Answers for Homework #3

(Please use Q&A board to ask questions)

- Solve the listed problems in our textbook.
  - 5.2.1~5.2.4, 5.7.1~5.7.4, 5.11.1~5.11.3

- You **don't have to submit** your report. However, this homework could be related to final exam, so we highly recommend you to solve the problems

5.2 Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 32-bit memory address references, given as word addresses.

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

5.2.1 For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

: Since the question gives address references as word addresses, it does not have to consider 2-bit byte offset. The question assumes the direct-mapped cache with 16 one-word blocks. So, there are not any bits for the block offset and 4 bits (16=2^4) are needed for an index. Remains (upper 28 bits) are used as a tag.

In the table, binary address omits upper 24 bits because they are always '0'. There are not any detailed classification for the miss.

Word Address	Binary Address	Tag	Index	Hit/Miss
3	0000 0011	0	3	M
180	1011 0100	11	4	M
43	0010 1011	2	11	M
2	0000 0010	0	2	M
191	1011 1111	11	15	M
88	0101 1000	5	8	M
190	1011 1110	11	14	M
14	0000 1110	0	14	M
181	1011 0101	11	5	M
44	0010 1100	2	12	M
186	1011 1010	11	10	M
253	1111 1101	15	13	M

5.2.2 For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with two-word blocks and a total size of 8 blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

: Differences between 5.2.1 and 5.2.2 are the number of words in a block and the total size of blocks in the cache. In this time, there are two-word blocks, so it needs 1 bit ($2 = 2^1$) for the block offset. Despite of the block offset, the bits for the index is decreased as 3 bits ($8 = 2^3$) for 8 blocks. Since the total number of used bits except the tag is same with 5.2.1, the tag is not changed.

Word Address	Binary Address	Tag	Index	Hit/Miss
3	0000 0011	0	1	M
180	1011 0100	11	2	M
43	0010 1011	2	5	M
2	0000 0010	0	1	H
191	1011 1111	11	7	M
88	0101 1000	5	4	M
190	1011 1110	11	7	H
14	0000 1110	0	7	M
181	1011 0101	11	2	H
44	0010 1100	2	6	M
186	1011 1010	11	5	M
253	1111 1101	15	6	M

5.2.3 You are asked to optimize a cache design for the given references. There are three direct-mapped cache designs possible, all with a total of 8 words of data: C1 has 1-word blocks, C2 has 2-word blocks, and C3 has 4-word blocks. In terms of miss rate, which cache design is the best? If the miss stall time is 25 cycles, and C1 has an access time of 2 cycles, C2 takes 3 cycles, and C3 takes 5 cycles, which is the best cache design?

In the question, it assumes the total of 8 words of data in the direct-mapped cache. It means that the below equation should always be established.

$$(The\ number\ of\ words\ in\ a\ block) * (The\ total\ number\ of\ blocks) = 8$$

With this equation, we can calculate the total number of blocks for each caches: C1 = 8 blocks, C2 = 4 blocks, and C3 = 2 blocks. There are 3 bits ($8 = 2^3$) for the equation, so the number of bits for the tag is always 29 bits for every caches.

Based on this information, we can calculate the number of bit for the index and the block offset for each caches.

Word Address	Binary Address	Tag	Cache 1		Cache 2		Cache 3	
			Index	hit/miss	Index	hit/miss	Index	hit/miss
3	0000 0011	0	3	M	1	M	0	M
180	1011 0100	22	4	M	2	M	1	M
43	0010 1011	5	3	M	1	M	0	M
2	0000 0010	0	2	M	1	M	0	M
191	1011 1111	23	7	M	3	M	1	M
88	0101 1000	11	0	M	0	M	0	M
190	1011 1110	23	6	M	3	H	1	H
14	0000 1110	1	6	M	3	M	1	M
181	1011 0101	22	5	M	2	H	1	M
44	0010 1100	5	4	M	2	M	1	M
186	1011 1010	23	2	M	1	M	0	M
253	1111 1101	31	5	M	2	M	1	M

For the miss rate,

$$(Miss\ rate) = \frac{(The\ number\ of\ misses)}{(The\ total\ number\ of\ addresses)}$$

For total cycles,

$$\begin{aligned}(Total\ cycles) &= (The\ number\ of\ misses) \times (The\ miss\ stall\ time) \\ &\quad + (The\ total\ number\ of\ addresses) \times (The\ access\ time)\end{aligned}$$

Cache	Miss rate	Total cycles
#1	12/12 = 1.000	12*25+12*2 = 324
#2	10/12 = 0.833	10*25+12*3 = 286
#3	11/12 = 0.917	11*25+12*5 = 335

Therefore, both terms of the miss rate and the total cycles, C2 shows the best performance.

