CSE141 Winter 2014

Discussion Section 4: Caches
Cache Introduction

- Cache is used by the CPU to reduce average access time to memory.

- Two fundamental principles drive the functionality of caches.
  - **Spatial Locality:**
    - You're more likely to access memory nearby memory you previously accessed.
    - The instruction cache exploits spatial locality as there is high likelihood of instructions being close in proximity.
  - **Temporal Locality:**
    - There is higher likelihood to access the same address multiple times sequentially than something completely new.
Simple Cache Structure

- Cache that are direct mapped indicate a 1 to 1 correlation between the index of the address and the location of the cache entry in the cache.
- The direct mapped structure in image 1 shows how we would map 16 blocks to a 4 block direct mapped cache.
- Note the way the cache is indexed using the lower 2-bits of the memory block.
- If a cache has N blocks, then address M goes into block M mod N of the cache.
Direct Mapped Cache

When we need to load/store data we decouple the bits of an address as follows:

<table>
<thead>
<tr>
<th>Tag [n:x]</th>
<th>Index [x-1:y]</th>
<th>Byte Offset [y-1:0]</th>
</tr>
</thead>
</table>

- In order to take advantage of spatial locality the cache line is split into blocks, in Figure 2, say the cache has a capacity of 4KB ($2^{12}$) with 4-Byte ($2^2$) blocks.
  - As we have 4 blocks we need 2 bits of byte offset to represent all 4 blocks.

- In order to determine the index we take the total capacity of the cache and divide it by the number of blocks i.e. ($2^{12}$)/($2^2$) or 1024 ($2^{10}$) line
  - This means that when we decode the address for the cache lookup we will use an index of 10 bits, 1024 places.
Direct Mapped Cache Contd.

- The Tag is the remaining bits used in order properly identify the addresses we have in the particular address of the cache.

- Imagine we have 4 GB of main memory or \(2^{32}\).

- In our example, the bits remaining for the TAG will be the main memory size (32 bits) - offset (2bits) – index (10 bits) = 20 bit TAG.
Alternative Cache Structures

- If a cache is direct mapped and two identical indexes exist, the newer address takes the place of the old address in the cache line.

- Associativity allows us to resolve this issue by allowing multiple tags exist for n-similar indexes where n is the associativity of the cache.

- Common types include:
  - Direct-Mapped (1-way Set Associative)
  - N-way Set Associative (SA)
    - Where n is a power of 2
  - Fully Associative
N-Way Set Associative

- If a cache is 2-way set associative there are two ways or lines per index.
- This means that the cache can handle two memory references with differing tags even if they have the same index.
- 2-way associative caches halves the number of usable indexes versus direct mapped caches, (Figure 3: now 512 \([2 \ 9]\) instead of 1024 \([2 \ 10]\)) and reduces the number of conflicts misses.
Example 1

Given a 2-word block size, find the number of bits required for the tag, index, and block offset of the 2-way set associative cache from before, given 32-bit addresses.
Example 1: Solution

Index: \( s = \log_2 (\# \text{ sets}) \)
Block offset: \( b = \log_2 (\text{block size}) \)
Tag: \( t = \text{everything else} \)

\[ \text{Index} = \log_2 (8) = 3 \text{ bits} \]
\[ \text{Block offset} = \log_2 (4) = 2 \text{ bits} \]
\[ \text{Tag} = 32 - (3 + 2) = 27 \text{ bits} \]
Fully Associative Cache

- A cache is fully associative if for every memory address the address can go anywhere in the cache
- This way the index is completely eliminated and the cache line just contain the tag and offset
- This also means that in order to check if a particular cache line is in a cache the CPU has to look at every slot in the cache and compare the address tag bits to the tag of each slot using a comparator
- The hardware needed for doing this is large and ends up being not practical
- Only exception to this is the use of fully-associative caches in the translation lookaside buffer (TLB)
Cache Misses

- Compulsory
  - This occurs the first time a cache line is accessed and all the valid bits are set to 0 to indicate there is no valid data there
  - This commonly occurs when the system first boots
- Conflict
  - The data does not exist in the cache line because other valid data currently exists there
  - This commonly happens in direct-mapped caches
- Capacity
  - The data cannot fit in the cache because the cache is filled with valid data
  - This commonly occurs with Set Associative caches when the number of ways run out
Decreasing Miss Rate (H/W)

- Increase block size
  - Exploits spatial locality but results in a larger miss penalty
  - The AMAT goes up and overall miss rate can increase because temporal
- Increase associativity
  - This reduces conflict misses but requires more power
- Victim Cache
  - This is a small fully associative cache between the real cache and its refill path (contains only blocks replaced on recent misses).
  - On a miss the victim cache is checked, if the entry is present then swap victim cache entry and cache entry else do the instruction fetch and put the new victim entry (current instruction) in the victim cache
CSE141 Winter 2014

Review Session

Special Thanks: Christine Luu for providing slides on caches
The Final

- 90 Minutes
- More or less the length of the midterm
- Similar format
  - True/False for topics like datacenters
  - Pipelining
  - Study the homeworks
- Topics covered
  - Pipelining (HW2)
  - Branch Predictors (HW3)
  - Caches
  - Virtual Memory
  - Cache Coherency
  - Datacenters etc.
Virtual Memory

1. Determine how many bits are required for the virtual address, physical address, virtual page number, physical page number, and offset for the given configuration

32-bit operating system, 4-KB pages, 1 GB of RAM

Source: Steve Swanson

Shelby Thomas
Virtual Memory: Solution

Virtual Address = OS address length
Physical Address = log 2 (RAM size) bits
Offset = log 2 (page size) bits
Virtual Page Number bits = Virtual Address - Offset
Physical Page Number bits = Physical Address - Offset

Virtual Address = 32
Physical Address = 30
Offset = 20
Virtual Page Number bits = 18
Physical Page Number bits = 12
Caches

Suppose we now have 2 levels of cache: L1 and L2. For a particular program, there are 200 cache accesses in L1. How many misses were there in L1 if the local miss rate of L2 was 40%, and the global miss rate of L2 was 2%?
Caches: Solution

Answer: 10 misses in L1

Local miss rate and global miss rate must have the same numerator. The denominator of the local miss rate is the number of L1 misses (or the number of L2 accesses).
local: \( \frac{x}{y} = 40\% \)
global: \( \frac{x}{200} = 2\% \)

From global, solve for \( x \): \( x = 4 \).
From local, \( \frac{4}{y} = 40\% \)
Solve for \( y \): \( y = 10 \)
End Announcements

- Homework 4 is Review
  - We covered some of the questions today
- Slides will be online
- Turning in Projects