# MEMORY SYSTEM

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

# Overview

- Notes
  - Homework 10 is April 18th
    - Verify your submitted file before midnight
- This lecture
  - Cache miss types
  - Virtual memory

# Cache Miss Classifications

- Start by measuring miss rate with an ideal cache
  - 1. ideal is fully associative and infinite capacity
  - 2. then reduce capacity to size of interest
  - 3. then reduce associativity to degree of interest

# Cache Miss Classifications

- Start by measuring miss rate with an ideal cache
  - 1. ideal is fully associative and infinite capacity
  - 2. then reduce capacity to size of interest
  - 3. then reduce associativity to degree of interest

1. Cold (compulsory)

# Cache Miss Classifications

- Start by measuring miss rate with an ideal cache
  - 1. ideal is fully associative and infinite capacity
  - 2. then reduce capacity to size of interest
  - 3. then reduce associativity to degree of interest

1. Cold (compulsory)

- Cold start: first access to block
- How to improve
  - large blocks
  - prefetching

# Cache Miss Classifications

- Start by measuring miss rate with an ideal cache
  - 1. ideal is fully associative and infinite capacity
  - 2. then reduce capacity to size of interest
  - 3. then reduce associativity to degree of interest

1. Cold (compulsory)

2. Capacity

- Cold start: first access to block
- How to improve
  - large blocks
  - prefetching

# Cache Miss Classifications

- Start by measuring miss rate with an ideal cache
  - 1. ideal is fully associative and infinite capacity
  - 2. then reduce capacity to size of interest
  - 3. then reduce associativity to degree of interest

| 1. Cold (compulsory) | 2. Capacity |
| --- | --- |
| ❑ Cold start: first access to block | ❑ Cache is smaller than the program data |
| ❑ How to improve | ❑ How to improve |
| ○ large blocks | ○ large cache |
| ○ prefetching | |

# Cache Miss Classifications

- Start by measuring miss rate with an ideal cache
  - 1. ideal is fully associative and infinite capacity
  - 2. then reduce capacity to size of interest
  - 3. then reduce associativity to degree of interest

| 1. Cold (compulsory) | 2. Capacity | 3. Conflict |
|---|---|---|

- Cold start: first access to block
- How to improve
  - large blocks
  - prefetching

- Cache is smaller than the program data
- How to improve
  - large cache

# Cache Miss Classifications

- Start by measuring miss rate with an ideal cache
  - 1. ideal is fully associative and infinite capacity
  - 2. then reduce capacity to size of interest
  - 3. then reduce associativity to degree of interest

| 1. Cold (compulsory) | 2. Capacity | 3. Conflict |
|---|---|---|
| ❑ Cold start: first access to block<br>❑ How to improve<br>○ large blocks<br>○ prefetching | ❑ Cache is smaller than the program data<br>❑ How to improve<br>○ large cache | ❑ Set size is smaller than mapped mem. locations<br>❑ How to improve<br>○ large cache<br>○ more assoc. |

# Miss Rates: Example Problem

- 100,000 loads and stores are generated; L1 cache has 3,000 misses; L2 cache has 1,500 misses. What are various miss rates?

# Miss Rates: Example Problem

- 100,000 loads and stores are generated; L1 cache has 3,000 misses; L2 cache has 1,500 misses. What are various miss rates?

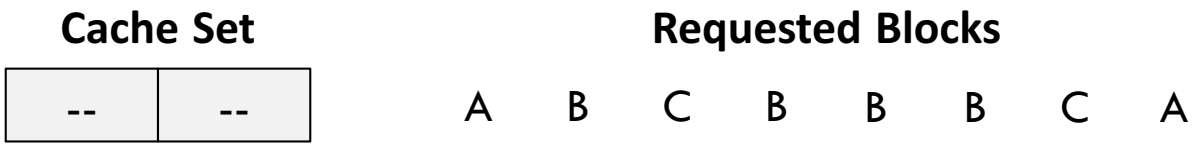
- L1 miss rates
  - 3,000/100,000 = 3%
- L2 miss rates
  - 1,500/3,000 = 50%

# Cache Replacement Policies

- Which block to replace on a miss?
  - Only one candidate in direct-mapped cache
  - Multiple candidates in set/fully associative cache