5.2.4 There are many different design parameters that are important to a cache's overall performance. Below are listed parameters for different direct-mapped cache designs.

- Cache Data Size: 32 KiB
- Cache Block Size: 2 words
- Cache Access Time: 1 cycle

Calculate the total number of bits required for the cache listed above, assuming a 32-bit address. Given that total size, find the total size of the closest direct-mapped cache with 16-word blocks of equal size or greater. Explain why the second cache, despite its larger data size, might provide slower performance than the first cache.

Because the first cache listed above uses the 32-bit address, the number of bytes per word is 4-byte.

$$(The\ total\ number\ of\ words) = \frac{(Cache\ data\ size)}{(The\ number\ of\ bytes\ per\ word)} = \frac{32\text{KiB}}{4\text{B}} = 8{,}192 \text{ words}$$

Moreover, it uses 2-word blocks, so the total number of blocks are

$$(The\ total\ number\ of\ blocks) = \frac{(The\ total\ number\ of\ words)}{(The\ number\ of\ words\ in\ a\ block)} = \frac{8{,}192}{2} = 4{,}096 \text{ blocks}$$

With the total number of blocks, we can calculate the number of bits for the index.

$$(The\ number\ of\ bits\ for\ the\ index) = \log_2(The\ total\ number\ of\ blocks) = \log_2 4{,}096 = 12 \text{ bits}$$

Since the cache uses 2-word blocks, it needs 1 bit for the block offset. The cache has 2 bits for the byte offset when it uses 32-bit word. For the length of the tag,

$$\begin{aligned}&(The\ total\ number\ of\ bits\ for\ the\ tag)\\&= (word\ address) - (index) - (block\ offset) - (byte\ offset)\\&= 32 - 12 - 1 - 2 = 17 \text{ bits}\end{aligned}$$

Lastly, each blocks should have 1 bit as the valid bit.

$$\begin{aligned}&(The\ tag\ field\ size) = \{(tag) + (valid\ bit)\} \times (The\ total\ number\ of\ blocks)\\&= (17 + 1) \times 4{,}096 = 73{,}728 \text{ bits} = 9{,}216 \text{ bytes}\end{aligned}$$

$$\begin{aligned}&(The\ total\ cache\ size) = (The\ tag\ field\ size) + (The\ cache\ data\ size) = 9{,}216 + 32{,}768\\&= 41{,}984 \text{ bytes}\end{aligned}$$

When the blocks are changed as 16-word blocks for the second cache, the block offset becomes 4 bits. Since the block offset increases 3 bits, the tag decreases 3 bits. Finally, its tag will be 14 bits. Because one block has 16 words, there are 16 * 4 * 8 = 512 bits for data in one block.

$$(The\ total\ size\ in\ one\ block) = (The\ tag\ field\ size\ in\ one\ block) + (Data\ size\ in\ one\ block) \\ = (14 + 1) + 512 = 527 \text{ bits}$$

So, if the total cache size is the same with the previous cache, the following equation should be established.

$$41{,}984 * 8 = 527 \times (The\ number\ of\ blocks)$$

$$\therefore (The\ number\ of\ blocks) = \frac{41{,}984 \times 8}{527} = 637.33$$

We can know that there will be about 638 blocks in the second cache, but because the number of blocks should be the power of two, it will be 1,024 blocks.

Even though the total cache size becomes larger than before, the total number of blocks is decreased. It means that there can be more conflict misses, and it will make larger miss rate. However, we cannot decide that the miss rate will surely increase because it can have possibility to decrease the miss rate with larger block size. The another reason for lower performance is that it should take more time for processing after selecting a block even though the block hits. Because there are more words in one block, there should be more complex circuits and steps to get the exact word that you want. It can affect to lower performance. Moreover, it should copy larger size of data from the lower level caches or the main memory. It means there is larger miss penalty than the first cache.

