ECE/CS 472/572 – Computer Architecture
Instructor: Prof. Lizhong Chen
Homework #3 –Solution

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)?

We have 0 - 6 in total 7 bits as offset bits.
2^7 = 128 bytes = 32 words.

1.2) How many entries does the cache have?

We have 11 - 7 in total 5 bits as index bits.
2^5 = 32 entries

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 = 2^{index} *(cache block size + tag bits + valid bit) = 32 * ((32 * 32) + (31 - 12 + 1) + 1) = 32 * 1045 bits
Total data storage bits = 2^{index} *cache block size =32* (32 * 32 bits) =32 * 1024 bits

Ratio $= \frac{\mathbf{32\ entries * 1045\ bits/entry}}{\mathbf{32\ entries * 1024\ bits/entry}} = \mathbf{1.02}$

1.4) How many blocks are replaced? [Hint: fill in the following table.]

Starting from power on, the following *byte-addressed* cache references are recorded.

1	348	756	9870	7980	364	4360	614	4740	3000	1440	2280

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? (Y/N)	N	N	N	N	N	N	Y	N	Y	N	N	N

In total 2 blocks are replaced.

1.5) What is the hit ratio?

1/12

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	Tag	Data
0	**0000 0000 0000 0000 0000**	**Mem(0 - 127)**
2	**0000 0000 0000 0000 0001**	**Mem(4352 - 4479)**
4	**0000 0000 0000 0000 0000**	**Mem(512 - 639)**
5	**0000 0000 0000 0000 0001**	**Mem(4736 - 4863)**
11	**0000 0000 0000 0000 0000**	**Mem(1408 - 1535)**
13	**0000 0000 0000 0000 0010**	**Mem(9856 - 9983)**
17	**0000 0000 0000 0000 0000**	**Mem(2176 - 2303)**
23	**0000 0000 0000 0000 0000**	**Mem(2944 - 3071)**
30	**0000 0000 0000 0000 0001**	**Mem(7936 - 8063)**

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.

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 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.

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	00	3	0-7											
180	22	M	10	4	0-7						176-183					
43	5	M	01	3	0-7			40-47			176-183					
2	0	H	00	2	0-7			40-47			176-183					
191	23	M	11	7	0-7			40-47			176-183			184-191		
88	11	M	11	0	0-7			40-47			176-183			184-191	88-95	
190	23	H	11	6	0-7			40-47			176-183			184-191	88-95	
14	1	M	01	6	0-7			40-47	8-15		176-183			184-191	88-95	

181	22	H	10	5	0-7			40-47	8-15		176-183			184-191	88-95	
44	5	H	01	4	0-7			40-47	8-15		176-183			184-191	88-95	
186	23	H	11	2	0-7			40-47	8-15		176-183			184-191	88-95	
253	31	M	11	5	0-7			40-47	8-15		176-183			184-191	88-95	248-255

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.

byte address	block address	Hit	Index	Block Offset	**Cache content**							
					0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3	0	M	N/A	3	0-3							
180	45	M	N/A	0	0-3	180-183						
43	10	M	N/A	3	0-3	180-183	40-43					
2	0	H	N/A	2	0-3	180-183	40-43					
191	47	M	N/A	3	0-3	180-183	40-43	188-191				
88	22	M	N/A	0	0-3	180-183	40-43	188-191	88-91			
190	47	H	N/A	2	0-3	180-183	40-43	188-191	88-91			

14	3	M	N/A	2	0-3	180-183	40-43	188-191	88-91	12-15		
181	45	H	N/A	1	0-3	180-183	40-43	188-191	88-91	12-15		
44	11	M	N/A	0	0-3	180-183	40-43	188-191	88-91	12-15	44-47	
186	46	M	N/A	2	0-3	180-183	40-43	188-191	88-91	12-15	44-47	184-187
253	63	M	N/A	1	0-3	180-183	252-255	188-191	88-91	12-15	44-47	184-187

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?

Clock Frequency of P1 = 1/0.5ns = 2 GHz

Clock Frequency of P2 = 1/1.5ns = 0.67 GHz

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

AMAT of P1 = 0.5 + 15% * 70 = 11 ns

AMAT of P2 = 1.5 + 3% * 70 = 3.6 ns

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?

CPI = base CPI + ($\frac{number\ of\ memory\ accesses}{total\ instructions}$) * (Miss rate L1) * (Miss penalty L1 in cycles)

CPI of P1 = 1 + 0.36 * 0.15 * (70 / 0.5) = 8.56 cycles
CPI of P2 = 1 + 0.36 * 0.03 * (70 / 1.5) = 1.5 cycles

Actual Time of P1 = 8.56 * 0.5 = 4.28 ns
Actual Time of P2 = 1.5 * 1.5 = 2.25 ns

Therefore, P2 is faster.

Extra credits for the following two questions

(5 pts each)
For the next two problems, we will 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%	4 ns

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?

