CS152 Section 3

Q1. Cache Organization

Consider a 1 KiB 4-way set-associative cache with 32-byte cache lines. The address is 12 bits wide. How are the address bits partitioned?

- Tag:
- Index:
- Offset

Q2. Replacement Policies

Suppose we see the following stream of accesses where A, B, C, D, and E represent unique addresses from different lines that all map to the same set:

A, B, C, D, B, A, E

Assume the cache has four ways and all lines in the set are initially invalid. When address E is accessed, one of the cache lines must be evicted. For each replacement policy, which line gets evicted?

- FIFO

Way0	Way1	Way2	Way3	Policy State after request

- NMRU

Way0	Way1	Way2	Way3	Policy State after request

- LRU

Way0	Way1	Way2	Way3	Policy State after request

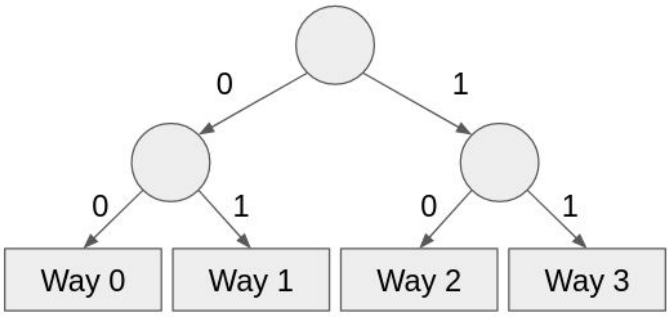
- PLRU

Way0	Way1	Way2	Way3	Policy State

Q3. Pseudo-LRU Replacement

PLRU works by maintaining a binary tree for every cache set, with the leaves representing ways (not stored) and the interior nodes holding directions (each stored as 1 bit). For each cache access, trace the path from the root to the leaf node. At each interior node, set the direction to be the opposite of the path you followed.

Complete the table. Assume cache lines are 32 bytes, there are 8 sets, and the tree starts with all bits set to 0. For simplicity, we consider accesses to sets 0 and 1 only.



	Cache Tags (hex)										
Address (hex)	**Set 0**					**Set 1**					**Hit?**
	Way 0	**Way 1**	**Way 2**	**Way 3**	**PLRU State**	**Way 0**	**Way 1**	**Way 2**	**Way 3**	**PLRU State**	
208	2	inv	inv	inv	110	inv	inv	inv	inv	000	no
22C						2				110	no
41C			4		011						no
604											
320											
214											
310											
50C											
404											

Q4. Cache Optimizations

For each technique, indicate whether implementing it will **increase**, **decrease**, or have **no change** on each aspect.

Technique	Hit Time	Miss Penalty	Miss Rate	Hardware Complexity
Smaller, simpler caches				
Multi-level caches				
Smart replacement policy				
Pipelined writes				
Write buffer				
Sub-blocks (sector cache)				
Code optimization				
Compiler prefetching				
Hardware prefetching (stream buffer)				
Victim cache				