Name: _____

Problem 1 (15%). Given the following "spatial" **direct-mapped** cache with 2 word blocks:
All instructions use byte addresses. The address bus is 10 bits. A word size is 16 bits.
Each cache entry (i.e. block size) contains 2 words. Total data cache size 16 words.

1a. (5%) How many bits is the index?
= $\log_2$ cache size in blocks = $\log_2$ 8 = 3
1b. (2%) How many bits is the byte offset?
= $\log_2$ number of bytes per block = $\log_2$ word size/8 = $\log_2$ 16/8 = $\log_2$ 2 = 1
1c. (1%) How many bits is the tag size?
= address size – index size – byte offset size = 10 – 3 – 1 = 6

1d. (7%) For the following instruction sequence, fill in the access bits to the **data cache**

tag	index	byte offset	instruction
000 001	**000**	**0**	**lw $1, 16($0)**
000 001	**110**	**0**	**sw $3, 24($0)**
			lui $1, 12
000 000	**111**	**1**	**lbu $5, 15($0)**
			sltiu $1, $2, 32

Problem 2 (15%). Given the following **4-way set associative** cache architecture:
All instructions use byte addresses. The address bus is 10 bits. A word size is 32 bits.
Total data cache size 16 words.

2a. (5%) How many bits is the index?
= $\log_2$ (cache size in blocks / N-way) = $\log_2$ (16/4) = $\log_2$ 4= 2
2b. (2%) How many bits is the byte offset?
= $\log_2$ number of bytes per word = $\log_2$ word size/8 = $\log_2$ 32/8 = $\log_2$ 4 = 2
2c. (1%) How many bits is the tag size?
= address size – index size – byte offset size = 10 – 2 – 2 = 6

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

tag	index	byte offset	instruction
000 000	**11**	**00**	lw $1, 12($0)
000 000	**11**	**01**	lbu $2, 13($0)

2f. (5%) Increase the cache block size from 1 word to 2. How many bits is in the index?
= $\log_2$ (cache size in blocks / N-way) = $\log_2$ (8/4) = $\log_2$ 2= 1

Problem 3. (15%) For the following instruction sequence fill in the direct-mapped cache
The word size is 16 bits.
Memory[0]=0x1066; Memory[8]=0x1453; Memory[16]=$3=0x1776; Memory[24]=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**
011	00	0	lw $2, 24($0)	**yes – tag**
010	00	0	sw $3, 16($0)	
010	00	1	lbu $5, 17($0)	
001	00	0	lw $6, 8($0)	**yes - tag**

3b. (10%) Show **all states** and **underline the final state** of the direct mapped data cache:

index	valid	tag	data
00	**N→Y**	**000→011→010→001**	**0x1066→0x1914→0x1776→0x1453**
01	**N**		
10	**N**		
11	**N**		

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[8]=0x1453; Memory[16]=$3=0x1776; Memory[24]=0x1914;

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

tag	index	byte offset	instruction	Valid or tag Cache Miss?
0000	0	0	lw $1, 0($0)	**yes**
0110	0	0	lw $2, 24($0)	**yes**
0100	0	0	sw $3, 16($0)	**no – flush oldest reference**
0100	0	1	lbu $5, 17($0)	
0010	0	0	lw $6, 8($0)	**yes– flush oldest reference**

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

index	valid	tag	data
0	**N→Y**	**0000→0100**	**0x1066→0x1776**
	N→Y	**0110→0010**	**0x1914→0x1453**
1	**N**		
	N		

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

2 clock cycle to send the address,
8 clock cycles to read access DRAM, 16 clock cycles to write to DRAM
3 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 = 2×(2 send address) + 2×(8 clocks to read) + 2×(3 send word) = 26 clocks

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 = 2×(2 send address) + 2×(3 send word) + 2×(16 clocks to write)
+ 2×(2 send address) + 2×(8 clocks to read) + 2×(3 clock to send word) = 68 clocks

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

All instructions use byte addresses. The virtual address bus is 20 bits. A word is 16 bits.
Total page size 16 bytes. The real memory address bus is 16 bits.

6a. (4%) How many bits is the page offset?

= $\log_2$ number of bytes per page = $\log_2 16 = \log_2 2^4 = 4$

6b. (4%) How many bits is the physical page number size?

= virtual address bit size – page offset bit size = 20 – 4 = 16

6c. (4%) How many page table entries?

= virtual address size/page size = $2^{20} / 2^4 = 2^{16}$

6d. (4%) How large is the page table?

= #entries × (bytes per entry to hold the real memory address)
= $2^{16} \times 2^1 = 2^{17}$ bytes = 131,072 bytes

6e. (4%) For the following instruction sequence, fill in the data access bits to the **page table**

virtual page number	page offset	instruction
0000 0000 0000 0000	**1000**	sw $1, 16($0)
0000 0000 0000 0001	**0000**	lw $2, 32($0)
0000 0000 0000 0001	**0001**	lbu $5, 33($0)

Problem 7. (10%) Assume 2048 bytes of real memory, LRU, a page size of 1024 bytes and no pages loaded in memory. Fill in the page fault columns. (Blank space implies No)

instruction	Page fault?	Flush which page?	Write flushed page to disk?	Load what new page
lw $1, 0($0)	**yes**			**0..1023**
lw $2, 1024($0)	**yes**			**1024..2047**
lbu $5, 0($0)				
sw $6, 2048($0)	**yes**	**1024..2047**		**2048..3071**
lw $7, 1024($0)	**yes**	**0..1023**		**1024..2047**
lw $1, 0($0)	**yes**	**2048..3071**	**Yes**	**0..1023**
lw $2, 1032($0)				