ECE506 Project 2

Coherence Simulator

Overview

- This is not a parallel programming assignment. You won't need OpenMP/MPI/ARC. You can code and run it on your own computer
- You will process number of memory requests for 4-processor system and output cache statistics
- Simulation abstracts a lot of details not under study
- Results are very dependent on the trace

Memory Trace Example

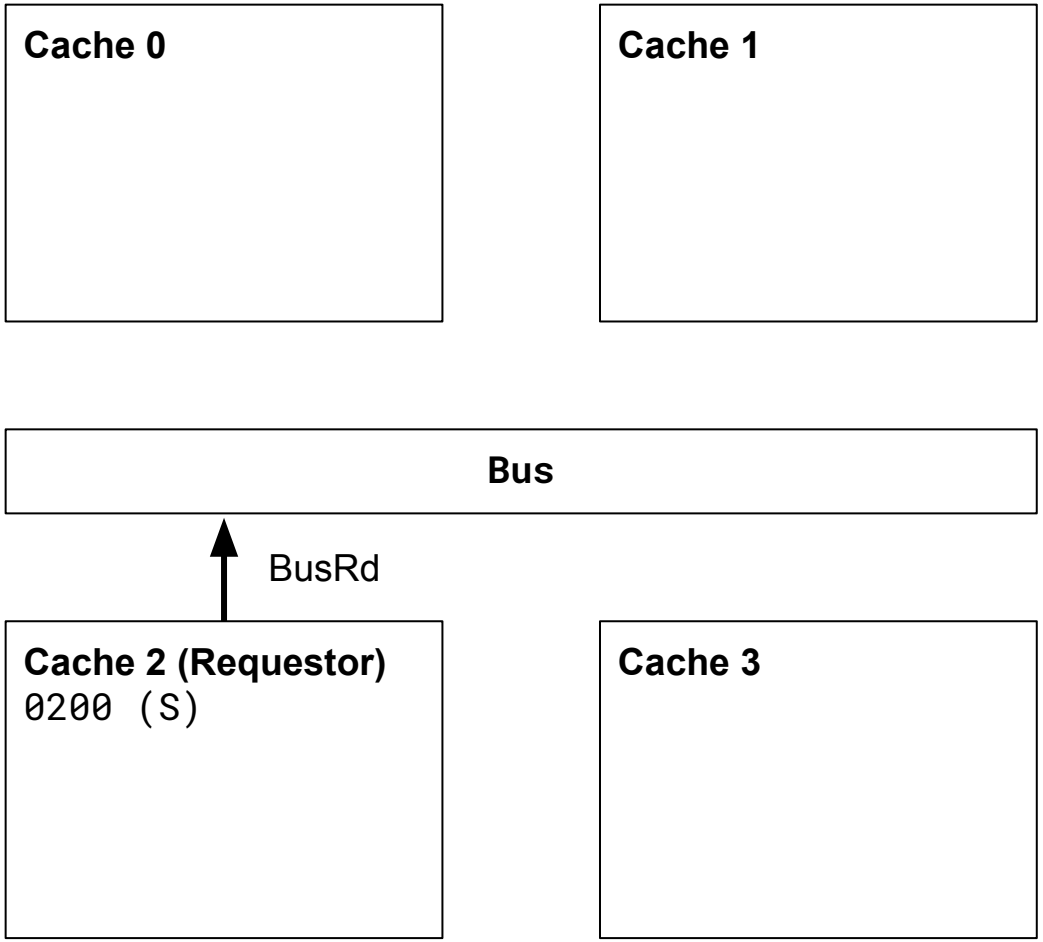
```
2 r 0200
2 w 0200
3 w 0200
1 r 0200
```

- Protocol: MSI
- We will simulate at request granularity, ignoring cycles
- For each request we will have two slides: one for requestor, one for snoopers

