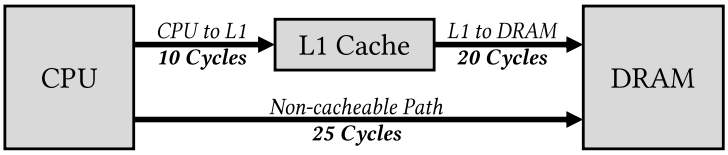
1 Cache Forward Engineering [60 points]

Consider a processor with the following configuration:

- In-order execution.
- Single L1 data cache that:
  - Is fully associative.
  - Has a capacity of 2 cache blocks.
  - Has a programmable replacement policy between LRU and Belady's Algorithm.

The system configuration looks like the following diagram:



There are two memory access paths:

- Through the L1 data cache, resulting in a miss latency of 30 cycles and a hit latency of 10 cycles
- Directly accessing memory (i.e., an *uncached* access) with a fixed latency of 25 cycles. Note that this type of access will bypass the cache entirely and will not affect the cache state.

However, only one path may be used at a time (i.e., all memory accesses are serialized).

You are interested in optimizing the execution time of the following performance-critical code, where LOAD($\mathbf{BLK_n}$) represents a load to the cache block address n:

```
1  set_cache_replacement_policy(DESIRED_POLICY); // LRU or Belady's
2
3  for(i = 0; /* infinite loop */; i = (i + 1) % 5)
4  {
5      if(i < 3)
6          LOAD(BLK1); // Load #1
7      if(i > 0 && i < 4)
8          LOAD(BLK2); // Load #2
9      if(i > 1)
10         LOAD(BLK3); // Load #3
11 }
```

This code results in accesses to three L1 cache block addresses in the following order (organized hierarchically in row-major format with one row per loop iteration for easier understanding):

$$\begin{array}{llllll} \mathbf{BLK_1} & \rightarrow & & & & \\ \mathbf{BLK_1} & \rightarrow & \mathbf{BLK_2} & \rightarrow & & \\ \mathbf{BLK_1} & \rightarrow & \mathbf{BLK_2} & \rightarrow & \mathbf{BLK_3} & \rightarrow \\ & & \mathbf{BLK_2} & \rightarrow & \mathbf{BLK_3} & \rightarrow \\ & & & & \mathbf{BLK_3} & \rightarrow \textbf{repeat} \end{array}$$

In this system, you have two degrees of freedom available to you for optimizing performance:

1. You may replace *any* of the three LOAD instructions on Lines 6, 8, or 10 with *uncached* LOAD (i.e., LOAD_UC) instructions.
2. You may choose between the *LRU* and the *Belady's* cache replacement policies on Line 1. For your reference:
   - The LRU replacement policy evicts the *least-recently used* cache line.
   - The Belady's replacement policy evicts the cache line that will be used *furthest into the future*.

Your task is to determine the *best performing* configuration.

1. [30 points] Fill in the following table assuming steady-state operation. You will need to provide the cache miss rate and the average latency for one iteration of the for loop. Assume that execution time is dominated by memory accesses such that there is no memory-level parallelism and the overall application latency can be computed directly from the memory access latencies.

Uncached Load Instructions	LRU Replacement		Belady's Replacement	
	Cache Miss Rate	Latency / Iteration	Cache Miss Rate	Latency / Iteration
None	3/9	150 Cycles	2/9	130 Cycles
#1	0/9	135 Cycles	0/9	135 Cycles
#2	0/9	135 Cycles	0/9	135 Cycles
#3	0/9	135 Cycles	0/9	135 Cycles
#1, #2	0/9	180 Cycles	0/9	180 Cycles
#1, #3	0/9	180 Cycles	0/9	180 Cycles
#2, #3	0/9	180 Cycles	0/9	180 Cycles
#1, #2, #3	0/9	225 Cycles	0/9	225 Cycles

2. [15 points] How many uncached loads are used in the overall best-performing configuration?

0

3. [15 points] Which replacement policy is used in the overall best-performing configuration?

Belady's Replacement