Jong Park

CS 472 – Computer Architecture

Professor Lizhong Chen

HW3 - Cache

May 22, 2019

**Problem 1 (20 pts)**

For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache.

|  |  |  |
| --- | --- | --- |
| Tag | Index | Offset |
| 31-12 | 11-7 | 6-0 |

**1.1) What is the cache block size (in words)?**

Offset bits (0-6) = 7

**1.2) How many entries does the cache have?**

Index bits (11-7) = 5

**1.3) What is the ratio of the total number of bits required for such a cache implementation (i.e., data, tag, valid bit) over the number of bits needed for data storage? [Hint: examples in the book around Figure 5.10.]**

**Total bits**  = entries \* (cache block bits + tag bits + valid bits)  
 = 32 entries \* ((32 \* 32) + (31-12+1) + 1) bits = **33440 bits**

**Total data storage bits**  = entries \* cache block bits  
 = 32 entries \* (32 \* 32) bits = **32768 bits**

**Ratio** = 33440 / 32768 = **1.021**

**1.4) How many blocks are replaced with the following accesses? [Hint: fill in the following table. “Block ID in cache” is “Block Address” mod “# of entries in cache”.]**

2 blocks (ID 2, 5) are replaced.

**Starting from an empty cache, the following byte-addressed cache references are recorded.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte Address | 1 | 348 | 756 | 9870 | 7980 | 364 | 4360 | 614 | 4740 | 3000 | 1440 | 2280 |
| Block Address | 0 | 2 | 5 | 77 | 62 | 2 | 34 | 4 | 37 | 23 | 11 | 17 |
| Block ID in Cache | 0 | 2 | 5 | 13 | 30 | 2 | 2 | 4 | 5 | 23 | 11 | 17 |
| Hit / Miss | M | M | M | M | M | H | M | M | M | M | M | M |
| Replace? | N | N | N | N | N | N | Y | N | Y | N | N | N |

**1.5) What is the hit ratio?**

1/12 = 0.083

**1.6) List the final state of the cache similar to Figure 5.9f. However, show only the final state (no intermediate steps) and only the valid entries (no need to show empty or not valid entries).**

|  |  |  |  |
| --- | --- | --- | --- |
| Index (Dec) | Index (Bin) | Tag | Data |
| 0 | 0 0000 | 0000 0000 0000 0000 0000 | Mem (0) |
| 2 | 0 0010 | 0000 0000 0000 0000 0001 | Mem (4360) |
| 4 | 0 0100 | 0000 0000 0000 0000 0000 | Mem (614) |
| 5 | 0 0101 | 0000 0000 0000 0000 0001 | Mem (756) |
| 11 | 0 1011 | 0000 0000 0000 0000 0000 | Mem (1440) |
| 13 | 0 1101 | 0000 0000 0000 0000 0010 | Mem (9870) |
| 17 | 1 0001 | 0000 0000 0000 0000 0000 | Mem (2280) |
| 23 | 1 0111 | 0000 0000 0000 0000 0000 | Mem (3000) |
| 30 | 1 1110 | 0000 0000 0000 0000 0001 | Mem (7980) |

**Problem 2 (20 pts)   
This exercise examines the impact of different cache designs, specifically comparing associative caches to the direct-mapped caches from Section 5.4. (In the table specify the block address range accessed, for instance 0-7, 0 is start address and 7 is end address)**

Word Length = 4 Bytes

Below is a list of 32-bit memory address references, given as ***byte addresses.***

3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253

2.1) Using the sequence of references from above, show the final cache contents for a three-way set associative cache with two-word blocks and a total cache size of 24 words. Use LRU replacement. For each reference identify the index bits, the block offset bits, and if it is a hit or a miss. The following table may help you to track the cache accesses.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte address | Block address | Hit | Index | Block Offset | Cache Content | | | | | | | | | | | |
| 0 | | | 1 | | | 2 | | | 3 | | |
| 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 |
| 3 | 0 | M | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 180 | 22 | M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 43 | 5 | M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** | **0** | **H** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 191 | 23 | M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 88 | 11 | M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **190** | **23** | **H** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 14 | 1 | M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **181** | **22** | **H** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **44** | **5** | **H** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **186** | **23** | **H** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 253 | 31 | M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

2.2) Using the references from above, show the final cache contents for a fully associative cache with one-word blocks and a total size of 8 words. Use LRU replacement. For each reference identify the index bits, the block offset bits, and if it is a hit or a miss.

Problem 3 (15 pts)

In this exercise, we will look at the different ways cache capacity affects overall performance. In general, cache access time is proportional to capacity. Assume that main memory accesses take 70 ns and that memory accesses are 36% of all instructions. The following table shows data for L1 caches attached to each of two processors, P1 and P2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | L1 Size | L1 Miss Rate | L1 Hit time |
| P1 | 2 KiB | 15.0% | 0.5ns |
| P2 | 4 KiB | 3.0 % | 1.5ns |

3.1) Assuming that the L1 hit time determines the cycle times for P1 and P2, what are their respective clock frequency?