Request 1

```
2 r 0200
2 w 0200
3 w 0200
1 r 0200
```

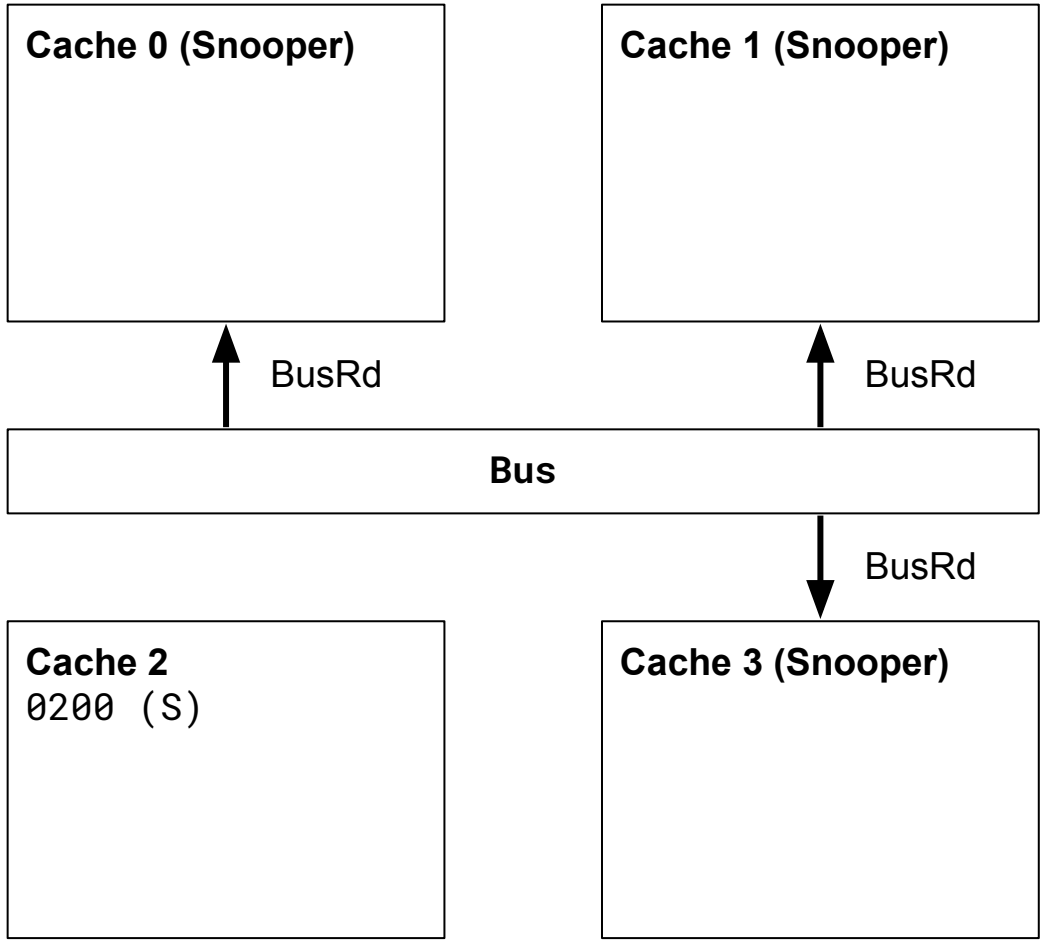
Stats	C0	C1	C2	C3
06. number of writebacks				
08. number of mem trans			1	
09. number of interventions				
10. number of invalidations				
11. number of flushes				
12. number of BusRdX				