AMAT of P1 = 0.5 + 15% * (4 + 70 * 80%) = 9.5 ns
Therefore, AMAT is better with L2 in P1

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?

CPI = base CPI + ($\frac{number\ of\ memory\ accesses}{total\ instructions}$) * (Miss rate L1) * (Miss penalty)
CPI of P1 = 1 + 0.36 * 0.15 * ((4 + 0.8 * 70) / 0.5) = 7.48 cycles

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?

Page Size = 8 KiB = 2^{13}
PTE entries = $2^{43 - 13}$ = 2^{30} entries
Page Table size = 2^{30} * 4 bytes = 2^{32} bytes = 4 GiB

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 total page table size for 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?

Page Size = 2 KiB = 2^{11}
Page Table size = $2^{32 - 11}$ * 4 bytes = 2^{23} = 8 MiB
Since we have 5 applications:
Total PT size = 5 * 8 MiB = 40 MiB

1 GiB / (8 MiB/application) = 128 application
Therefore, with 1 GiB, the maximum number of applications that can be run simultaneously is 128.

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 an entry holds, the least recent used by system)**. If pages must be brought in from disk, increment the next largest page number.

<u>7843</u>, <u>1998</u>, <u>16744</u>, <u>13344</u>, <u>53233</u>, <u>33214</u>, <u>55167</u>

TLB

Valid	Tag	Physical Page Number	LRU Tag
1	11	12	3
1	7	4	4

1	3	6	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 $1_{decimal}$, so it misses in TLB as no tag matches.This is a 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, 14), and the 4th entry in TLB is (1, 1, 14). Feel free to add more rows for PT if you need.]

Virtual Address	7843	1998	16744	13344	53233	33214	55167
Virtual Page	1	0	4	3	12	8	13
TLB Hit	N	N	N	Y	N	N	N

Page Fault	Y	N	N	N	Y	Y	Y

7843:

TLB

Valid	Tag	Physical Page Number	LRU Tag
1	11	12	3
1	7	4	4
1	3	7	2
1	1	14	1

1998:

TLB

Valid	Tag	Physical Page Number	LRU Tag
1	11	12	4
1	0	5	1
1	3	7	3
1	1	14	2

16744:

TLB

Valid	Tag	Physical Page Number	LRU Tag
1	4	9	1
1	0	5	2
1	3	7	4
1	1	14	3

13344:

TLB

Valid	Tag	Physical Page Number	LRU Tag
1	4	9	2
1	0	5	3

1	3	7	1
1	1	14	4

53233:

TLB

Valid	Tag	Physical Page Number	LRU Tag
1	4	9	3
1	0	5	4
1	3	7	2
1	12	15	1

33214:

TLB

Valid	Tag	Physical Page Number	LRU Tag
1	4	9	4
1	8	16	1
1	3	7	3
1	12	15	2

55167:

TLB

Valid	Tag	Physical Page Number	LRU Tag
1	13	17	1
1	8	16	2
1	3	7	4
1	12	15	3

Page Table

Valid	Virtual Page	Physical Page or in Disk
1	0	5
1	1	14

0	2	Disk
1	3	7
1	4	9
1	5	11
0	6	Disk
1	7	13
1	8	16
0	9	Disk
1	10	3
1	11	12
1	12	15
1	13	17

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

Virtual Address	7843	1998	16744	13344	53233	33214	55167
Virtual Page	1	0	4	3	12	8	13
Tag	0	0	1	0	3	2	3
Index	1	0	0	3	0	0	1
TLB Hit	N	N	N	N	N	N	N
Page Fault	Y	N	N	N	Y	Y	Y

7843:

TLB

Set	Valid	Tag	Physical Page Number
0	1	11	12
1	1	0	14
2	1	3	7

3	0	4	9

1998:

TLB

Set	Valid	Tag	Physical Page Number
0	1	0	5
1	1	0	14
2	1	3	7
3	0	4	9

16744:

TLB

Set	Valid	Tag	Physical Page Number
0	1	1	9
1	1	0	14
2	1	3	7
3	0	4	9

13344:

TLB

Set	Valid	Tag	Physical Page Number
0	1	1	9
1	1	0	14
2	1	3	7
3	1	0	7

53233:

TLB

Set	Valid	Tag	Physical Page Number
0	1	3	15
1	1	0	14

2	1	3	7
3	1	3	7

33214:

TLB

Set	Valid	Tag	Physical Page Number
0	1	2	16
1	1	0	14
2	1	3	7
3	1	3	7

55167:

TLB

Set	Valid	Tag	Physical Page Number
0	1	2	16
1	1	3	17
2	1	3	7
3	1	3	7

Page Table

Valid	Virtual Page	Physical Page or in Disk
1	0	5
1	1	14
0	2	Disk
1	3	7
1	4	9
1	5	11
0	6	Disk
1	7	13

1	8	16
0	9	Disk
1	10	3
1	11	12
1	12	15
1	13	17