Name D PORTS

Computer System Architecture
6.823 Quiz #2
October 20th, 2006
Professor Krste Asanovic
Dr. Joel Emer

Name: DAN PORTS

This is a closed book, closed notes exam.
80 Minutes
10 Pages

Notes:

- Not all questions are of equal difficulty, so look over the entire exam and budget your time carefully
- Please carefully state any assumptions you make
- Please write your name on every page in the quiz
- You must not discuss a quiz's contents with other students who have not yet taken the quiz

Writing name on each sheet	2	2 Points
Question 1	20	20 Points
Question 2	29	29 Points
Question 3	26	29 Points
TOTAL	77	**80 Points**

Name D Ponts

Question 1: Different Cache Configurations and their Performance (20 points)

In this question we are trying to determine the best cache configuration for a new processor. We know how to build two kinds of caches: direct-mapped and fully set-associative. We want to know how these two different configurations affect the cache miss rate.

Part A (4 points) 4

Assume a 32 bytes cache with 8 bytes cache lines. The processor generates byte addresses of 8 bits. Fill in the address breakdown table for this cache with the following **fields** and their **corresponding address bits**. (The byte offset selects the portion of the cache line being accessed)

4 lines

Index
Tag
Byte offset

Address Bits	7 5	4 3	2 0
Field	tag	index	offset

Part B (4 points) 4

Consider a fully set-associative cache. Assume that the total cache size is still 32 bytes and that all other parameters (such as the input address, cache line, etc.) are the same as in part A. Fill in the address breakdown table for this cache with the following **fields** and their **corresponding address bits**.

Index
Tag
Byte offset

Address Bits	7	3	2	0
Field	tag		offset	

Name D Ports

Part C (6 points) \6

In this part of the question, we will perform a miss rate analysis on the direct-mapped cache presented in Part A.

We test the cache by accessing the given sequence of binary byte addresses, starting with an empty cache. Complete the following table for the direct-mapped cache and show the progression of cache contents as accesses occur (in the tables, `inv` = invalid, and the column shows the first byte address that the cache line contains). *The first 3 rows of the table have been filled out for you. You only need to fill in the elements in the table when a value changes except for the 'Hit?' column which you should fill out completely.*

Direct Mapped Access Address	**Hit?**	**State after Access**			
		Line in Cache			
		L0	**L1**	**L2**	**L3**
1001 1001	No	inv	inv	inv	1001 1000
0100 0011	No	0100 0000			
1001 1111	Yes				
1111 1111	No				1111 1000 ✓
0011 0111	No			0011 0000 ✓	
1011 1000	No				1011 1000 ✓
1100 1000	No		1100 1000 ✓		
1111 1100	No				1111 1000 ✓
1011 1110	No				1011 1000 ✓
0011 0000	Yes				

Name D PORTS

Part D (6 points) 6

Now we will perform a miss rate analysis on the fully set-associative cache presented in Part B. This cache employs the least recently used (LRU) replacement policy.

For the same access sequence as in Part C, complete the following table for the fully set-associative cache showing the progression of cache contents as accesses occur. *The first 2 rows of the table have been filled out for you. You only need to fill in the elements in the table when a value changes except for the 'Hit?' column. If multiple entries in a set of the fully set-associative cache are* `inv`*, way(i) is filled first, where i has the smallest possible value.*

2-way Set Associative		**State after Access**			
		Line in Cache			
Access Address	**Hit?**	**way0**	**way1**	**way2**	**way3**
1001 1001	No	1001 1000	inv	inv	inv
0100 0011	No		0100 0000		
1001 1111	Yes				
1111 1111	No			1111 1000	
0011 0111	No				0011 0000
1011 1000	No		1011 1000		
1100 1000	No	1100 1000			
1111 1100	Yes				
1011 1110	Yes				
0011 0000	Yes				

Name D PORTS

Question 2: Virtually-Indexed Physically-Tagged Cache (29 Points)

In this question, we will study the operations of a virtually-indexed physically-tagged 2-way set-associative cache.