Request 1

```
2 r 0200
2 w 0200
3 w 0200
1 r 0200
```

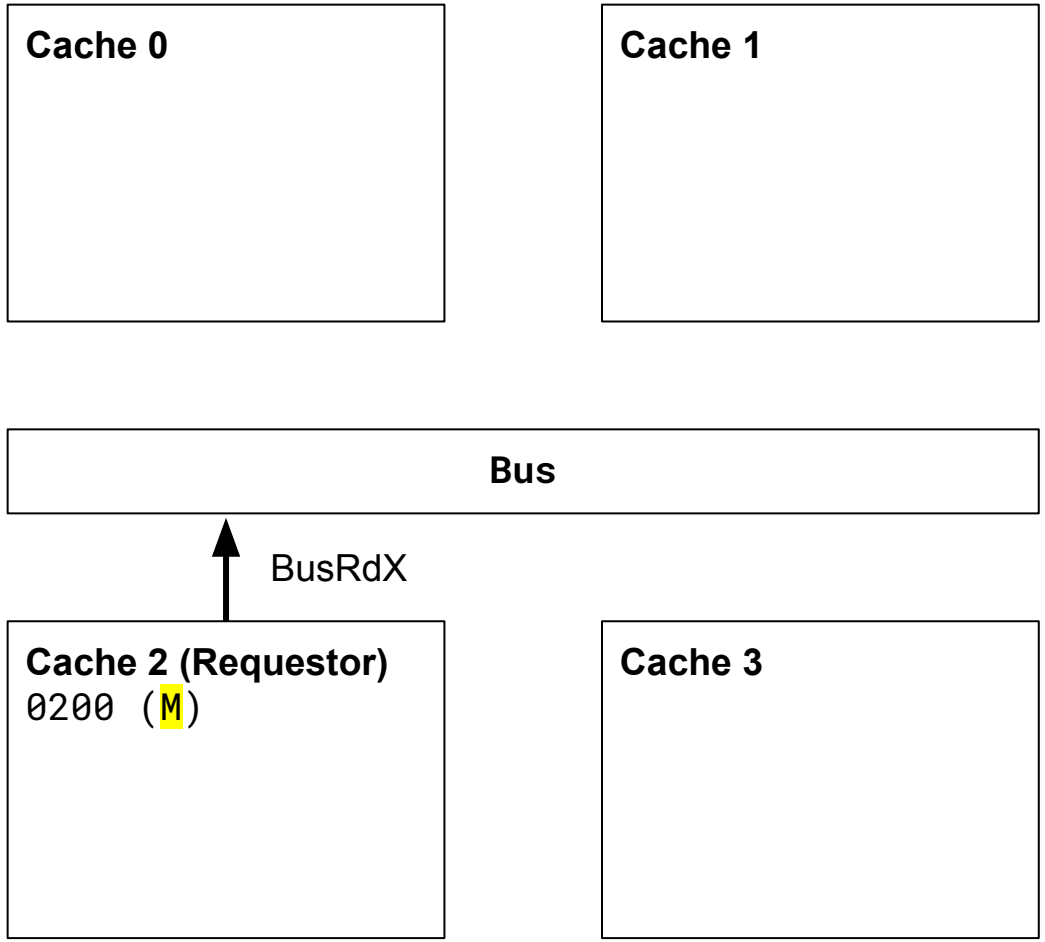
Stats	C0	C1	C2	C3
06. number of writebacks				
08. number of mem trans			1	
09. number of interventions				
10. number of invalidations				
11. number of flushes				
12. number of BusRdX				



Request 2

```
2 r 0200
2 w 0200
3 w 0200
1 r 0200
```

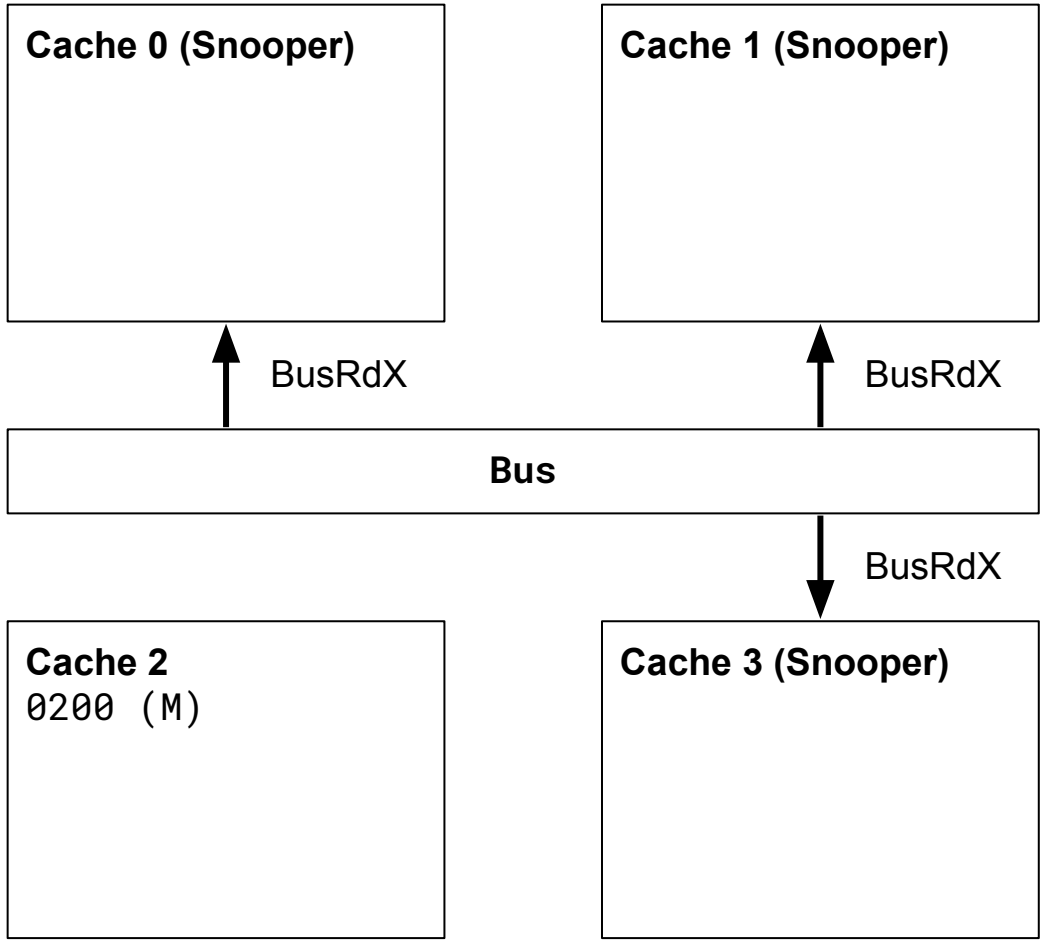
Stats	C0	C1	C2	C3
06. number of writebacks				
08. number of mem trans			2	
09. number of interventions				
10. number of invalidations				
11. number of flushes				
12. number of BusRdX			1	



