CS31 Written Homework 4

In this assignment, you will be tracing memory accesses in a system with the following architecture:

- 8-bit addresses
- 2-byte cache lines
- 16-byte page size
- 4-line direct-mapped L1 cache
- 4-set 2-way set associative L2 cache
- 4 pages of physical RAM

Question 1

How many bytes of data can a single process store in each level of the memory hierarchy?

(a) L1 cache (b) L2 cache (c) physical memory (d) virtual memory

Question 2

For each level of cache, divide the address bits into the `tag`, `index`, and `byte offset`. In part (c) divide the address into the `page number` and `page offset`.

(a) L1 cache

`1 0 1 0 1 0 1 0`

(b) L2 cache

`1 0 1 0 1 0 1 0`

(c) memory

`1 0 1 0 1 0 1 0`

Question 3

On the accompanying L1 cache diagram, show the results of the following memory operations. Within each box, time should progress downward, so the first address loaded appears at the top and subsequent changes are written below. To the right of the table, label each change with number of the operation that caused it. Annotate each operation below with *hit* or *miss* to indicate whether the data was found in L1 cache. Don't forget to update the dirty and valid bits!

1. `read  1 1 0 0 0 0 1 0`
2. `write 1 1 0 0 0 0 1 1`
3. `read  1 1 0 1 1 1 0 0`
4. `read  1 1 0 1 1 1 1 0`
5. `read  0 1 0 0 1 1 0 0`
6. `write 0 1 0 0 0 0 0 1`
7. `read  0 0 0 0 0 0 0 0`
8. `read  1 0 1 0 1 0 1 1`
9. `write 1 1 0 0 0 0 1 0`
10. `read  1 1 1 0 0 0 0 1`

Question 4

On the accompanying L2 cache diagram, show the results of the following memory operations. Assume LRU replacement within each set, and note the initial values of the LRU bits. Within each box, time should progress downward, so the first address loaded appears at the top and subsequent changes are written below. To the right of the table, label each change with number of the operation that caused it. Annotate each operation below with *hit* or *miss* to indicate whether the data was found in L2 cache. Don't forget to update the dirty, valid, and LRU bits!

1. `read  1 1 0 0 0 0 1 0`
2. `write 1 1 0 0 0 0 1 1`
3. `read  1 1 0 1 1 1 0 0`
4. `read  1 1 0 1 1 1 1 0`
5. `read  0 1 0 0 1 1 0 0`
6. `write 0 1 0 0 0 0 0 1`
7. `read  0 0 0 0 0 0 0 0`
8. `read  1 0 1 0 1 0 1 1`
9. `write 1 1 0 0 0 0 1 0`
10. `read  1 1 1 0 0 0 0 1`

Question 5

On the accompanying Page Table diagram, show the results of the following memory operations. Assume first-in-first-out replacement. Within each box, time should progress downward, so the first address loaded appears at the top and subsequent changes are written below. To the right of the table, label each change with number of the operation that caused it. Annotate each operation below with *hit* or *page fault* to indicate whether the data was found in physical memory. Don't forget to update the valid bits, especially when a page is kicked out!

1. `read  1 1 0 0 0 0 1 0`
2. `write 1 1 0 0 0 0 1 1`
3. `read  1 1 0 1 1 1 0 0`
4. `read  1 1 0 1 1 1 1 0`
5. `read  0 1 0 0 1 1 0 0`
6. `write 0 1 0 0 0 0 0 1`
7. `read  0 0 0 0 0 0 0 0`
8. `read  1 0 1 0 1 0 1 1`
9. `write 1 1 0 0 0 0 1 0`
10. `read  1 1 1 0 0 0 0 1`

L1 Cache

index	dirty	valid	tag
0		0	
1		0	
2		0	
3		0	

L2 Cache

set	LRU	D	V	tag	D	V	tag
0	1		0			0	
1	0		0			0	
2	0		0			0	
3	1		0			0	

Page Table

index	V	frame
0	0	
...	...	...
4	0	
...	...	...
10	0	
...	...	...
12	0	
13	0	
14	0	