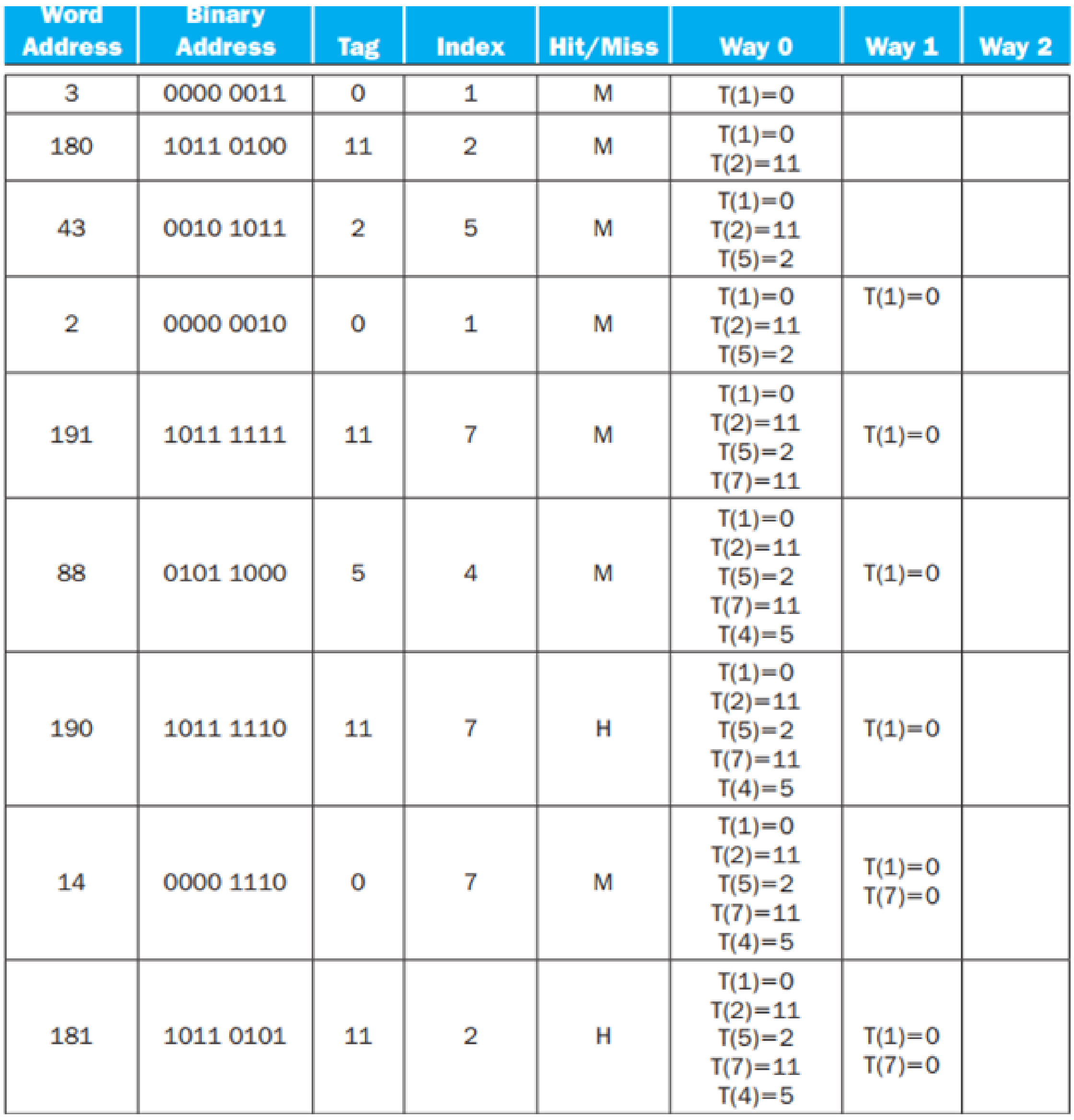
5.7 This exercise examines the impact of different cache designs, specifically comparing associative caches to the direct-mapped caches from section 5.4. For this exercise, use the address stream shown in Exercise 5.2

