

Name: \_\_\_\_\_

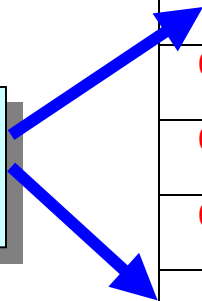
**Problem 1 (20%).** Given the following **direct-mapped** cache architecture:  
 All instructions use byte addresses. The address bus is 8 bits. A word size is 16 bits.  
 Total data cache size 4 words.

- 1a. (2%) How many bits is the index?  
 =  $\log_2 \text{cache size in words} = \log_2 4 = 2$
- 1b. (2%) How many bits is the byte offset?  
 =  $\log_2 \text{number of bytes per word} = \log_2 \text{word size}/8 = \log_2 16/8 = \log_2 2 = 1$
- 1c. (2%) How many bits is the tag size?  
 =  $\text{address size} - \text{index size} - \text{byte offset size} = 8 - 2 - 1 = 5$

1d. (14%) For the following instruction sequence, fill in the access bits to the **data cache**

tag	index	byte offset	instruction
00100	00	0	lw \$1, 32(\$0)
00100	01	0	lw \$2, 34(\$0)
			add \$3, \$1, \$2
00101	10	0	sw \$3, 44(\$0)
00001	10	0	lw \$4, 12(\$0)
00101	00	1	lbu \$5, 41(\$0)
			beq \$1, \$2, 34(\$0)

Does not access **data memory**. Therefore does not access the data cache!



**Problem 2 (10%).** Given the following **2-way set associative** cache architecture:  
 All instructions use byte addresses. The address bus is 8 bits. A word size is 16 bits.  
 Total data cache size 4 words.

- 2a. (2%) How many bits is the index?  
 =  $\log_2 (\text{cache size in words} / \text{N-way}) = \log_2 (4/2) = \log_2 2 = 1$
- 2b. (2%) How many bits is the byte offset?  
 =  $\log_2 \text{number of bytes per word} = \log_2 \text{word size}/8 = \log_2 16/8 = \log_2 2 = 1$
- 2c. (2%) How many bits is the tag size?  
 =  $\text{address size} - \text{index size} - \text{byte offset size} = 8 - 1 - 1 = 6$

2d. (4%) For the following instruction sequence, fill in the access bits to the **data cache**

tag	index	byte offset	instruction
001000	0	0	lw \$1, 32(\$0)
001000	0	1	lbu \$2, 33(\$0)

**Problem 3.** (15%) For the following instruction sequence fill in the direct-mapped cache  
 The word size is 16 bits.  
 Memory[0]=0x1066; Memory[2]=0x1453; Memory[16]=\$3=0x1776; Memory[20]=0x1914;

**3a.** (5%) Fill in the miss cache column.

tag	index	byte offset	instruction	Valid or Tag Cache Miss?
000	00	0	lw \$1, 0(\$0)	yes - valid
000	01	0	lw \$2, 2(\$0)	yes - valid
010	00	0	sw \$3, 16(\$0)	
010	10	1	lbu \$5, 21(\$0)	yes
010	10	0	lw \$6, 20(\$0)	

**3b.** (10%) The final state of the direct mapped data cache is:

index	valid	tag	data
00	N→Y	000→010	0x1066→0x1776
01	N→Y	000	0x1453
10	N→Y	010	0x1914
11	N		

A store never creates a cache miss! Cache misses are a property of reads

Load byte must first read the word from memory (20) and then separates out the byte.

**Problem 4.** (15%) For the following instruction sequence fill in the 2-way set associative LRU cache  
 The word size is 16 bits.  
 Memory[0]=0x1066; Memory[2]=0x1453; Memory[16]=\$3=0x1776; Memory[20]=0x1914;

**4a.** (5%) Fill in the miss cache column.

tag	index	byte offset	instruction	Valid or tag Cache Miss?
000	0	0	lw \$1, 0(\$0)	yes
100	0	0	sw \$3, 16(\$0)	
000	0	1	lbu \$5, 1(\$0)	
101	0	0	lw \$6, 20(\$0)	yes - flush oldest reference
000	1	0	lw \$1, 2(\$0)	yes

**4b.** (10%) The final state of the 2-way set associative LRU data cache is:

index	valid	tag	data
0	N→Y	000	0x1066 (accessed by lw and lbu)
	N→Y	100→101	0x1776→0x1914
1	N→Y	000	0x1453
	N		

**Problem 5. (10%)** Given a 1-word cache entry block size and one word wide memory bus organization (figure 7.13a, page 561), and the following access times:

- 1 clock cycle to send the address,
- 8 clock cycles to read access DRAM, 16 clock cycles to write to DRAM
- 1 clock cycle to to send a word

5a. (5%) What is the miss penalty for a write-through direct mapped cache?

**The miss penalty only include data reads not writes, since write-through is handled on the store instruction.**

**Miss penalty = (1 send address) + (8 clocks to read) + (1 send word) = 10 clock cycles**

5b. (5%) What is the miss penalty for a write-back direct mapped cache?

**This miss penalty includes only data reads BUT on write back the cache entry may contain a block not previously written out before a new entry can be read in (page 554).**

**Miss penalty = (1 send address) +(1 send word ) + 1x(16 clocks to write)  
+ (1 send address) + (8 clocks to read) + (1 clock to send word) = 28 clocks**

**Problem 6 (20%).** Given the following virtual memory architecture:

All instructions use byte addresses. The virtual address bus is 16 bits. A word is 16 bits.  
Total page size 8 bytes. The real memory address bus is 16 bits.

6a. (2%) How many bits is the page offset?

**=  $\log_2$  number of bytes per page =  $\log_2 8 = \log_2 2^3 = 3$**

6b. (2%) How many bits is the physical page number size?

**= virtual address bit size – page offset bit size = 16 – 3 = 13**

6c. (2%) How many page table entries?

**= virtual address size/page size =  $2^{16} / 2^3 = 2^{13}$**

6d. (2%) How large is the page table?

**= #entries × (bytes per entry to hold the real memory address)  
=  $2^{13} \times 2^1 = 2^{14}$  bytes = 16,384 bytes (i.e. a quarter the size of memory!)**

6e. (12%) For the following instruction sequence, fill in the data access bits to the **page table**

virtual page number	page offset	instruction
0000 0000 00100	000	lw \$1, 32(\$0)
0000 0000 00100	010	lw \$2, 34(\$0)
0000 0000 00101	001	lbu \$5, 41(\$0)

**Problem 7. (10%)** Assume 2K real memory, LRU, a page size of 1K and no pages loaded.

Fill in the page fault columns

instruction	Page fault?	Flush which page?	Write flushed page to disk?	Load what new page
lw \$1, 32(\$0)	yes			0..1023
lw \$2, 34(\$0)	no			
lbu \$5, 41(\$0)	no			
sw \$6, 1100(\$0)	yes			1024..2047
lw \$7, 2000(\$0)	no			
lw \$1, 4100(\$0)	yes	0..1023 (oldest)	No – only lw's	4096..5119
lw \$2, 32(\$0)	yes	1024..2047 (oldest)	Yes – had a sw	0..1023