Request 2

```
2 r 0200
2 w 0200
3 w 0200
1 r 0200
```

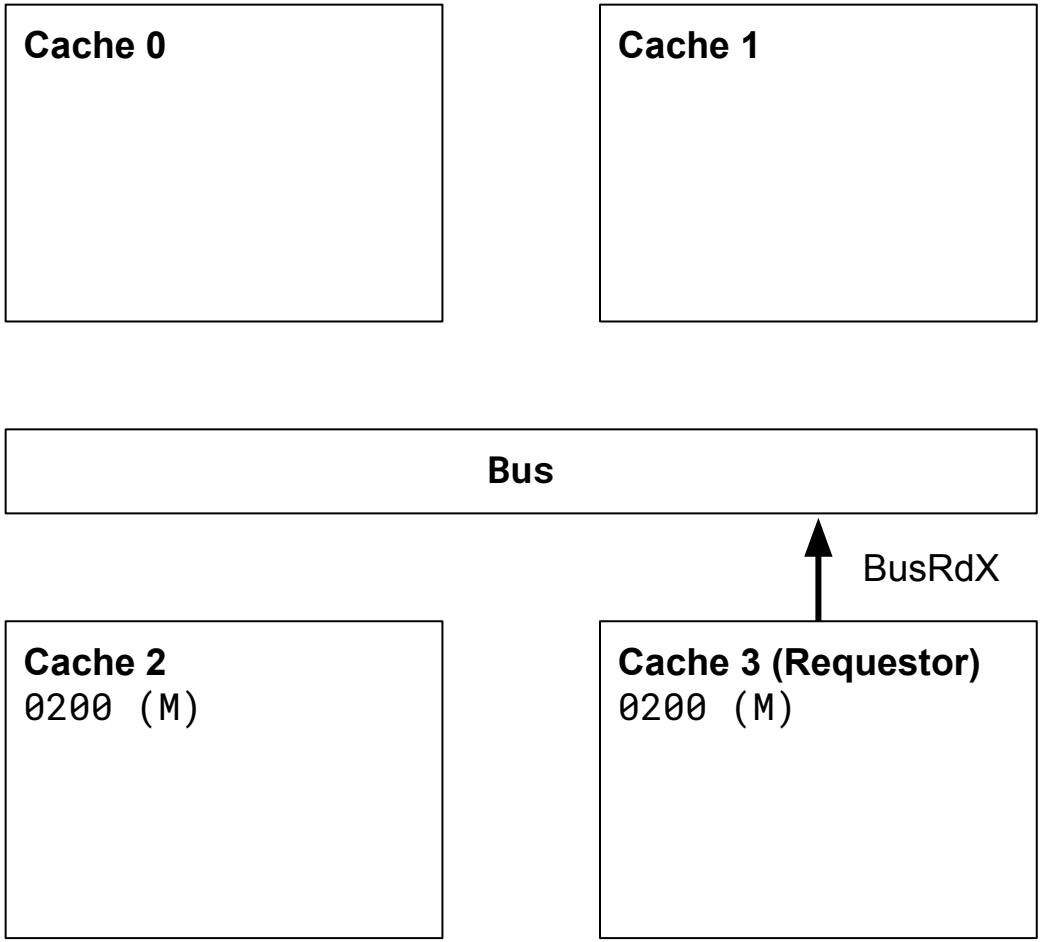
Stats	C0	C1	C2	C3
06. number of writebacks				
08. number of mem trans			2	
09. number of interventions				
10. number of invalidations				
11. number of flushes				
12. number of BusRdX			1	



