CS M151B Homework 8

Charles Zhang

March 2, 2022

Problem 5.16

As described in `Section 5.7`, 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 4KiB pages, a 4-entry fully associative TLB, and true LRU replacement. If pages must be brought in from disk, increment the next largest page number.

Address:

Decimal	4669	2227	13916	34587	48870	12608	49225
Hex	`0x123d`	`0x08b3`	`0x365c`	`0x871b`	`0xbee6`	`0x3140`	`0xc049`

TLB:

Valid	**Tag**	**Physical Page Number**	**Time Since Last Access**
1	11	12	4
1	7	4	1
1	3	6	3
`0`	4	9	7

Page table:

Index	**Valid**	**Physical Page or in Disk**
`0`	1	5
1	`0`	Disk
2	`0`	Disk
3	1	6
4	1	9
5	1	11
6	`0`	Disk
7	1	4
8	`0`	Disk
9	`0`	Disk
a	1	3
b	1	12

a) For each access shown above, list

- **whether the access is a hit or miss in the TLB,**
- **whether the access is a hit or miss in the page table,**
- **whether the access is a page fault,**
- **the updated state of the TLB.**

Entry	0x123d	0x08b3	0x365c	0x871b	0xbee6	0x3140	0xc049
TLB	Miss	Miss	Hit	Miss	Miss	Miss	Miss
Page Table	Miss	Hit	N/A	Miss	Hit	Hit	Miss
Fault?	Yes	No	No	Yes	No	No	Yes

Valid	Tag	Physical Page Number	Time Since Last Access
1	12	15	1
1	11	12	3
1	8	14	4
1	3	6	2

b) Repeat 5.16.1, but this time use 16KiB pages instead of 4KiB pages. What would be some of the advantages of having a larger page size? What are some of the disadvantages?

Entry	0x123d	0x08b3	0x365c	0x871b	0xbee6	0x3140	0xc049
TLB	Miss	Hit	Hit	Miss	Hit	Hit	Hit
Page Table	Hit	N/A	N/A	Miss	N/A	N/A	N/A
Fault?	No	No	No	Yes	No	No	No

Valid	Tag	Physical Page Number	Time Since Last Access
1	2	13	3
1	7	4	7
1	3	6	1
1	0	5	2

Some advantages of a larger page size include lower miss/fault rates and smaller page table sizes. Some disadvantages of a larger page size include increased fragmentation and larger miss penalties.

c) Repeat `Exercise 5.16.1`, but this time use 4KiB pages and a two-way set associative TLB.

Entry	`0x123d`	`0x08b3`	`0x365c`	`0x871b`	`0xbee6`	`0x3140`	`0xc049`
TLB	Miss	Miss	Miss	Miss	Miss	Hit	Miss
Page Table	Miss	Hit	Hit	Hit	Hit	N/A	Miss
Fault?	Yes	No	No	No	No	No	Yes

Index	**V1**	**T1**	**PPN1**	**Time 1**	**V2**	**T2**	**PPN1**	**Time 2**
`0`	1	6	14	1	1	4	9	4
1	1	1	6	2	1	5	12	3

d) Repeat `Exercise 5.16.1`, but this time use 4KiB pages and a direct mapped TLB.

Entry	`0x123d`	`0x08b3`	`0x365c`	`0x871b`	`0xbee6`	`0x3140`	`0xc049`
TLB	Miss	Miss	Miss	Miss	Miss	Miss	Miss
Page Table	Miss	Hit	Hit	Miss	Hit	Hit	Miss
Fault?	Yes	No	No	Yes	No	No	Yes

Index	**Valid**	**Tag**	**Physical Page Number**
`0`	1	3	15
1	1	0	13
2	0	-	-
3	1	0	6

e) Discuss why a CPU must have a TLB for high performance. How would virtual memory accesses be handled if there were no TLB?

A TLB is required, because, if it didn't exist, any loads from memory would first require the PTE to be loaded from memory. This would result in a steep performance drop, as each load would be accompanied by an extra load. In the case of hierarchal page tables, this effect would be worsened even further, as each level of page table would add an additional load to the process.

Problem 5.20

In this exercise, we will examine how replacement policies impact miss rate. Assume a 2-way set associative cache with 4 one word blocks. Consider the following word address sequence: 0, 1, 2, 3, 4, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0.

a) Assuming an LRU replacement policy, which accesses are hits?

In this case, no accesses are hits, as the cache constantly cycles through each word in order.

b) Assuming an MRU (most recently used) replacement policy, which accesses are hits?

This would allow the second 3, the second 0, the second 1, and the last 0 to all be hits. Once 0 and 1 enter the cache, they are never evicted since there will always be a more recently used value in the same index. The second 3 appears before any other odd value can evict the 3 cached from the first 3.

c) Simulate a random replacement policy by flipping a coin. For example, "heads" means to evict the first block in a set and "tails" means to evict the second block in a set. How many hits does this address sequence exhibit?

The second 3, second 4, second 1, and last 7 were all hits under this random policy.

d) Describe an optimal replacement policy for this sequence. Which accesses are hits using this policy?

An optimal replacement policy for this sequence would cache the words at 0, 1, 2, and 3, and then replace 1 with 4. This allows the second 2, 3, and 4 to be hits, as well as the second 0 and third 2, 3, 4, and 0.

e) Describe why it is difficult to implement a cache replacement policy that is optimal for all address sequences.

For a cache replacement policy to be optimal, even in a specific case, it would require knowledge of all cache accesses within the access stream. This alone would require a significant amount of computational complexity that would inhibit the performance of the caches. A robust policy would be even harder to implement, as various access patterns all would have vastly different optimal policies, resulting in even more computational complexity needed to address this issue.

f) Assume you could make a decision upon each memory reference whether or not you want the requested address to be cached. What impact could this have on miss rate?

Assuming this decision could be made with the entire reference sequence in mind, this could improve the miss rate by looking ahead and checking to see if caching the current memory reference would be worth evicting another memory reference by seeing which may appear later.