CS 520 Project 2
Professor Timothy Normand Miller
Spring 2017
Deadline: May 12, 2017

Description

For this project, you will implement the MOESI protocol between three caches. This will function similarly to the demo that will be shown in class, although with a simple text-based interface. Additional materials describing the MOESI protocol are posted on blackboard, and you will need to refer to those.

Please assume the following:

1. There are three caches numbered: 0,1,2
2. Four lines with address 0,1,2,3
3. All lines start out in the Invalid state

Note: To keep things simple, this project has no distinction between line and line frame. There are only four addresses in the whole system, and your protocol operates in terms of those addresses.

Input format

Your program will read input from *stdin*. Each line of input will have the following three columns:

- Cache number (0, 1, 2)
- Command (**r** for read, **w** for write, **e** for evict)
- Line number (0, 1, 2, 3)

Example Command: 1r3

This command tells cache 1 to read from line 3. Based on the availability of the line in the cache, more than one message might be sent from the requesting cache to other caches.
As will be explained in the messages section, if the line is not present in the cache, the CPU will send bus messages to read the line from other caches.

Exchanged Messages - And outputs

- Read command -- Message from local CPU to read its own cache.
    - Print the command as is. Example “1r1”
    - On a hit :
        - Hit : If line is not dirty
        - Hit Dirty : If line is dirty
    - On a miss
        - Print “Miss”
        - On a cache miss, the Cache will send a bus_read command to all other caches. The result of each bus_read invocation must be printed. Please refer to the bus_read specification for details.
    - Print the old status of the lines and the new status. Below are just examples of possible print statements:
        - I -> E
        - E -> E
    - If the bus does not exist in any of the caches, it may be fetched from main memory. Indicate by printing “MEMORY READ”

- Bus_read command -- Message from a remote CPU.
    - Print “Bus Read” followed by the cache number
        - Example (CPU 2 receives a bus read command for line 1):
            - **Cache 2, Bus Read 1**
    - Prints an appropriate combination of the following messages:
        - On a cache miss print “Miss”
        - On a cache hit, print:
            - Hit Dirty -- If the line is dirty
            - Hit -- If the line is not dirty
        - Any change in status as a result of this change. Below is an example:
            - E -> O
    - At the end of the bus message print “End Bus Read”

- Write command -- Message from local CPU to read its own cache.
    - Print the command as is. Example : “1r1”
    - On a hit: print “Hit”
    - On a miss:

        - The CPU needs to send a bus_write to every other CPU to inform them that the lines has been written to.
    - Print the old status of the lines and the new status. Below are just examples of possible print statements:
        - I -> M
    - For some hits, it is still necessary to send a bus_write.

- Bus_write command -- Message from a remote CPU.
    - Print the “Bus Write” followed by the cache number.
        - Example (CPU 2 gets a write command for line 0):
            - **Cache 2, Bus Write 0**
    - Prints an appropriate combination of the following messages:
        - On a Cache Miss Print -- “Miss”
        - On a Cache Hit Print -- “Hit” or “Hit Dirty”
        - Print “Flush” if the CPU needs to write the value to DRAM.
        - Print the old status and the new status of this line in the cache. Below are examples
            - S -> I
            - O -> I
    - At the end of the bus message print “End Bus Write”

- Evict command -- Message from local CPU to evict line
    - Print the command
    - Print old and new status of line, along with any flush to main memory that might occur. (e.g. S -> I)
    - To implement this correctly, pay special attention to “M -> I” and “O -> I” transitions.

Output format

Initially all lines are considered to be invalid in all caches.
Below is an example of an 0r0 command

The output for this command will be:

0r0
Cache 0, Miss 0

Cache 1, Bus Read 0
Miss
I -> I
End Bus Read

Cache 2, Bus Read 0
Miss
I -> I
End Bus Read

Cache 0
I -> E

Sample inputs/Outputs

You should generate your own test cases and feel free to share with other teams. Below is one example. We may add more sequences.

0r0
0r0
0w0 //Notice in this case there is no need to send bus_write, CPU knows line is in Owned State.
0r0
0w0 //No need to send write_bus, CPU knows line is in Owned State.
1w0
0r1
1r1
1w1
1r1
0r1
0w1
1r1

Messages Exchanged for the above sequence

0r0
Cache 0, Miss 0

Cache 1, Bus Read 0
Miss
I -> I
End Bus Read

Cache 2, Bus Read 0
Miss
I -> I
End Bus Read

Cache 0
I -> E
MEMORY READ

0r0
Hit
E -> E

0w0
Hit
E -> M

0r0
Hit Dirty
M -> M

0w0
Hit Dirty
M-> M

1w0
Miss

Cache 0, Bus Write 0
Hit Dirty
Flush
M -> I
End Bus Write

Cache 2, Bus Write 0
Miss
I -> I
End Bus Write

I -> M