Request 3

```
2 r 0200
2 w 0200
3 w 0200
1 r 0200
```

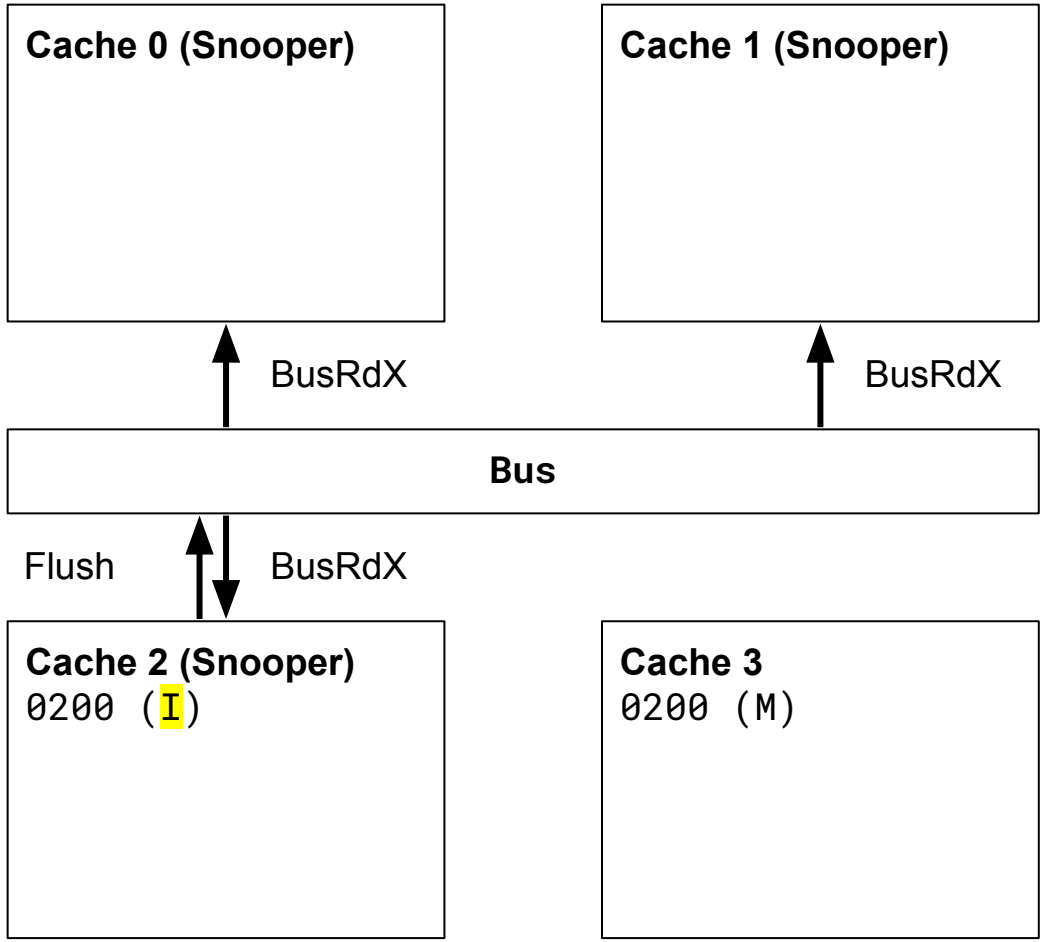
Stats	C0	C1	C2	C3
06. number of writebacks				
08. number of mem trans			2	1
09. number of interventions				
10. number of invalidations				
11. number of flushes				
12. number of BusRdX			1	1



