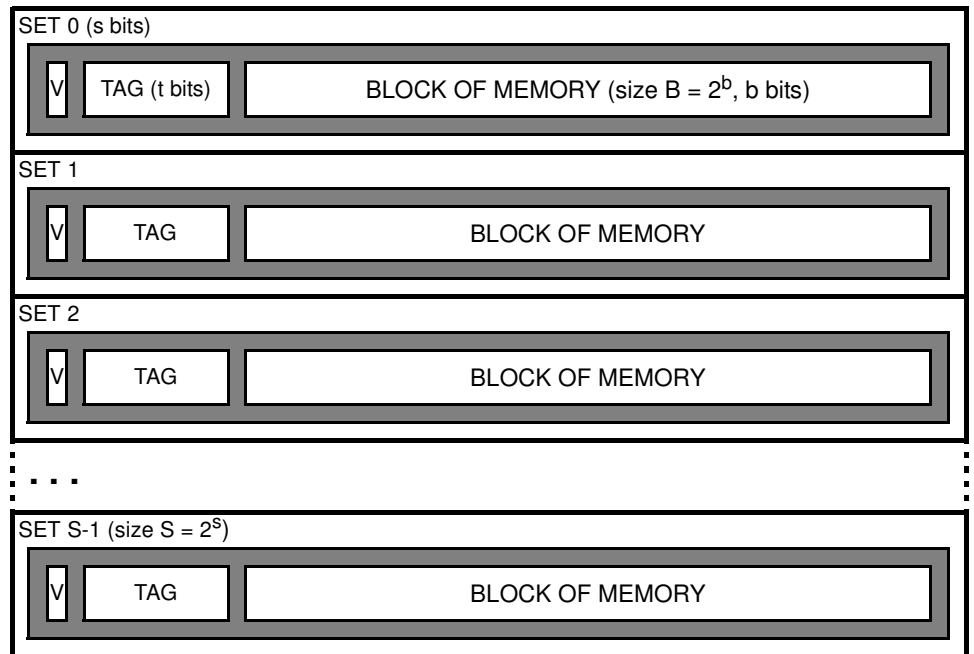3.) `0 0x00EA` (block of bytes not shown) *Miss, line is invalid*

Improving Direct Mapped Cache

Direct Mapped Cache

is a cache where each memory block maps to one set and
each cache set has exactly one line.

+ no need to search
to find if desired is
cached

SET 0 (s bits)
V	TAG (t bits)	BLOCK OF MEMORY (size B = 2^b, b bits)

SET 1
V	TAG	BLOCK OF MEMORY

+ simple circuitry is
required to match
the tag

SET 2
V	TAG	BLOCK OF MEMORY

. . .

SET S-1 (size S = 2^s)
V	TAG	BLOCK OF MEMORY

→ What happens when two different memory blocks map to the same set?

conflict miss - a blocks collide when mapping the same set

→ Improvements?