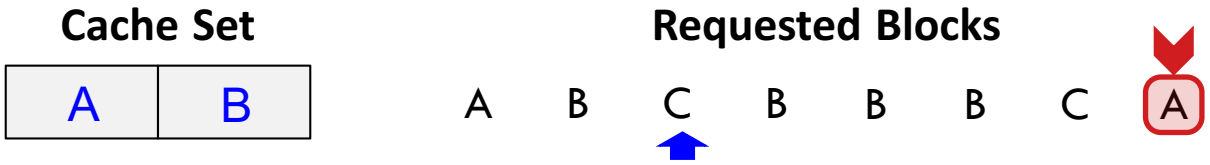
# Cache Replacement Policies

- Which block to replace on a miss?
  - Only one candidate in direct-mapped cache
  - Multiple candidates in set/fully associative cache
- Ideal replacement (Belady's algorithm)
  - Replace the block accessed farthest in the future



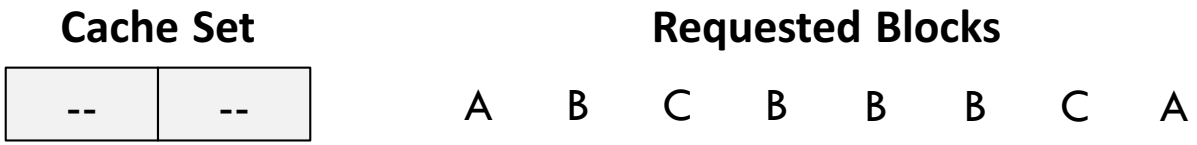
# Cache Replacement Policies

- Which block to replace on a miss?
  - Only one candidate in direct-mapped cache
  - Multiple candidates in set/fully associative cache
- Ideal replacement (Belady's algorithm)
  - Replace the block accessed farthest in the future



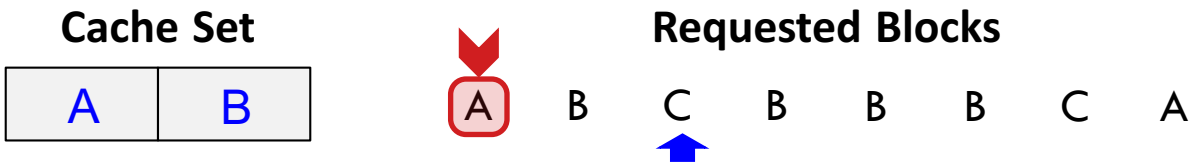
# Cache Replacement Policies

- Which block to replace on a miss?
  - Only one candidate in direct-mapped cache
  - Multiple candidates in set/fully associative cache
- Ideal replacement (Belady's algorithm)
  - Replace the block accessed farthest in the future
- Least recently used (LRU)
  - Replace the block accessed farthest in the past



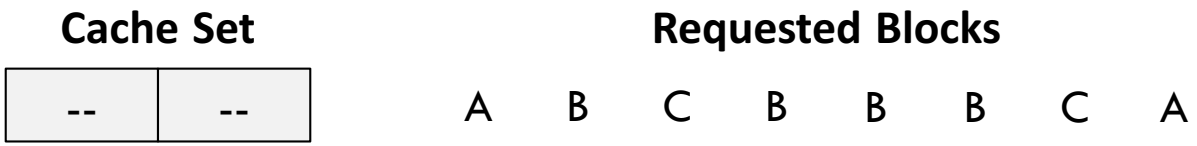
# Cache Replacement Policies

- Which block to replace on a miss?
  - Only one candidate in direct-mapped cache
  - Multiple candidates in set/fully associative cache
- Ideal replacement (Belady's algorithm)
  - Replace the block accessed farthest in the future
- Least recently used (LRU)
  - Replace the block accessed farthest in the past



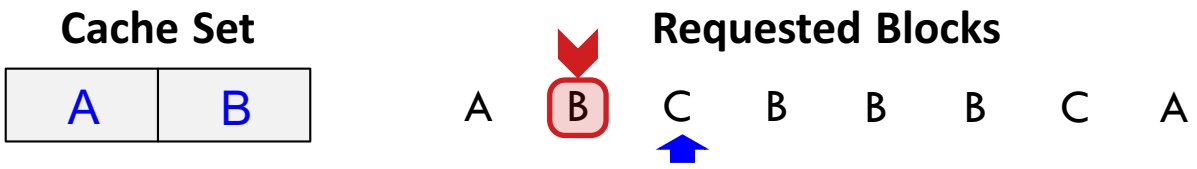
# Cache Replacement Policies

- Which block to replace on a miss?
  - Only one candidate in direct-mapped cache
  - Multiple candidates in set/fully associative cache
- Ideal replacement (Belady's algorithm)
  - Replace the block accessed farthest in the future
- Least recently used (LRU)
  - Replace the block accessed farthest in the past
- Most recently used (MRU)
  - Replace the block accessed nearest in the past