Part A (4 Points)

Assume that a byte-addressed machine uses 20-bit virtual addresses and 16-bit physical addresses. The page size is 256 bytes.

Fill in the following tables to show the address breakdown of the virtual address and the physical address into the following **fields** and their **corresponding address bits**.

VPN
PPN
Page offset

Address Bits	19 8	7 0
Virtual Address Field	VPN	offset

Address Bits	15 8	7 0
Physical Address Field	PPN	offset

✓

Part B (3 Points)

In this machine, the size of the virtually-indexed physically-tagged 2-way set-associative cache is 64 bytes (32 bytes per way) and it employs the Least Recently Used (LRU) replacement policy. Each cache line is 16 bytes long. The cache only stores the smallest physical tags required which are taken from the high order bits of an address.

Based on the given information, fill in the following blanks.

The machine will use address bit 4 to 4 of the Virtual (**Virtual** or **Physical**) address as the index to access the cache.

The machine will use address bit 15 to 5 of the Physical (**Virtual** or **Physical**) address as the tag to perform the cache tag comparison. ✓

7

Name D POINTS

Part C (12 Points)

In this part of the question, we ask you to show step-by-step operation of the cache given in Part B. Assume that the cache is initially empty and the TLB states are given by the table below. (Only VPNs and PPNs are shown for the TLB)

VPN	PPN
0x001	0x0C
0x007	0x4C
0x05A	0x08
0x06B	0x8C
0x1C3	0x4C
0x3E6	0xC8

TLB states

The following table shows the cache state after the first 3 accesses from a virtual address access sequence. Complete the following table to show the progression of the cache contents as accesses occur. (Only valid (V) bits, LRU (L) bits and physical addresses (PA) are shown for the cache, the L bit is 1 if way0 was accessed last and 0 otherwise. The PA column shows the first byte address that the cache line contains. *You should write the PA in hexadecimal form. And you only need to fill in the elements in the table when a value changes except for the 'Hit?' column.*

Access Address	Hit	State after Access									
		Line in Cache									
		Set0					Set1				
			way0		Way1			way0		way1	
		L	V	PA	V	PA	L	V	PA	V	PA
0x00704	No	1	1	4C00	0		0	0		0	
0x05A14	No						1	1	0810		
0x06B20	No	0			1	8C20					
0x1C304 4C04	Yes	1									
0x3E648 C848	No	0				C840					
0x00150 0C50	No						0			1	0C50
0x06B28 8C28	No	1		8C20							
0x0070C 4C0C	No	0				4C00					

12

Name D Ponts

Part D (5 Points)

Is it possible for this cache to have an aliasing problem (i.e. multiple copies of the data from the same physical address reside in the cache)? Please briefly explain your answer.

No. We could only have an aliasing problem if one physical address could be loaded into both sets based on the virtual address. But each physical address can only reside in one set because the set is selected by bit 4, which is part of the page offset and will never change based on the page mappings. ✓

Part E (5 Points)

If we increase the cache size to 8 KB, what is maximum number of cache sets can a given physical address reside in? Please briefly explain your answer.

8 KB ⇒ 4 KB / way ⇒ $\frac{4\text{ KB}}{16\text{ B}} = 256$ sets

Bits 11 - 4 of the virtual address determine the set.

So bits 11 - 8 are the intersection of the VPN and the index. Varying these bits allows the same PA to be stored in multiple locations. So there are $2^4 = \boxed{16}$ possibilities. ✓

10

Name D POINTS

Question 3 (29 Points)

In this question, we will study how memory hierarchies affect system performance. We start with a 5-stage Harvard-style MIPS pipeline. This machine has magic memories for both the instruction memory and the data memory (i.e. all memory accesses are serviced in 1 cycle). Moreover, there is no virtual memory support in this machine. When we run a test bench on this machine, the cycles per instruction (CPI) is 3. In the following parts of the question, we will replace the data memory step by step with an increasingly realistic memory system and study the effects of each change. On the other hand, we will assume that the instruction memory will remain magical throughout the question.