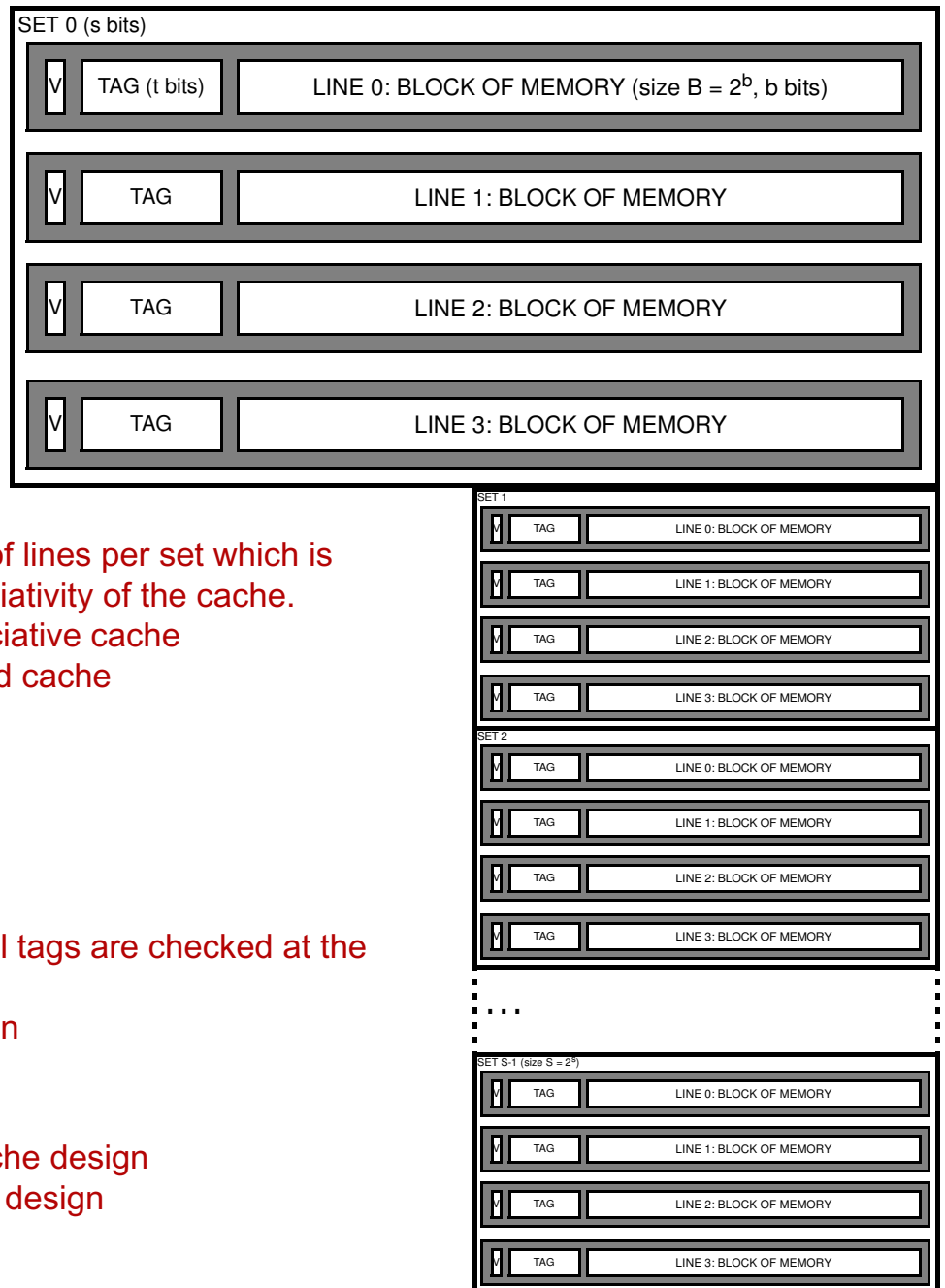enable a set to store more blocks by adding more lines each set.

Set Associative Cache

Set Associative Cache

is a cache where each block maps to one set and each
set has two or more lines

+ reduces
conflict misses

SET 0 (s bits)

V | TAG (t bits) | LINE 0: BLOCK OF MEMORY (size B = 2^b, b bits)

V | TAG | LINE 1: BLOCK OF MEMORY

V | TAG | LINE 2: BLOCK OF MEMORY

V | TAG | LINE 3: BLOCK OF MEMORY

- more complicated
circuitry is needed
to match tags

SET 1

V | TAG | LINE 0: BLOCK OF MEMORY
V | TAG | LINE 1: BLOCK OF MEMORY
V | TAG | LINE 2: BLOCK OF MEMORY
V | TAG | LINE 3: BLOCK OF MEMORY

let E be the number of lines per set which is
considered the associativity of the cache.
E=4: 4-way set associative cache
E=1: directed mapped cache

SET 2

V | TAG | LINE 0: BLOCK OF MEMORY
V | TAG | LINE 1: BLOCK OF MEMORY
V | TAG | LINE 2: BLOCK OF MEMORY
V | TAG | LINE 3: BLOCK OF MEMORY

. . .

Operation

1. set extraction
2. line matching but all tags are checked at the same time
3. if hit, word extraction

SET S-1 (size S = 2^s)

V | TAG | LINE 0: BLOCK OF MEMORY
V | TAG | LINE 1: BLOCK OF MEMORY
V | TAG | LINE 2: BLOCK OF MEMORY
V | TAG | LINE 3: BLOCK OF MEMORY

C = (S, E, B, m)

characterizes a cache design
Let C be the cache design
C size is S * B * E

→ How big is a cache given (1024, 4, 32, 32)?

