CACHE ARCHITECTURE

Mahdi Nazm Bojnordi

Assistant Professor
School of Computing
University of Utah
Overview

- Announcement
  - Homework 3 will be released on Oct. 31st

- This lecture
  - Cache addressing and lookup
  - Cache optimizations
    - Techniques to improve miss rate
    - Replacement policies
    - Write policies
Recall: Cache Addressing

- Instead of specifying cache address we specify main memory address
- Simplest: direct-mapped cache

Note: each memory address maps to a single cache location determined by modulo hashing

How to exactly specify which blocks are in the cache?
Direct-Mapped Lookup

- Byte offset: to select the requested byte
- Tag: to maintain the address
- Valid flag (v): whether content is meaningful
- Data and tag are always accessed
Example Problem

Find the size of tag, index, and offset bits for an 8MB, direct-mapped L3 cache with 64B cache blocks. Assume that the processor can address up to 4GB of main memory.
Example Problem

- Find the size of tag, index, and offset bits for an 8MB, direct-mapped L3 cache with 64B cache blocks. Assume that the processor can address up to 4GB of main memory.
- \(4\text{GB} = 2^{32} \text{ B} \Rightarrow \text{address bits} = 32\)
- \(64\text{B} = 2^6 \text{ B} \Rightarrow \text{byte offset bits} = 6\)
- \(8\text{MB}/64\text{B} = 2^{17} \Rightarrow \text{index bits} = 17\)
- \(\text{tag bits} = 32 - 6 - 17 = 9\)
Example Problem

- Find the size of tag, index, and offset bits for an 8MB, direct-mapped L3 cache with 64B cache blocks. Assume that the processor can address up to 4GB of main memory.
Example Problem

- Find the size of tag, index, and offset bits for an 8MB, direct-mapped L3 cache with 64B cache blocks. Assume that the processor can address up to 4GB of main memory.

- \( 4\text{GB} = 2^{32} \text{ B} \rightarrow \text{address bits} = 32 \)

- \( 64\text{B} = 2^{6} \text{ B} \rightarrow \text{byte offset bits} = 6 \)

- \( 8\text{MB}/64\text{B} = 2^{17} \rightarrow \text{index bits} = 17 \)

- \( \text{tag bits} = 32 - 6 - 17 = 9 \)
Cache Optimizations

- How to improve cache performance?
  \[ AMAT = t_h + r_m t_p \]

- Reduce hit time \((t_h)\)

- Improve hit rate \((1 - r_m)\)

- Reduce miss penalty \((t_p)\)
Cache Optimizations

- How to improve cache performance?
  \[ AMAT = t_h + r_m \cdot t_p \]

- Reduce hit time \( (t_h) \)
  - Memory technology, critical access path

- Improve hit rate \( (1 - r_m) \)

- Reduce miss penalty \( (t_p) \)
How to improve cache performance?

\[ AMAT = t_h + r_m \cdot t_p \]

- Reduce hit time \((t_h)\)
  - Memory technology, critical access path
- Improve hit rate \((1 - r_m)\)
  - Size, associativity, placement/replacement policies
- Reduce miss penalty \((t_p)\)
Cache Optimizations

- How to improve cache performance?

\[ AMAT = t_h + r_m t_p \]

- Reduce hit time \((t_h)\)
  - Memory technology, critical access path

- Improve hit rate \((1 - r_m)\)
  - Size, associativity, placement/replacement policies

- Reduce miss penalty \((t_p)\)
  - Multi level caches, data prefetching
Set Associative Caches

- Improve cache hit rate by allowing a memory location to be placed in more than one cache block
  - N-way set associative cache
  - Fully associative
- For fixed capacity, higher associativity typically leads to higher hit rates
  - more places to simultaneously map cache lines
  - 8-way SA close to FA in practice

```c
for (i=0; i<10000; i++) {
    a++;
    b++;
}
```
Set Associative Caches

- Improve cache hit rate by allowing a memory location to be placed in more than one cache block
  - N-way set associative cache
  - Fully associative
- For fixed capacity, higher associativity typically leads to higher hit rates
  - more places to simultaneously map cache lines
  - 8-way SA close to FA in practice

```cpp
for (i=0; i<10000; i++) {
    a++;
    b++;
}
```
n-Way Set Associative Lookup

- Index into cache sets
- Multiple tag comparisons
- Multiple data reads
- Special cases
  - Direct mapped
    - Single block sets
  - Fully associative
    - Single set cache
Find the size of tag, index, and offset bits for an 4MB, 4-way set associative cache with 32B cache blocks. Assume that the processor can address up to 4GB of main memory.
Example Problem

Find the size of tag, index, and offset bits for an 4MB, 4-way set associative cache with 32B cache blocks. Assume that the processor can address up to 4GB of main memory.

- $4\text{GB} = 2^{32} \text{ B} \Rightarrow \text{address bits} = 32$
- $32\text{B} = 2^{5} \text{ B} \Rightarrow \text{byte offset bits} = 5$
- $4\text{MB} / (4 \times 32\text{B}) = 2^{15} \Rightarrow \text{index bits} = 15$
- tag bits = $32 - 5 - 15 = 12$
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

- 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)

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

2. Capacity

- 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

<table>
<thead>
<tr>
<th>1. Cold (compulsory)</th>
<th>2. Capacity</th>
<th>3. Conflict</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold start: first access to block</td>
<td>Cache is smaller than the program data</td>
<td>How to improve</td>
</tr>
<tr>
<td>How to improve</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o large blocks</td>
<td>o large cache</td>
<td></td>
</tr>
<tr>
<td>o prefetching</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

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

3. Conflict
   - Set size is smaller than mapped mem. locations
   - How to improve
     - large cache
     - more assoc.
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**
  - Local/global: $\frac{3,000}{100,000} = 3\%$

- **L2 miss rates**
  - Local: $\frac{1,500}{3,000} = 50\%$
  - Global: $\frac{1,500}{100,000} = 1.5\%$