5.7.1 Using the sequence of addresses given, 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 tag bits, the block offset bits, and if it is a hit or miss

$$: (Total\ size) = (Set\ Num) \times (Associativity) \times (block\ size)$$

$$24\ words = (Set\ Num) \times 3 \times 2\ words$$

$$(Set\ Num) = 4$$



Word Address	Binary Address	Tag	Index	Hit/Miss	Way 0	Way 1	Way 2
3	0000 0011	0	1	M	T(1)=0		
180	1011 0100	11	2	M	T(1)=0 T(2)=11		
43	0010 1011	2	5	M	T(1)=0 T(2)=11 T(5)=2		
2	0000 0010	0	1	M	T(1)=0 T(2)=11 T(5)=2	T(1)=0	
191	1011 1111	11	7	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11	T(1)=0	
88	0101 1000	5	4	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5	T(1)=0	
190	1011 1110	11	7	H	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5	T(1)=0	
14	0000 1110	0	7	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5	T(1)=0 T(7)=0	
181	1011 0101	11	2	H	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5	T(1)=0 T(7)=0	

44	0010 1100	2	6	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5 T(6)=2	T(1)=0 T(7)=0	
186	1011 1010	11	5	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5 T(6)=2	T(1)=0 T(7)=0 T(5)=11	
253	1111 1101	15	6	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5 T(6)=2	T(1)=0 T(7)=0 T(5)=11 T(6)=15	

word addres 3 : binary address : 0000 0011

0 bit is block offset

3 ~ 1 bit is index

7 ~ 4 bit is tag

T(1) = 0 means that the tag of index 1 is 0

T(x) = y menas that the tag of index y is x.

5.7.2 Using the references given, 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 tag bits, and if it is a hit or miss.

: The fully associaitivity cache : 1 sets / block size = 1 words, block per set = 8

Tag	Hit/Miss	Contents
3	M	3
180	M	3, 180
43	M	3, 180, 43
2	M	3, 180, 43, 2
191	M	3, 180, 43, 2, 191
88	M	3, 180, 43, 2, 191, 88
190	M	3, 180, 43, 2, 191, 88, 190
14	M	3, 180, 43, 2, 191, 88, 190, 14
181	M	181, 180, 43, 2, 191, 88, 190, 14
44	M	181, 44, 43, 2, 191, 88, 190, 14
186	M	181, 44, 186, 2, 191, 88, 190, 14
253	M	181, 44, 186, 253, 191, 88, 190, 14

5.7.3 Using the references from Exercise 5.2, what is the miss rate for a fully associative cache with two-word blocks and a total size of 8 words, using LRU replacement? What is the miss rate using MRU (most recently used) replacement? Finally what is the best possible miss rate for this cache, given any replacement policy?

Multilevel caching is an important technique to overcome the limited amount of space that a fi rst level cache can provide while still maintaining its speed. Consider a processor with the following parameters

: The fully associaitivity cache : 1 sets / block size = 2 words, block per set = 4

LRU policy

The content represents the tag of the block according to the access time.

Address	Tag	Hit/Miss	Contents
3	1	M	1
180	90	M	90,1
43	21	M	21,90,1
2	1	H	1,21,90
191	95	M	95,1,21,90
88	44	M	44,95,1,21
190	95	H	95,44,1,21
14	7	M	7,95,44,1
181	90	M	90,7,95,44
44	22	M	22,90,7,95
186	143	M	143,22,90,7
253	126	M	126,143,22,90

Miss rate : 10/12

MRU policy

Address	Tag	Hit/Miss	Contents
3	1	M	1
180	90	M	90,1
43	21	M	21,90,1
2	1	H	1,21,90
191	95	M	95,1,21,90
88	44	M	44,1,21,90
190	95	M	95,1,21,90
14	7	M	7,1,21,90
181	90	H	90,7,1,21
44	22	M	22,7,1,21
186	143	M	143,7,1,21
253	126	M	126,7,1,21

Miss rate : 10/12

Since the tag 1, 90 ,95 are the only ones who reuses again. Let's consider the access sequence of the tag 1, 90, 95

1 -> 90 -> 1 -> 95 -> 95 -> 90

The only thing we have to do is ensure the reuse block is not evicted before reaccess. Since the block per set is 4, the cache can hold the tag 1, 90, 95 without conflict miss between reuse blocks.

Best possible miss rate : 9/12

5.7.4 Calculate the CPI for the processor in the table using: 1) only a first level cache, 2) a second level direct-mapped cache, and 3) a second level eight-way set associative cache. How do these numbers change if main memory access time is doubled? If it is cut in half?