3.2) What is the Average Memory Access Time for P1 and P2?   
[AMAT = hit time + miss rate \* miss penalty]

3.3) Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 and P2? Which processor is faster?

Extra credits for the following two questions (5 pts each)

For the next two problems, we consider the addition of an L2 cache to P1 to presumably make up for its limited L1 cache capacity. Use the L1 cache capacities and hit times from the previous table when solving these problems. The L2 miss rate indicated is its local miss rate.

|  |  |  |
| --- | --- | --- |
| L2 Size | L2 Miss Rate | L2 Hit Time |
| 1 MiB | 80% | 4ns |

3.4) What is the AMAT for P1 with the addition of an L2 cache? Is the AMAT better or worse with the L2 cache?

3.5) Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 with the addition of an L2 cache?

Problem 4 (10 pts)

The following list provides parameters of a virtual memory system.

|  |  |  |  |
| --- | --- | --- | --- |
| Virtual Address | Physical DRAM Installed | Page Size | PTE Size |
| 43 bits | 16 GiB | 8 KiB | 4 bytes |

For a single-level page table, how many page table entries (PTEs) are needed? How much physical memory is needed for storing the page table?

Problem 5 (15 pts)

There are several parameters that impact the overall size of the page table. Listed below are key page table parameters

|  |  |  |
| --- | --- | --- |
| Virtual Address | Page Size | Page Table Entry Size |
| 32 bits | 2 KiB | 4 bytes |

Given the parameters shown above, calculate the storage size needed for the total page tables of a system running 5 applications. If we have a 1GiB physical DRAM, what is the maximum number of applications that can be run simultaneously due to the storage issue of page tables?

Problem 6 (20 pts)

Virtual memory uses a page table to track the mapping of virtual addresses to physical addresses. This exercise shows how this table must be updated as addresses are accessed. The following data constitutes a stream of virtual addresses as seen on a system. Assume 4 KiB pages, a 4-entry fully associative TLB, and true LRU replacement (the larger LRU tag indicates less recently used). If pages must be brought in from disk, increment the next largest page number.

7843, 1998, 16744, 13344, 53233, 33214, 55167

TLB

|  |  |  |  |
| --- | --- | --- | --- |
| Valid | Tag | Physical Page Number | LRU Tag |
| 1 | 11 | 12 | 3 |
| 1 | 7 | 4 | 4 |
| 1 | 3 | 7 | 1 |
| 0 | 4 | 9 | 2 |

Page Table

|  |  |  |
| --- | --- | --- |
| Valid | Virtual Page | Physical Page or in Disk |
| 1 | 0 | 5 |
| 0 | 1 | Disk |
| 0 | 2 | Disk |
| 1 | 3 | 7 |
| 1 | 4 | 9 |
| 1 | 5 | 11 |
| 0 | 6 | Disk |
| 1 | 7 | 13 |
| 0 | 8 | Disk |
| 0 | 9 | Disk |
| 1 | 10 | 3 |
| 1 | 11 | 12 |

6.1) Given the address stream shown, and the initial TLB and page table states provided above, show the final state of the system. Also list for each reference if it is a hit in the TLB, a hit in the page table, or a page fault. You can assume that the initial TLB is filled from top to bottom (e.g., the top one is the oldest; note that you should always try to fill the empty (invalid) one first in fully-associate TLB).

[Hint: the virtual page number for 7843 is 1, so it misses in TLB as no tag matches. This is a decimal page fault and the page needs to be brought from disk. Based on the assumption that “If pages must be brought in from disk, increment the next largest page number”, the physical page number for this new page would be 14. Then we update the page table and the TLB. The second entry in the updated page table is (1, 1, 14), and the 4th entry in TLB is (1, 1, 14). Feel free to add more rows for the page table if needed.]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Virtual Address | 7843 | 1998 | 16744 | 13344 | 53233 | 33214 | 55167 |
| Virtual Page | 1 |  |  |  |  |  |  |
| TLB Hit | N |  |  |  |  |  |  |
| Page Fault | Y |  |  |  |  |  |  |

TLB

|  |  |  |  |
| --- | --- | --- | --- |
| Valid | Tag | Physical Page Number | LRU Tag |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Page Table

|  |  |  |
| --- | --- | --- |
| Valid | Virtual Page | Physical Page or in Disk |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

6.2) Show the final contents of the TLB if it is direct mapped (4 KiB page size).

TLB

|  |  |  |  |
| --- | --- | --- | --- |
| Valid | Tag | Physical Page Number | LRU Tag |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |