Problem M17.1: Reliability (Spring 2015 Quiz 4, Part A)

Ben Bitdiddle has a two-core processor with the following characteristics:

- Each core has a single-set, two-way set-associative private cache.
- Caches use the LRU replacement policy.
- Caches use a snoopy, bus-based MSI coherence protocol.
- Each cache line has the following fields: tag, data, and coherence state.
- The coherence state field is two bits (M state = 11, S state = 10, I state = 0X). The highorder bit represents whether the line is valid, and the low-order bit represents whether the line is dirty.
- Both cache lines share a single LRU bit. If the LRU bit is 1, block 1 will be replaced first. Otherwise, block 2 will be replaced.
- All instructions complete in a single cycle, including cache misses and bus transfers.

The structure of the private caches in the processor is shown below.

Core P1

	Cache block 1			Cache block 2		
LRU bit	Coherence state (2 bits)	Tag (3 bits)	Data (8 bits)	Coherence state (2 bits)	Tag (3 bits)	Data (8 bits)

Core P2

	Cache block 1			Cache block 2		
LRU bit	Coherence state (2 bits)	Tag (3 bits)	Data (8 bits)	Coherence state (2 bits)	Tag (3 bits)	Data (8 bits)

Suppose the private caches start with all their bits set to 0. Ben's target program for this multicore processor is the following code sequence:

```
Time :          Operation :          ACE Instruction? :
Cycle 0         P1: read 0x0A        Yes
Cycle 1         P2: read 0x0A        No
Cycle 2         P2: write 0x0A       Yes
Cycle 3         P1: read 0x0A        Yes
Cycle 4         P1: read 0x0B        Yes
Cycle 5         P1: read 0x0B        No
Cycle 6         P1: read 0x0A        Yes
Cycle 7         P1: read 0x0C        No
Cycle 8         P1: halt             No
Cycle 9         P2: halt             No
```

Problem M17.1.A

Suppose the first load in core P1 brings A to cache block 1. During which cycles are the following bits ACE? Use Y to indicate the bit is ACE, or N to indicate it is un-ACE.

Cycle	0	1	2	3	4	5	6	7	8	9
LRU bit in P1	N	N	N	N	N	N	N	N	N	N

LRU bit is not ACE for the sequence.

P1 Cache block 1

Cycle	0	1	2	3	4	5	6	7	8	9
High-order bit of coherence state	N	N	Y	Y	N	N	N	N	N	N
Low-order bit of coherence state	N	N	N	N	N	N	N	N	N	N

The high-order bit is ace during 2~3, when it is invalidated by P2. If the bit flip from 0 to 1 after P2 writes but before P1 reads, it will read wrong data and tag will match.

Since P1 only reads, the low-order bit does not matter at all

P1 Cache block 2

Cycle	0	1	2	3	4	5	6	7	8	9
High-order bit of coherence state	N	N	N	N	N	N	N	N	N	N
Low-order bit of coherence state	N	N	N	N	N	N	N	N	N	N

The high-order bit is never ACE since if 1 goes to 0, just read from memory, and if 0 goes to 1, the tag will not match.

Again, since P1 only reads, the low-order bit does not matter at all.

However, since in this question the time line is not defined very precisely, it’s okay to have a shift.

Problem M17.1.B

What is the **AVF** of the coherence state fields in the private cache of core P1 over cycles 0-9? (Consider the coherence state fields only, not other fields)

AVF = # cycle of bits is ACE / total # cycles = 2 / 40 = 5%

Problem M17.1.C

Ben wants to protect his processor from cosmic rays by adding a protection mechanism. He wants to know the AVF of tag, data, and the LRU bit in the private cache first before adding his mechanism. Help Ben classify these three fields by their AVF into the following three categories: high (AVF near 100%), low (AVF near 0%), and medium (in between). Use one or two sentences to explain your answer for each case.

Tag is medium or low because for false positive is rare, and false negative is dangerous only when data is dirty.