: **Only a first level cache**

$$\begin{aligned} CPI_{total} &= CPI_{base} + CPI_{memory\ stalls} \\ &= CPI_{base} + (L1\ miss\ rate) \times (Main\ memory\ access) \\ &= 1.5 + 0.07 \times 100ns \times 2Ghz = 1.5 + 14 = 15.5 \end{aligned}$$

Main memory doubled: $CPI_{total} = 1.5 + 0.07 \times 200ns \times 2Ghz = 29.5$

Main memory halved: $CPI_{total} = 1.5 + 0.07 \times 50ns \times 2Ghz = 8.5$

First level and second level direct-mapped cache

$$\begin{aligned} CPI_{total} &= CPI_{base} + CPI_{memory\ stalls} \\ &= CPI_{base} + (L1\ miss\ rate) \times (L2\ access) + (L2\ miss\ rate) \times (Main\ memory\ access) \\ &= 1.5 + 0.07 \times 12 + 0.035 \times 100ns \times 2Ghz = 9.34 \end{aligned}$$

Main memory doubled: $CPI_{total} = 1.5 + 0.07 \times 12 + 0.035 \times 200ns \times 2Ghz = 16.34$

Main memory halved: $CPI_{total} = 1.5 + 0.07 \times 12 + 0.035 \times 50 \times 2Ghz = 5.84$

First level and second level eight-way set associative

$$\begin{aligned} CPI_{total} &= CPI_{base} + CPI_{memory\ stalls} \\ &= CPI_{base} + (L1\ miss\ rate) \times (L2\ access) + (L2\ miss\ rate) \times (Main\ memory\ access) \\ &= 1.5 + 0.07 \times 28 + 0.015 \times 100ns \times 2Ghz = 6.46 \end{aligned}$$

Main memory doubled: $CPI_{total} = 1.5 + 0.07 \times 28 + 0.015 \times 200ns \times 2Ghz = 9.46$

Main memory halved: $CPI_{total} = 1.5 + 0.07 \times 28 + 0.015 \times 50ns \times 2Ghz = 4.96$

5.11 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 4 KiB 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

4669, 2227, 13916, 34587, 48870, 12608, 49225

Initial TLB:

Valid	Tag	Physical Page Number
1	11	12
1	7	4
1	3	6
0	4	9

Initial Page Table:

Valid	Physical Page or in Disk
1	5
0	Disk
0	Disk
1	6
1	9
1	11
0	Disk
1	4
0	Disk
0	Disk
1	3
1	12

5.11.1 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.

: For each access in the stream, the table on the next page describe the corresponding **virtual page number**, **hit/miss in the TLB**, **hit/miss in the page table (PT)**, **page fault (PF)**, and the **state of TLB** after the access. The final state of the system is the state of the TLB and PT after the last access.

A virtual page number can be obtained by a modulo operation:
(virtual page number) = (address) mod (page size in bytes)

Note that since TLB uses LRU as its replacement policy, the order of the last access for each entry is marked along with the valid bit. Because there is no given information about the recency of access for existing entries in the initial state of TLB, assumed that the entry placed upper place is accessed less recently.

※ Here, the concept of '**page table hit (miss)**' is considered as the case when the entry corresponding to the accessing virtual address does (not) exist in the page table. But note that this is not a practical concept since the page table should cover the entire virtual address space that the system provides. Even for the case of multi-level page table, if the PTE for the page table which supposed to map a certain virtual address is invalid in the upper level page table, it is considered as a page fault in the upper level page table, rather than a 'page table miss'.

5.11.2 Repeat 5.11.1, but this time use 16 KiB pages instead of 4 KiB pages. What would be some of the advantages of having a larger page size? What are some of the disadvantages?

: The table to describe each access is on the next page. It is notable that a larger page size can reduce TLB miss rate. However the disadvantage is that it can incur higher internal fragmentation of a physical pages, means lower utilization of the physical memory.