Replacement Policies

Assume the following sequence of memory blocks

are fetched into <u>the same set</u> of a 4-way associative cache:
b1, b2, b3, b1, b3, b4, b4, b7, b1, b8, b4, b9, b1, b9, b9, b2, b8, b1

~~b1~~ ~~b2~~ ~~b3~~ ~~b4~~
~~b7~~ b1 b8 b2
b9

1. *Random Replacement*

→ Which of the following would be possible outcomes after the sequence finishes?
Note cache lines are listed left to right instead of top to bottom.

```
b9  b1  b8  b2   Yes
b1  b2  --  b8   Empty lines are used before replacing
b1  b4  b7  b3   OYO
b1  b2  b8  b1   Duplicates won't occur rather it would result in a hit
```

2. *Least Recently Used* (LRU)

Need to track when a line was last used
Done by using a LRU queue: move most recent to queue's front

→ Which blocks will remain in the cache after the sequence finishes?

b1 b8 b2 b9

3. *Least Frequently Used* (LFU)

Need to track how often each line is used
Use a counter that is incremented on each
use of a line

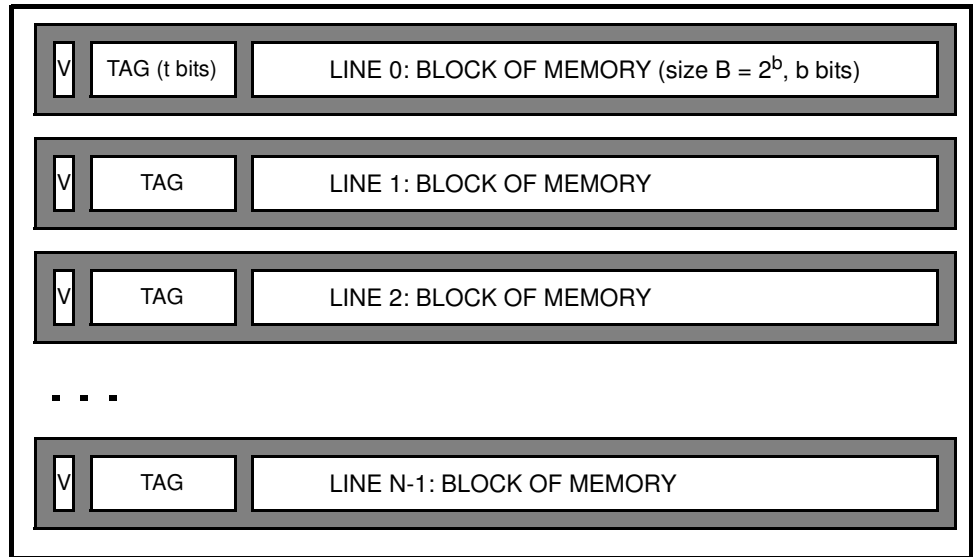→ What is the outcome after the sequence finishes?

1 4 8 9

※ Exploiting replacement policies to improve code
performance isn't easy so programmers don't
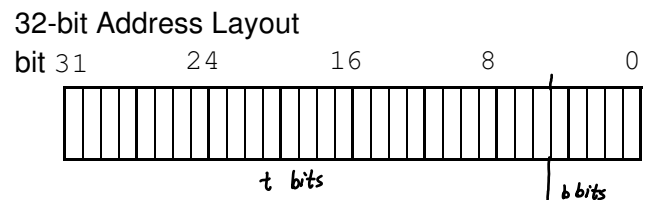
Fully Associative Cache

Fully Associative Cache

    is a cache having only one set!
               so that blocks can be stored
               anywhere in the cache (in any line)

- must search entire cache to determine if desired block is cached.

- expensive and complicated circuitry required to do line matching

V	TAG (t bits)	LINE 0: BLOCK OF MEMORY (size B = 2^b, b bits)
V	TAG	LINE 1: BLOCK OF MEMORY
V	TAG	LINE 2: BLOCK OF MEMORY

. . .

V	TAG	LINE N-1: BLOCK OF MEMORY

→ What is the address layout if blocks are 32 bytes?

32-bit Address Layout

bit 31   24   16   8   0

t bits   b bits

→ How many lines should a fully associative cache have?

    First determine the cache size
    Then block size and compute
    C/B=E

➢ Why isn't it possible for E > C/B?

    OYO

❊ appropriate only for small caches (Fast)

Writing to a Cache

Write Hits   writing to a block that is in the cache

→ When should a block in lower cache levels/main memory be updated to reflect writes?
Since the same memory block can be copied in multiple
cache levels, keeping the data consistent is a concern.

  1. Write Through   update lower levels immediately

   + simple
   - slow if you must wait for lower levels to update
   - more bus traffic

  2. Write Back

Write Misses   writing to a block isn't in the cache

→ Should we allocate space in this cache for the block that has a write miss?

  1. No Write Allocate

  2. Write Allocate

Typical Designs

  1. Write through paired with no write allocate
  2. Write back paired with write allocate

  → Which best exploits locality?