Alternative Cache Designs Assessment

Purpose

The purpose of this quiz is twofold:

- To provide you with additional opportunities to review and retrieve information from your memory, thereby strengthening these memories,
- To help you measure your level of familiarity and proficiency in the following skills & knowledge that are essential to your success in this course and professional life beyond school,
- Understand alternatives in cache design and their impact on cost/performance.

Question 1

Consider a sequence of memory references of word addresses:

22, 25, 24, 56, 40, 24, 20

Cache size is **8 words** and assume the LRU replacement strategy.
Mark each reference as a hit (h) or miss (m) and show the final content of the cache for different cache organizations.

Direct Mapped, block size: 1 word

Block address	Binary address	Cache index	Hit/miss	Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
22	0001 0**110**	6	Miss							Mem[22]	
25	0001 1**001**	1	Miss		Mem[25]					Mem[22]	
24	0001 1**000**	0	Miss	Mem[24]	Mem[25]					Mem[22]	
56	0011 1**000**	0	Miss	Mem[56]	Mem[25]					Mem[22]	
40	0010 1**000**	0	Miss	Mem[40]	Mem[25]					Mem[22]	
24	0001 1**000**	0	Miss	Mem[24]	Mem[25]					Mem[22]	
20	0001 0**100**	4	Miss	Mem[24]	Mem[25]			Mem[20]		Mem[22]	

1. **Number of cache blocks** = num words/block size = 8 words/1 word = 8 blocks.

2. In binary...
Since **block size** is 1 word, **offset** => 2^0 words => 0 bits.
Since **cache size** is 8 words, **index** => 2^3 words => 3 LSB.

In decimal...
Cache index = (block address) % (number of cache blocks)

3. Use LRU replacement strategy.

Notes from prof:

Number of **index** bits = log (number of cache blocks/sets)
→ depends on number of blocks for DM cache [draw cache as 1D vertical array]
→ depends on number of sets for set associative cache [draw cache as 2D array, each row is a set and has a number, block can go anywhere in a specific row/set]
→ 0 for fully associative [draw cache as 1D horizontal array]
8 blocks = 2^3 blocks → 3 index bits

Number of **offset** bits depends on the block size compared to the type of addressing (independent of cache organization)

Word addresses (number) and each block is 1 word= 2^0 words → no offset bits

Final memory references:
22 (m), 25 (m), 24 (m), 56 (m), 40 (m), 24 (m), 20 (m).

Final cache contents:

Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
Mem[24]	Mem[25]	-	-	Mem[20]	-	Mem[22]	-

2-Way Set Associative, block size: 2 words

Block address	Binary address	Set index (set)	Offset bit ([0,1])	Hit/miss
22	0001 011**0**	1	0	Miss
25	0001 100**1**	0	1	Miss
24	0001 100**0**	0	0	**Hit [24,25]**
56	0011 100**0**	0	0	Miss
40	0010 100**0**	0	0	Miss
24	0001 100**0**	0	0	Miss
20	0001 010**0**	0	0	Miss

Set 0	~~[24,25]~~, ~~[40,41]~~, [20,21]	~~[56,57]~~, [24,25]
Set 1	[22,23]	

1. **Number of cache blocks** = cache size/block size = 8 words/2 words = 4 blocks

2. **Number of sets** = (cache size) / (block size * associativity)
   Number of sets = (8 words) / (2 words * 2 associativity) = 2 sets

3. Since there are 2 (2^1) **words** per block, the **offset** is 1 bit (LSB).

4. Since there are 2 (2^1) **sets**, the **number of index bits** is 1. The cache index is the 2nd-most LSB.

5. Use LRU replacement strategy

Final memory references:
22 (m), 25 (m), 24 (h), 56 (m), 40 (m), 24 (m), 20 (m).

Final cache contents:

Set 0	[20,21]	[24,25]
Set 1	[22,23]	

Fully Associative, block size: 1 word

Block address	Binary address	Hit/miss	Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
22	0001 0110	Miss	Mem[22]							
25	0001 1001	Miss	Mem[22]	Mem[25]	A1					
24	0001 1000	Miss	Mem[22]	Mem[25]	Mem[24]					
56	0011 1000	Miss	Mem[22]	Mem[25]	Mem[24]	Mem[56]				
40	0010 1000	Miss	Mem[22]	Mem[25]	Mem[24]	Mem[56]	Mem[40]			
24	0001 1000	Hit	Mem[22]	Mem[25]	Mem[24]	Mem[56]	Mem[40]			
20	0001 0100	Miss	Mem[22]	Mem[25]	Mem[24]	Mem[56]	Mem[40]	Mem[20]		

Number of cache blocks = 8 words/1 word = 8

Final memory references:
22 (m), 25 (m), 24 (m), 56 (m), 40 (m), 24 (h), 20 (m).

Final cache contents:

Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
Mem[22]	Mem[25]	Mem[24]	Mem[56]	Mem[40]	Mem[20]	-	-