CS 31 Homework 6: Direct Mapped and Set Associative caching

(due Tuesday, November 21)

Names, Usernames, and Lab Sections:

Part 1: Direct Mapped Caches

In the first part of this assignment, you will answer questions about **direct mapped** caches. For these qustions, assume a system with:

- 8-bit memory addresses
- 2-byte cache block size
- 4 cache lines

Question 1.1

How many bytes of data can be stored in the cache? Do not include metadata like the tag or valid bit.

Question 1.2

How are the tag bits used in a direct mapped cache? How are the byte offset bits used in a direct mapped cache?

Question 1.3

Divide the following two sets of address bits into the `tag` and `byte offset`.

`1 0 1 0 1 1 0 0` `0 0 1 1 1 1 0 1`

Question 1.4

On the diagram of the Direct Mapped Cache on the next page, show the results of the following memory operations (R: read, W: write). 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 the cache. Don't forget to update the dirty and valid bits!

1. `R 0 0 1 0 1 0 1 0`
2. `W 0 0 1 0 1 0 1 1`
3. `R 1 1 0 0 1 1 0 0`
4. `R 1 1 0 0 1 0 1 0`
5. `R 1 1 0 0 1 0 1 1`
6. `W 0 0 1 1 1 1 0 1`
7. `R 0 0 1 1 1 1 0 0`
8. `W 0 0 1 0 1 1 0 0`

Table 1: Direct Mapped Cache

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

Part 2: Set Associative Caches

In the second part of this assignment, you will answer questions about **set associative** caches. For these qustions, assume a system with:

- 8-bit memory addresses
- 2-byte cache block size
- The cache is 2-way set associative with a total of 4 sets

Question 2.1

How many bytes of data can be stored in the cache? Do not include metadata like the tag, dirty bit, valid bit, or LRU bit.

Question 2.2

How are the index bits used in a set associative cache?

How are the tag bits used in a set associative cache?

Question 2.3

Divide the following two addresses into their `tag`, `index` and `byte offset` parts.

`1 0 1 0 1 0 1 0` `0 0 1 1 1 0 1 1`

Question 2.4

On the diagram of the set associative cache on the next page, show the results of the following memory operations (R: read, W: write). 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 block was found in the cache. Don't forget to update the dirty, valid, and LRU bits!

Assume that an LRU value of 0 means the left line in the set was least recently used and that 1 means the right line was used least recently.

1. `R 0 0 0 1 1 0 1 0`
2. `W 0 0 0 1 1 0 1 1`
3. `R 1 1 1 1 1 0 0 0`
4. `R 1 1 1 1 1 0 1 0`
5. `R 0 1 1 0 1 0 0 0`
6. `W 0 0 0 0 1 0 0 1`
7. `R 0 0 0 0 0 0 0 0`
8. `W 0 0 0 1 1 0 1 0`

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