# Cache Replacement Policies

- Which block to replace on a miss?
  - Only one candidate in direct-mapped cache
  - Multiple candidates in set/fully associative cache
- Ideal replacement (Belady's algorithm)
  - Replace the block accessed farthest in the future
- Least recently used (LRU)
  - Replace the block accessed farthest in the past
- Most recently used (MRU)
  - Replace the block accessed nearest in the past



# Cache Replacement Policies

- Which block to replace on a miss?
  - Only one candidate in direct-mapped cache
  - Multiple candidates in set/fully associative cache
- Ideal replacement (Belady's algorithm)
  - Replace the block accessed farthest in the future
- Least recently used (LRU)
  - Replace the block accessed farthest in the past
- Most recently used (MRU)
  - Replace the block accessed nearest in the past
- Random replacement
  - hardware randomly selects a cache block to replace

# Example Problem

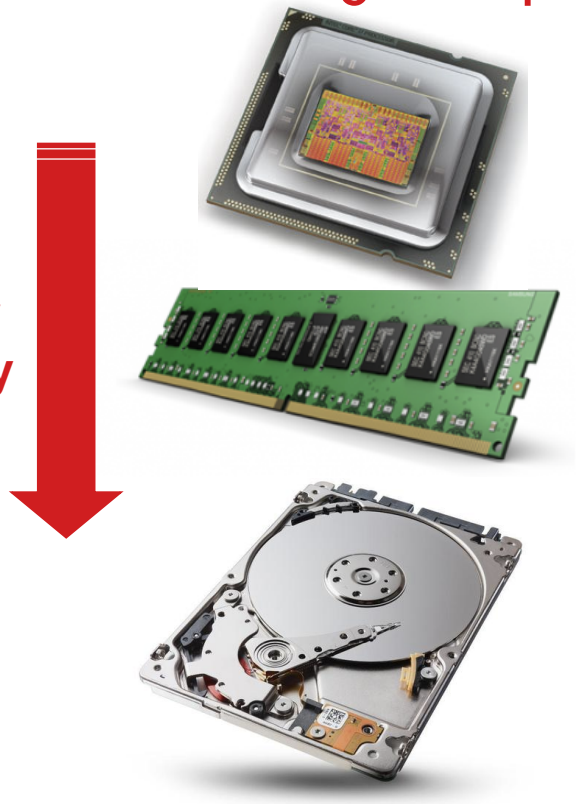
- Blocks A, B, and C are mapped to a single set with only two block storages; find the miss rates for LRU and MRU policies.
- 1. A, B, C, A, B, C, A, B, C
- 2. A, A, B, B, C, C, A, B, C

# Example Problem

- Blocks A, B, and C are mapped to a single set with only two block storages; find the miss rates for LRU and MRU policies.
- 1. A, B, C, A, B, C, A, B, C
  - LRU : 100%
  - MRU : 66%
- 2. A, A, B, B, C, C, A, B, C
  - LRU : 66%
  - MRU : 44%

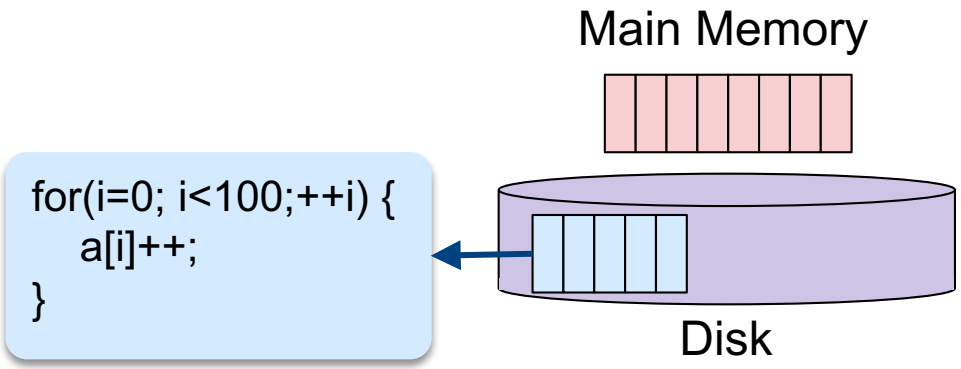
# Memory Hierarchy

- Lower levels provide greater capacity longer time
  - Does the program fit in main memory?
  - What if running multiple programs?



# Virtual Memory

- Use the main memory as a "cache" for secondary memory
  - Placement policy?



# Virtual Memory

- Use the main memory as a "cache" for secondary memory
  - Placement policy?
- Allow efficient and safe sharing the physical main memory among multiple programs
  - Replacement policy?