Address	Virtual Page	TLB H/M	TLB		
			Valid	Tag	Physical Page
4669	0	TLB miss PT hit	1	11	12
			1	7	4
			1	3	6
			1 (last access 0)	0	5
2227	0	TLB hit	1	11	12
			1	7	4
			1	3	6
			1 (last access 1)	0	5
13916	0	TLB hit	1	11	12
			1	7	4
			1	3	6
			1 (last access 2)	0	5
34587	2	TLB miss PT hit PF	1 (last access 3)	2	13
			1	7	4
			1	3	6
			1 (last access 2)	0	5
48870	2	TLB hit	1 (last access 4)	2	13
			1	7	4
			1	3	6
			1 (last access 2)	0	5
12608	0	TLB hit	1 (last access 4)	2	13
			1	7	4
			1	3	6
			1 (last access 5)	0	5
49225	3	TLB hit	1 (last access 4)	2	13
			1	7	4
			1 (last axxess 6)	3	6
			1 (last access 5)	0	5

Page table after the last access:

Valid	Physical Page or in Disk
1	5
1	Disk
0	13
1	6
1	9
1	11
0	Disk
1	4
1	Disk
0	Disk
1	3
1	12

5.11.3 Show the final contents of the TLB if it is 2-way set associative. Also show the contents of the TLB if it is direct mapped. Discuss the importance of having a TLB to high performance. How would virtual memory accesses be handled if there were no TLB?

: The tables for the two associativity configurations are on the next page.

For the case of 2-way set associative, since the TLB has two sets (indices), a single bit (the least significant bit) of virtual page number is used for indexing (addressing) the TLB entry and the reminder becomes the tag for TLB entry.

For the case of direct mapping, since the TLB has four sets, two least significant bits are used for TLB indexing and the remainder becomes the tag for TLB entry.

Page table after the last access is identical to that of **5.11.1** since there are no difference in page size and access order. The configuration of TLB does not affect to the page table content itself since it is just a translation cache and does not influence to the mapping between virtual address and physical address.

If there is no TLB, all virtual address transformation will require extra memory access for referencing the page table and that means virtual memory references could require two memory accesses (cache miss, single level page table; for multi-level, it could be worse). Obviously, this will increase the memory access latency significantly and the role of TLB is reducing the memory access for referencing the page table so that it can minimize the additional latency due to the usage of virtual memory.

2-way set associative:

Address	Virtual Page	Tag	Index	TLB H/M	TLB			
					Valid	Tag	Physical Page	Index
4669	1	0	1	TLB miss PT hit PF	1	11	12	0
					1	7	4	1
					1	3	6	0
					1 (last access 0)	0	13	1
2227	0	0	0	TLB miss PT hit	1 (last access 1)	0	5	0
					1	7	4	1
					1	3	6	0
					1 (last access 0)	0	13	1
13916	3	1	1	TLB miss PT hit	1 (last access 1)	0	5	0
					1 (last access 2)	1	6	1
					1	3	6	0
					1 (last access 0)	1	13	1
34587	8	4	0	TLB miss PT hit PF	1 (last access 1)	0	5	0
					1 (last access 2)	1	6	1
					1 (last access 3)	4	14	0
					1 (last access 0)	1	13	1
48870	11	5	1	TLB miss PT hit	1 (last access 1)	0	5	0
					1 (last access 2)	1	6	1
					1 (last access 3)	4	14	0
					1 (last access 4)	5	12	1
12608	3	1	1	TLB hit	1 (last access 1)	0	5	0
					1 (last access 5)	1	6	1
					1 (last access 3)	4	14	0
					1 (last access 4)	5	12	1
49225	12	6	0	TLB miss PT miss	1 (last access 6)	6	15	0
					1 (last access 5)	1	6	1
					1 (last access 3)	4	14	0
					1 (last access 4)	5	12	1

Direct mapped:

Address	Virtual Page	Tag	Index	TLB H/M	TLB Valid	TLB Tag	TLB Physical Page	TLB Index
4669	1	0	1	TLB miss PT hit PF	1	11	12	0
					1	0	13	1
					1	3	6	2
					0	4	9	3
2227	0	0	0	TLB miss PT hit	1	0	5	0
					1	0	13	1
					1	3	6	2
					0	4	9	3
13916	3	0	3	TLB miss PT hit	1	0	5	0
					1	0	13	1
					1	3	6	2
					1	0	6	3
34587	8	2	0	TLB miss PT hit PF	1	2	14	0
					1	0	13	1
					1	3	6	2
					1	0	6	3
48870	11	2	3	TLB miss PT hit	1	2	14	0
					1	0	13	1
					1	3	6	2
					1	2	12	3
12608	3	0	3	TLB miss PT hit	1	2	14	0
					1	0	13	1
					1	3	6	2
					1	0	6	3
49225	12	3	0	TLB miss PT miss	1	3	15	0
					1	0	13	1
					1	3	6	2
					1	0	6	3