Data is high or medium, depending on the assumption of how is the data being reused.

LRU bit is low because it does not affect the program outcome.

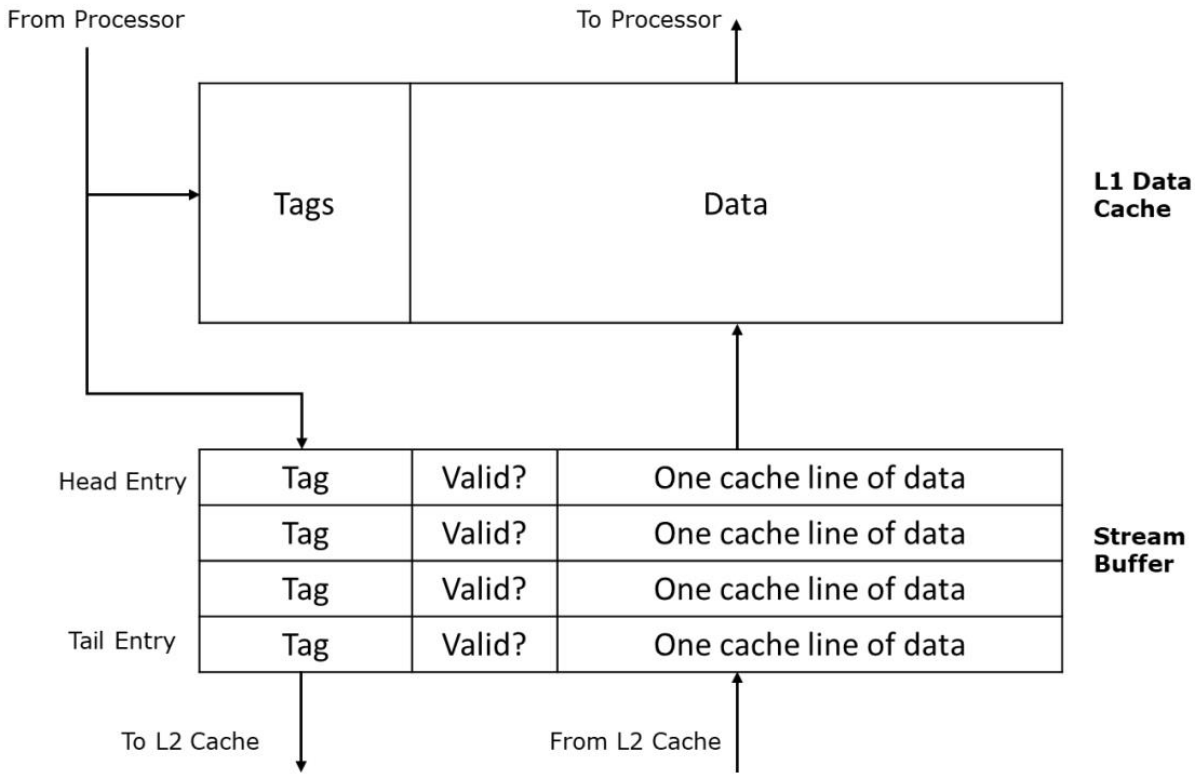
Problem M17.1.D

After finding the AVFs, Ben decides to add parity bits to all fields (coherence state, tag, data, and LRU bit). However, this causes a large number of false DUE events (detected unrecoverable errors). What structure in the private cache has the largest fraction of false DUE events, relative to their total DUE events?

It is LRU bit. LRU bit always causes false DUE, so the fraction of false DUE relative to its total DUE is 100%.

Problem M16.2: Reliability (Spring 2019 Quiz 4, Part C)

Ben Bitdiddle wants to add a stream prefetcher between the L1 data cache and the L2 cache of his processor. This stream prefetcher predicts L1 cache misses and fetches the predicted cache lines speculatively. To avoid polluting the L1 cache, the stream prefetcher buffers prefetched lines into a **stream buffer**, a 4-entry tagged FIFO queue shown below. Each stream buffer entry contains the prefetched data, the corresponding tag, and a valid bit.