Part A (5 Points) 5

As a first step, we replace the magic data memory with an L1 cache and a real memory. The access latencies of the L1 and the real memory are 1 cycle and 100 cycles respectively. If the L1 misses 25 times per 1000 instructions, what will be the CPI for this new machine? Assume that the system only accesses the real memory when L1 misses and that the entire pipeline stalls while waiting for a cache miss to be serviced.

The entire pipeline is stalled for 100 cycles during a L1 miss, which occurs 25/1000 instructions, so the CPI is

$CPI = 3 + 100 \cdot 25/1000 = \boxed{5.5}$ ✓

Part B (5 Points) 5

Now, we would like to add an L2 cache in between the L1 cache and the memory from Part A. The access latency of the L2 is 10 cycles. If the L2 misses 5 times per 1000 instructions, what will be the CPI for this new machine? Assume that the system only accesses L2 when L1 misses and accesses the memory when L2 misses, and that the entire pipeline stalls while waiting for a cache miss to be serviced.

The pipeline is stalled for 10 cycles each L1 miss & 100 cycles every L2 miss.

$CPI = 3 + 10 \cdot \frac{25}{1000} + 100 \cdot \frac{5}{1000}$ ✓

$= 3 + 0.25 + 0.5 = \boxed{3.75}$

Name 0 Points

Part C (5 Points) 5

What will be the CPI for Part B, if the system accesses the memory and the L2 at the same time whenever there is an L1 miss?

Same as before (pt. B) except that the L2 miss cost = mem. latency − L2 latency
= 100 − 90 = 90 cycles

$$CPI = 3 + 10 \cdot \frac{25}{1000} + 90 \cdot \frac{5}{1000}$$

$$= 3 + 0.25 + 0.45 = \boxed{3.7}$$ ✓

Part D (4 Points) 2

Please give an advantage and a disadvantage of the design in Part C in comparison with the design in Part B?

Advantage: lower CPI ✓

Disadvantage: Memory must be able to support having its address lines changed every 10 cycles (1/10 of its access time), which some memory may not support X

Name 0 Points

Part E (5 Points) 4

Instead of adding an L2 to the machine as in Part B and Part C, Ben Bitdiddle is more interested in adding virtual memory support to the system. In his system, the L1 cache becomes a virtually-indexed physically-tagged cache. He also adds a TLB to the system. TLB lookups are performed in parallel with the cache accesses. Therefore, the pipeline remains 5-stage. Whenever there is a TLB miss, his system will perform the translation by walking through a 2-level hierarchical page table and then save the result in the TLB.

Assume that the TLB misses 10 times per 1000 instructions, 40% of the accesses to a Level 1 page table can be serviced by the L1 cache, 20% of the accesses to a Level 2 page table can be serviced by the L1 cache, and all page tables and data are resident in the memory. What will be the CPI of this machine? Assume that entire pipeline stalls while waiting for a TLB miss to be serviced.

avg TLB miss cost = 1 + (0.6 · 100) + (0.8 · 100)
(L1 miss) (Level 2 fetch)
= 140 cycles

X

CPI = 3 + L1 miss rate · mem latency + TLB miss rate · (TLB miss cost)

$= 3 + \frac{25}{1000} \cdot 100 + \frac{10}{1000} \cdot 140$

$= 3 + 2.5 + 1.4 = \boxed{6.9}$

Part F (5 Points) 5

Can you briefly explain why you would expect a Level 1 page table access to have a higher hit rate on the L1 cache than a Level 2 page table access?

Memory accesses exhibit spatial locality, so a common pattern will be to access multiple Level 2 pages pointed to by the same Level 1 page. Since on each TLB miss, both the Level 1 and Level 2 page tables are loaded into the cache, this pattern means the shared Level 1 page table will be in the cache more frequently