CS318 Spring 2018 Lab 8 (the last lab!)
Memory

Due date: Monday, April 30 at 10:10 AM
Upload your solutions to Blackboard as a PDF file.

Instructions:

- Answer all questions.
- Neatly type your solutions.
- Convert your solution file to PDF and upload the PDF file to the Blackboard assignment.

Problem 1: Cache and Main Memory Design ***[40 points: 8 points per sub-problem]***
Given the following information about a two-way set associative cache:

- 10 bits are used to specify the cache set (unique address for each set of lines in cache)
- 12 bits are used to specify the tag (identifier for a block of memory)
- The block size is 4 bytes (data is transferred between main memory and cache in groups of 4 bytes)

  a) How many sets are in this cache?
  2^10 sets

  b) How many lines are in this cache?
  2 lines per set *2^10 sets = 2^11 lines.

  c) What is the size of the data that can be stored in this cache in bytes?
  2^11 lines * 4 bytes per line = 2^13 bytes

  d) How many blocks are in main memory?
  2 ^ (22 bits per address) = 2^22 blocks

  e) What is the size of main memory in bytes?
  2^2 Bytes per block*2^22 blocks= 2^24 bytes

Problem 2: Cache and Main Memory Design ***[40 points: 8 points per sub-problem]***
Use the following system properties for this problem:

- Total size of main memory = 4 GiB
- Total size of data in cache = 32 MiB
- Block size = size of data in each cache line = 16 bytes. This means that 16 bytes of main memory (16 main memory addresses) are transferred into cache at one time.

  a) How many bits are needed to specify a unique address for each byte of main memory?
  4GiB = 2^32 bytes, so a unique address would require 32 bits.

  b) Using the block size stated above, how many bits are needed to specify a unique address for each block of main memory?
  2^2 * 2^30 bytes in memory / 2^4 bytes per block = 2^28 blocks, so a 28 bit address is needed.

  c) Assuming a direct-mapped cache: how many bits are needed for the cache index (unique address for each cache line), and how many bits are used for the tag (the tag will identify a block of memory)?

2^6 * 2^20 bytes in cache / 2^4 bytes per line = 2 ^ 22 lines, so 22 bits for the cache index and 6 bits for the tag

d) Assuming a four-way set associative cache: how many bits are needed for the set index (unique address for each cache set), and how many bits are used for the tag?
2^22 lines / 4 lines per set = 2^20 sets, so 20 bits for the set index, and 8 bits for the tag

e) Assuming a sixteen-way set associative cache: how many bits are needed for the set index, and how many bits are used for the tag?
2^22 lines / 2^4 lines per set = 2^18 sets, so 18 bits for the set index, and 10 bits for the tag

Problem 3: Cache Mapping *[20 points: 5 points per instruction]*
Demonstrate the use of a two-way set associative cache. This cache uses the Least Recently Used (LRU) algorithm when deciding which line in the set should be replaced.

You are provided with the following information:

- The table titled "Instruction List" contains the four instructions that you will use for this demonstration.
- The table titled "Two-Way Set Associative Cache" shows the current state of the cache.
- The table titled "Memory" is the main memory that is referred to by the "Instruction List" table.

Update the table titled "Two-Way Set Associative Cache" to show the state of this cache after the four instructions in the table titled "Instruction List" have been executed by the processor. In the "Instruction List" table, indicate if each instruction results in a cache hit or miss. To make updates to the "Two-Way Set Associative Cache" table, draw a line through the old information then write in the new information. The memory referred to by these instructions is shown in the table titled "Memory".

The shaded cells in the "Instruction List" and "Two-Way Set Associative Cache" tables are the only table cells that you may alter in this problem.

Memory	
Address	**Contents**
0000	0000 1011
0001	0000 1100
0010	0000 1110
0011	0101 1011
0100	0011 1001
0101	1110 0010
0110	0111 0010
0111	0100 0011
1000	1101 1001
1001	0011 0000
1010	0110 0110
1011	1100 0110
1100	0000 0011
1101	1100 1011
1110	1111 0110
1111	0101 0001

Instruction List			
Number	**Operation**	**Mem. Address**	**Hit or Miss?**
11	READ	1000	miss
12	READ	1011	miss
13	READ	0000	hit
14	READ	1111	miss

Two-Way Set Associative Cache
(contains data from previously executed instructions)

Set	Valid	Tag	Last	Data
00	Y	~~01~~ 10	~~2~~ 11	~~00111001~~ 1101 1001
	Y	00	7	00001011
01	Y	00	1	00001100
	Y	11	8	11001011
10	Y	01	3	01110010
	Y	10	4	01100110
11	Y	~~11~~ 10	~~5~~ 12	~~01010001~~ 1100 1011
	Y	~~01~~ 11	~~9~~ 14	~~01000011~~ 0101 0001