When a cache miss occurs in the L1 data cache, the stream prefetcher requests the next 4 consecutive cache lines from the L2, enters their tags in the buffer, and sets the valid bits to zero. Each valid bit is set once the corresponding entry is prefetched from the L2 cache. For instance, an L1 miss to a cache line with *line address* L will cause lines L+1, L+2, L+3, and L+4 to be prefetched.

Subsequent accesses to the L1 data cache that miss compare their address against the head of the buffer to see if it contains a valid entry with a matching tag. If the access hits in the head entry of the buffer, the buffer serves the data to the L1 cache, and the prefetcher initiates a fetch for the line that follows the tail entry of the buffer. If the access misses in the head entry of the buffer, the request is forwarded to the L2, the stream buffer is flushed, and the next 4 consecutive lines are prefetched from the L2.

Problem M17.2.A

The L1 data cache consists of **16B cache lines** and is initially empty. The following sequence of events occur in the system:

Cycle	Event
`0`	Load to address `0x00` misses in L1 cache and the stream buffer
`50`	Four following cache lines arrive at the stream buffer
`100`	Load to address `0x10` misses in L1 cache, and hits in the head of the stream buffer
`110`	Load to address `0x20` misses in L1 cache, and hits in the head of the stream buffer
`120`	Load to address `0xC0` misses in L1 cache and the stream buffer
`170`	Four following cache lines arrive at the stream buffer
`200`	Load to address `0xD0` misses in L1 cache, and hits in the head of the stream buffer

Indicate whether the different **fields of the head entry of the buffer** are ACE, unACE, or unknown for each of the following cycle intervals. Explain any assumptions you make.

Cycle Interval	Tag	Valid bit	Data
`0-50`	unACE	unACE	unACE
`50-100`	unACE	unACE	ACE
`100-110`	unACE	unACE	ACE
`110-120`	ACE	unACE	unACE
`120-170`	unACE	unACE	unACE
`170-200`	unACE	unACE	ACE

The Tag is only ACE if the head is going to be accessed by a load that misses in the buffer, since a bit flip may cause a hit that serves the wrong data.
The valid bit is always unACE since it is never accessed when it is zero (the only ACEness we care about it is if valid bit flips from 0->1, serving invalid data).
The Data is unACE if it is overwritten before being read by a load, and ACE otherwise (we assume all loads are from ACE instructions).

Problem M17.2.B

For the given sequence of events in Question A, what is the Architectural Vulnerability Factor (AVF) of the data field in the head entry?

AVF = # Average number of ACE bits in a cycle / total # bits in the structure = 90/200 = 0.45

Problem M17.2.C

We now have a different program that traverses a linked list. Each node object in the linked list is stored across 2 consecutive cache lines, and different nodes are stored in non-consecutive cache lines. Qualitatively describe how the AVF of the data field would differ between the head and tail entry for this program.

The AVF would be higher in the head entry than the tail entry. Almost all of the head entries will be read, whereas almost none of the tail entries will be read.

Problem M17.2.D

Ben wants to add protection from random bit flips for the three fields in his stream buffer. For each field, indicate the most appropriate protection mechanism among the following:

- **No protection**
- **Parity bit**: Ability to detect single bit flips
- **ECC**: Ability to correct single bit flips, and detect two bit flips.

Justify your answer for each field with one or two sentences.

All three fields need some form of protection since they are sometimes ACE (even the valid bit, since we can generally have cases where a load accesses the buffer while it is prefetching lines from another miss). However, ECC is overkill since the stream buffer contains a copy of the data -- if we detect an error, the processor can simply go to the L2 cache to fetch the correct data. Thus, the correct answer here is parity bit for all fields.

If you answered No protection for Valid bit because it is unACE for the given trace, we gave full points also.