Request 3

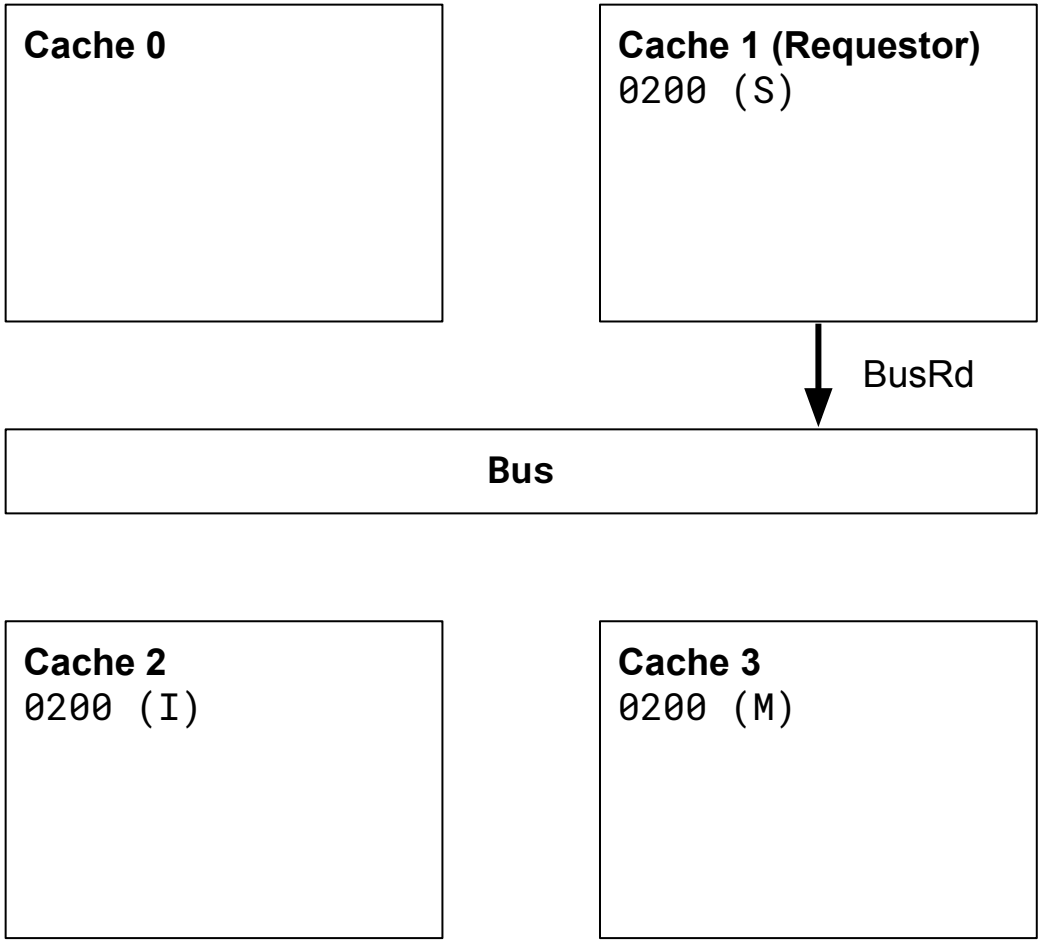
```
2 r 0200
2 w 0200
3 w 0200
1 r 0200
```

Stats	C0	C1	C2	C3
06. number of writebacks			1	
08. number of mem trans			3	1
09. number of interventions				
10. number of invalidations			1	
11. number of flushes			1	
12. number of BusRdX			1	1



