Module 10 Assignment – Direct-mapped Cache

Part 1: Calculating a Direct-Mapped Cache

In this question you're given a 16-byte memory segment and an 8-byte cache. Given the following series of memory accesses, complete the table below. Use the first Contents/Tag column to insert an item to the cache the first time and use the second Contents/Tag column if a cache entry is overwritten. Note: no index will have more than two blocks mapped to it. The first two examples have been provided.

How many cache hits occurs given the memory accesses? How many cache misses are there?

Additionally, what is the name of the cache eviction policy being used here? Namely, our strategy is to overwrite existing elements in the cache if cache slot is already full. In 1-2 paragraphs, discuss the performance implications (advantages/disadvantages) of using this eviction heuristic.

Memory access: ***0x48, 0x4C, 0x46, 0x4D, 0x42, 0x4B, 0x45, 0x41, 0x48, 0x4F, 0x4A, 0x47, 0x40, 0x43, 0x4C, 0x4E, 0x40, 0x49, 0x47***

Memory Address	**Block Contents**
0x40	3
0x41	a
0x42	9
0x43	R
0x44	!
0x45	8
0x46	p
0x47	d
0x48	J
0x49	7
0x4A	?
0x4B	V
0x4C	s
0x4D	A
0x4E	T
0x4F	4

Index	Contents	Tag	Contents	Tag
000	**J**	**1001**	**3**	**1000**
001	**a**	**1000**	**7**	**1001**
010	**9**	**1000**	**?**	**1001**
011	**V**	**1001**	**R**	**1000**
100	**s**	**1001**	**!**	**1000**
101	**A**	**1001**	**8**	**1000**
110	**p**	**1000**	**T**	**1001**
111	**4**	**1001**	**d**	**1000**

There are 4 cache hits and 15 cache misses. The cache eviction policy being used here is called LRU, as the previously existing elements are overwritten by latest used ones.

The LRU algorithm is based on a heuristic, trying to exploit temporal locality. It assumes that the least recently used cache entry is the cache entry that will be needed at the furthest point in the future. However, this is not always as efficient. The memory block maps to a fixed location in the cache. If two blocks map to the same location in cache and they are alternatively referenced, the two blocks will be continually swapped in and out.

Part 3: Conceptual Questions

1. Provide an example of an input (of at least 10 characters) that a direct-mapped cache (with LRU) will have a 100% miss rate. Explain why this happens. In other words, what are the drawbacks of direct-mapped caches?

Input: JspA9V8a?4
This input, in length 10, will have 100 % miss rate.

Memory Access = 0x48
Memory address in Binary = 0100 1000
Cache Index = 000
Tag = 01001
Cache Miss
Contents from memory = **J**

Memory Access = 0x4C
Memory address in Binary = 0100 1100
Cache Index = 100
Tag = 01001
Cache Miss
Contents from memory = **s**

Memory Access = 0x46
Memory address in Binary = 0100 0110
Cache Index = 110
Tag = 01000
Cache Miss
Contents from memory = **p**

Memory Access = 0x4D
Memory address in Binary = 0100 1101
Cache Index = 101
Tag = 01001
Cache Miss
Contents from memory = **A**

Memory Access = 0x42
Memory address in Binary = 0100 0010
Cache Index = 010
Tag = 01000
Cache Miss
Contents from memory = **9**

Memory Access = 0x4B
Memory address in Binary = 0100 1011
Cache Index = 011
Tag = 01001
Cache Miss

Contents from memory = **V**

Memory Access = 0x45
Memory address in Binary = 0100 0101
Cache Index = 101
Tag = 01000
Cache Miss
Contents from memory = **8**

Memory Access = 0x41
Memory address in Binary = 0100 0001
Cache Index = 001
Tag = 01000
Cache Miss
Contents from memory = **a**

Memory Access = 0x4A
Memory address in Binary = 0100 1010
Cache Index = 010
Tag = 01001
Cache Miss
Contents from memory = **?**

Memory Access = 0x4F
Memory address in Binary = 0100 1111
Cache Index = 111
Tag = 01001
Cache Miss
Contents from memory = **4**

In a direct-mapped cache each addressed location in main memory maps to a single location in cache memory. Since main memory is much larger than cache memory, there are many addresses in main memory that map to the same single location in cache memory. Direct-mapped cache's performance is degraded if two or more blocks that map to the same location are used alternately because of continuous swapping made at the same locations, when referencing is made at the same location. This is know as thrashing. Direct-mapped cache has greater access time than other method.

2. Assume a program of 1000 instructions has 60% data access instructions. A cache hit takes 3 clock cycles to move data from cache into a register and a cache miss takes 24 CPU cycles to fetch the data from RAM and move it into a register. How many CPU cycles do the data access instructions require, in total, if 60% of the data accesses result in cache hits? What is the percentage of cache misses and cache hits?

CPU cycles = Cache hit cycle * data access instructions percentage * data accesses result in cache hits percentage + Cache miss cycle * data access instructions percentage * data accesses result in cache misses percentage
= 1000 * 60% * 60% * 3 + 1000 * 60% * 40% *24 = 1080 + 5760 = 6840 CPU cycles

Percentage of cache misses and cache hits = 60% * 60% / 60% * 40% = 1.5