CS 354 - Machine Organization & Programming
Thursday, October 24, 2019

Project p3 (6%): DUE at 10 pm on Monday, October 28th
Homework hw4 (1.5%): DUE at 10 pm on Thursday, October 31st

Last Time

Caching Basic Idea
Designing a Cache - Blocks
Designing a Cache - Sets and Tags
Basic Cache Lines
Basic Cache Operation
Basic Cache Practice
Direct Mapped Cache

Today

Set Associative Cache (from last time)
Replacement Policies
Fully Associative Cache
Writing to Caches
Cache Performance Metrics
Cache Parameters and Performance

Next Time

Finish Caching
Assembly Language
Read: B&O 3 Intro, 3.1 - 3.4

Replacement Policies

Assume the following sequence of memory blocks

are fetched into the same set of a 4-way associative cache that is initially empty:
`b1, b2, b3, b1, b3, b4, b4, b7, b1, b8, b4, b9, b1, b9, b9, b2, b8, b1`

1. *Random Replacement*

→ Which of the following four outcomes is possible after the sequence finishes?
Assume the initial placement is random.

```
   L0 L1 L2 L3
1. b9 b1 b8 b2

2. b1 b2 -- b8

3. b1 b4 b7 b3

4. b1 b2 b8 b1
```

2. *Least Recently Used* (LRU)

→ What is the outcome after the sequence finishes?
Assume the initial placement is in ascending line order (left to right below).

L0 L1 L2 L3

3. *Least Frequently Used* (LFU)

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

✵ *Exploiting replacement policies*

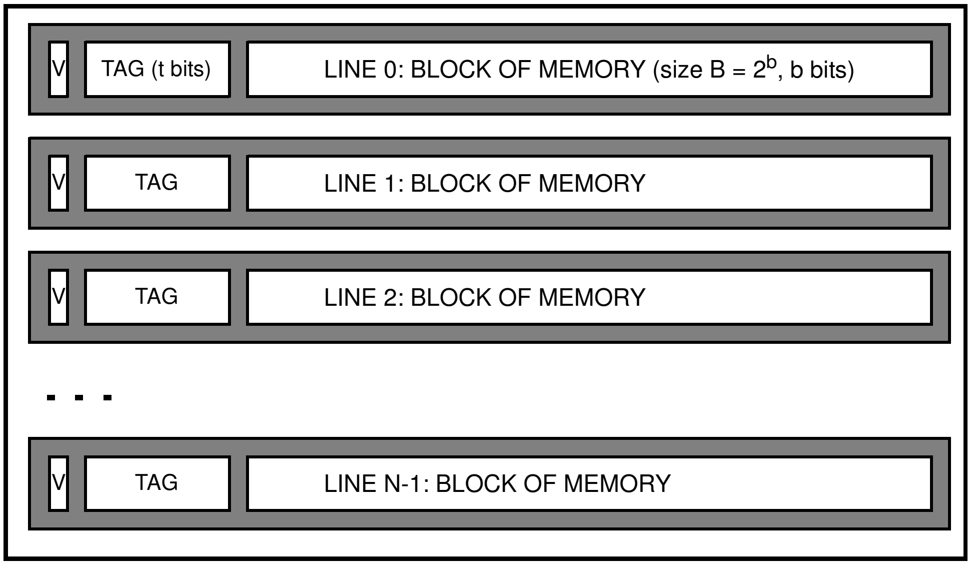
Fully Associative Cache

Fully Associative Cache

is a cache

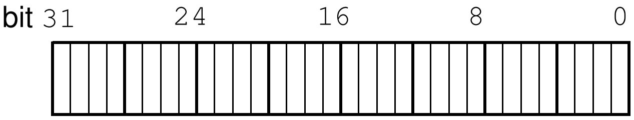
-

-



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

32-bit Address Breakdown



→ How many lines should a fully associative cache have?

➢ Why isn't it possible for E > C/B?

✵

Writing to a Cache

- ✵ *Reading data copies of*
- ✵ *Writing data requires that*

Write Hits

occur when writing to a block

→ When should a block be updated in lower memory levels?

1. *Write Through*:

2. *Write Back*:

Write Misses

occur when writing to a block

→ Should space be allocated in this cache for the block being changed?

1. *No Write Allocate*:

2. *Write Allocate*:

Typical Designs

1. Write Through paired with
2. Write Back paired with

→ Which best exploits locality?

Cache Performance Metrics

<u>Hit Rate</u>

<u>Hit Time</u>

<u>Miss Penalty</u>

L1 hit
L1 miss served from L2
L1 miss served from L3

L1 miss served from MM

Cache Parameters and Performance

Larger Blocks (S and E unchanged)

hit rate

hit time

miss penalty

THEREFORE

More Sets (B and E unchanged)

hit rate

hit time

miss penalty

THEREFORE

More Lines E per Set (B and S unchanged)

hit rate

hit time

miss penalty

THEREFORE

Intel Quad Core i7 Cache (gen 7)

all: 64 byte block, use pseudo LRU, write back

L1: 32KB, 4-way Instruction & 32KB 8-way Data, no write allocate
L2: 256KB, 8-way, write allocate
L3: 8MB, 16-way (2MB/Core shared), write allocate