Request 4

```
2 r 0200
2 w 0200
3 w 0200
1 r 0200
```

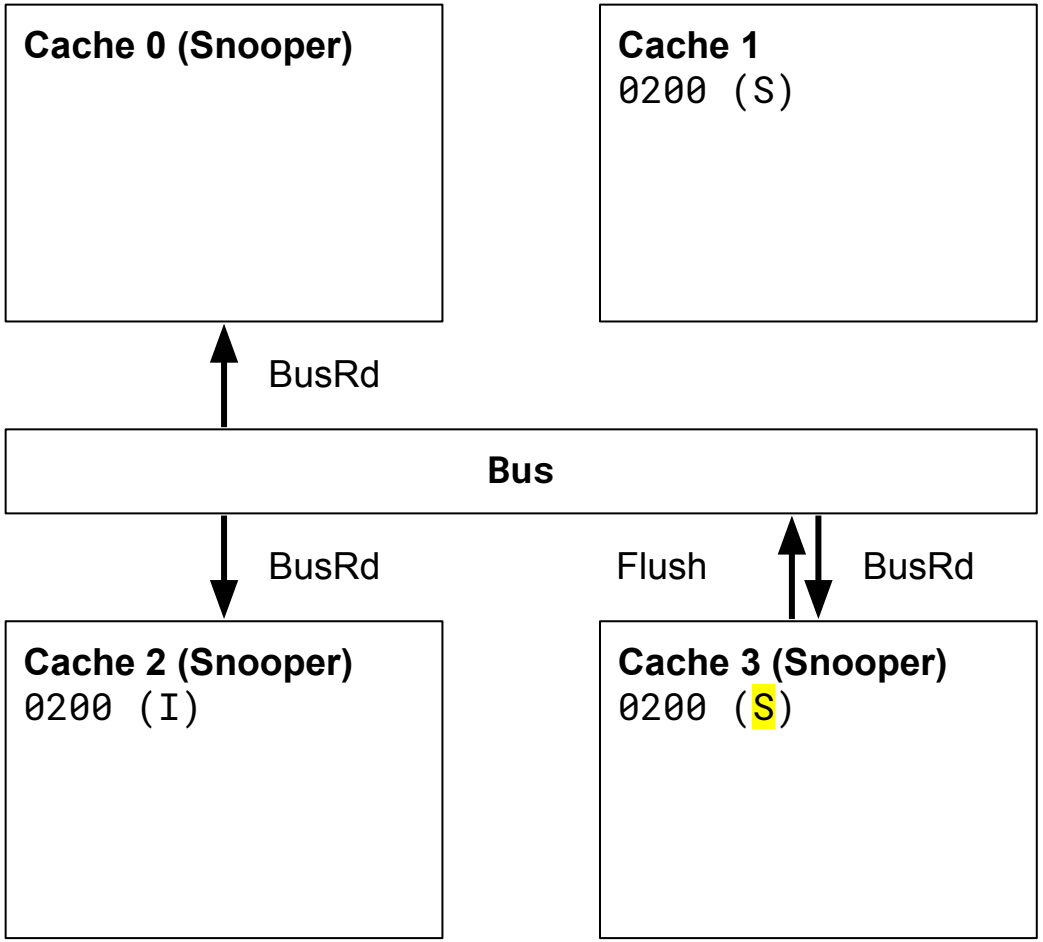


Stats	C0	C1	C2	C3
06. number of writebacks			1	
08. number of mem trans		1	3	1
09. number of interventions				
10. number of invalidations			1	
11. number of flushes			1	
12. number of BusRdX			1	1

Request 4

```
2 r 0200
2 w 0200
3 w 0200
1 r 0200
```

Stats	C0	C1	C2	C3
06. number of writebacks			1	1
08. number of mem trans		1	3	2
09. number of interventions				1
10. number of invalidations			1	
11. number of flushes			1	1
12. number of BusRdX			1	1



Main Ideas

- Instance of cache for every processor
- For every request, Access(addr) in Requestor and Snoop() in Snoopers is called
  - Requestor: updates cache line status, increments counters, issues signals
  - Snoopers: handles signal, updates cache line status, increments counters

Project Starter

Project Specification

How to Send Signal

- Return signal from Access(), pass to Snoop()'s
- Pass-by-reference variable modified in Access() and read in Snoop()'s

How to Debug

- Check state diagrams in the book
- Create short trace and calculate step-by-step statistics
- Print line-by-line statistics
- Add prints for problematic state transitions. Not every state might be visited, highly trace-dependent
- Make sure cache lines doesn’t have states from other protocols
- Make sure cache line are properly invalidated

- Check if there is updated version of starter/specs every time you resume working after many days