Computer Organization and Design

Homework 8

5.12 In this exercise, we will examine space/time optimizations for page tables. The following list provides parameters of a virtual memory system.

Virtual Address (bits)	Physical DRAM Installed	Page Size	PTE Size (byte)
43	16 GiB	4 KiB	4

5.12.1 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?
5.12.2 Using a multilevel page table can reduce the physical memory consumption of page tables, by only keeping active PTEs in physical memory. How many levels of page tables will be needed in this case? And how many memory references are needed for address translation if missing in TLB?
5.12.3 An inverted page table can be used to further optimize space and time. How many PTEs are needed to store the page table? Assuming a hash table implementation, what are the common case and worst case numbers of memory references needed for servicing a TLB miss?
The following table shows the contents of a 4-entry TLB.

Entry-ID	Valid	VA Page	Modified	Protection	PA Page
1	1	140	1	RW	30
2	0	40	0	RX	34
3	1	200	1	RO	32
4	1	280	0	RW	31

5.12.4 Under what scenarios would entry 2's valid bit be set to zero?
5.12.5 What happens when an instruction writes to VA page 30? When would a software managed TLB be faster than a hardware managed TLB?
5.12.6 What happens when an instruction writes to VA page 200?

Solution:
5.12.1 Page Size $= 4\text{KiB}$, so there are $log_2 4096 = 12\ bits$ page offset.
For a single-level page table, there will be 43-12=31 bits virtual page number.
So page table entries $= 2^{31}$
PTE Size =4 Bytes. We need $2^{31} \times 4\ bytes = 2^{33}\ Bytes = 8G\ Bytes$ physical memory.

5.12.2 With only two levels, the designer can select the size of each page table segment. In a multi-level scheme, reading a PTE requires an access to each level of the table.

5.12.3 In an inverted page table, the number of PTEs can be reduced to the size of the hash table plus the cost of collisions. In this case, serving a TLB miss requires an extra

reference to compare the tag or tags stored in the hash table.

5.12.4 It would be invalid if it was paged out to disk.

5.12.5 A write to page 30 would generate a TLB miss. Software-managed TLBs are faster in cases where the software can pre-fetch TLB entries.

5.12.6 When an instruction writes to VA page 200, and interrupt would be generated because the page is marked as read only.

5.13 In this exercise, we will examine how replacement policies impact miss rate. Assume a 2-way set associative cache with 4 blocks. To solve the problems in this exercise, you may find it helpful to draw a table like the one below, as demonstrated for the address sequence "0, 1, 2, 3, 4."

Address of Memory Block Accessed	Hit or Miss	Evicted Block	Contents of Cache Blocks After Reference			
			Set 0	Set 0	Set 1	Set 1
0	Miss		Mem[0]			
1	Miss		Mem[0]		Mem[1]	
2	Miss		Mem[0]	Mem[2]	Mem[1]	
3	Miss		Mem[0]	Mem[2]	Mem[1]	Mem[3]
4	Miss	0	Mem[4]	Mem[2]	Mem[1]	Mem[3]
...						

Consider the following address sequence: 0, 2, 4, 8, 10, 12, 14, 16, 0

5.13.1 Assuming an LRU replacement policy, how many hits does this address sequence exhibit?
5.13.2 Assuming an MRU (most recently used) replacement policy, how many hits does this address sequence exhibit?
5.13.3 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?
5.13.4 Which address should be evicted at each replacement to maximize the number of hits? How many hits does this address sequence exhibit if you follow this "optimal" policy?
5.13.5 Describe why it is difficult to implement a cache replacement policy that is optimal for all address sequences.
5.13.6 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?

Solution:
5.13.1

Address of Memory Block Accessed	Hit or Miss	Evicted Block	Content of Cache Block After Reference			
			Set 0	Set 0	Set 1	Set 1
0	Miss		Mem[0]			
2	Miss		Mem[0]	Mem[2]		
4	Miss	0	Mem[4]	Mem[2]		
8	Miss	2	Mem[4]	Mem[8]		
10	Miss	4	Mem[10]	Mem[8]		
12	Miss	8	Mem[10]	Mem[12]		
14	Miss	10	Mem[14]	Mem[12]		
16	Miss	12	Mem[14]	Mem[16]		
0	Miss	14	Mem[0]	Mem[16]		

0 hit

5.13.2

Address of Memory Block Accessed	Hit or Miss	Evicted Block	Content of Cache Block After Reference			
			Set 0	Set 0	Set 1	Set 1
0	Miss		Mem[0]			
2	Miss		Mem[0]	Mem[2]		
4	Miss	2	Mem[0]	Mem[4]		
8	Miss	4	Mem[0]	Mem[8]		
10	Miss	8	Mem[0]	Mem[10]		
12	Miss	10	Mem[0]	Mem[12]		
14	Miss	12	Mem[0]	Mem[14]		
16	Miss	14	Mem[0]	Mem[16]		
0	Hit		Mem[0]	Mem[16]		

1 hit

5.13.3 0 or 1(1 hit or fewer)

5.13.4 Any address sequence is fine so long as the number of hits are correct. 1 hit.

5.13.5 The best block to evict is the one that will cause the fewest misses in the future. Unfortunately, a cache controller cannot know the future! Our best alternative is to make a good prediction.

5.13.6 If you knew that an address had limited temporal locality and would conflict with another block in the cache, it could improve miss rate. On the other hand, you could worsen the miss rate by choosing poorly which addresses to cache.