UG-CS-4 Sem: Jan-Jun 2021

CS4401 (Computer Architecture): Assignment 2

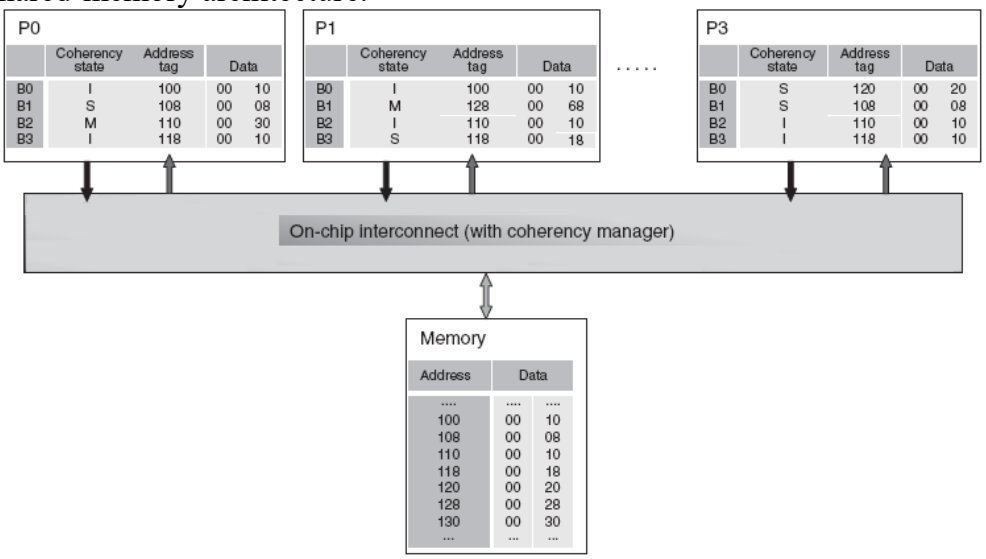
Submission deadline: **23-Apr-2021 17:30 hours** Weightage: 10%

[Answer in your own handwriting using blue/black ink]

(**MANDATORILY** write Roll no., Course code and Page no. at top margin of every page; upload single PDF)

(Common data for both questions)

Consider the following 4-core multiprocessor implementing symmetric shared-memory architecture.



Each processor has a single, private cache with coherence maintained using MSI protocol (coherence states are denoted M: Modified, S: Shared, I: Invalid). Each cache is direct-mapped, with four blocks, where each block is holding two words. Consider that the cache-address tag contains the full address, and each word shows only two hex characters, with the least significant word on the right. Assume that the initial cache and memory state are as per the given figure.

For the above system, consider the following sequence of actions:

P0: read 120
P0: write 120 ← 80
P3: write 120 ← 80
P1: read 110
P0: write 108 ← 48
P0: write 130 ← 78
P3: write 130 ← 78

Each action specifies a CPU operation of the form:

P#: <op> <address> [← <value>]

where P# designates the CPU (e.g., P0), <op> is the CPU operation (e.g., read or write), <address> denotes the memory address, and <value> indicates the new word to be assigned on a write operation. Treat each action as: independently applied to the initial state as given in above figure.

Q.No. 1. What is the resulting state (i.e. coherence state, tags, and data) of the caches and memory after each action, and what value is returned by each read operation?

[Hint: Show (in tabular form as per table given below) only the blocks that

change; for example, P0.B0: (I, 120, 00 01) indicates that CPU P0's block B0 has the final state of I, tag of 120, and data words 00 and 01. Two entries are shown for your convenience.]

P0: read 120	→	P0.B0: (S, 120, 00 20)	Read returns 0020
P0: write 120 ← 80	→	P0.B0: (M, 120, 00 80) P3.B0: (I, 120, 00 20)	

Q.No. 2. The performance of any MSI protocol implementation is determined by how quickly cache responds with data in E-state or M-state block. In some implementation, CPU read miss to E-state cache line in another processor is faster than miss to memory block. This is because smaller and faster caches respond rapidly to read misses. Conversely, in some implementation, misses satisfied by memory are faster than those satisfied by caches. This is because caches are generally optimized for "front side" or CPU references, rather than "back side" or snooping accesses.

For the multiprocessor shown above, consider the execution of a sequence of operations on a single CPU, where

- CPU read and write hits generate no stall cycles.
- CPU read and write misses generate N_{memory} and N_{cache} stall cycles, if satisfied by memory and cache respectively.
- CPU write hits that generate an invalidate incur $N_{invalidate}$ stall cycles.
- A write-back of a memory block, due to either a conflict or another processor's request to an exclusive block, incurs an additional $N_{writeback}$ stall cycles.

Consider two implementations of MSI protocol for the above system with following performance characteristics:

Parameter	Implementation 1	Implementation 2
N_{memory}	100	100
N_{cache}	40	130
$N_{invalidate}$	15	15
$N_{writeback}$	10	10

For each of the following sequence of operations, assuming the initial cache state as per above figure, consider that the second operation begins after the first completes (even though they are on different processors). For each sequence of operations, how many stall cycles are generated by each implementation?

a) P1: read 110
   P3: read 110
b) P0: read 120
   P0: read 128
   P0: read 130
c) P0: read 100
   P0: write 108 ← 48

P0: write 130 ← 78

d) P1: read 120
P1: read 128
P1: read 130

e) P1: read 100
P1: write 108 ← 48
P1: write 130 ← 78

[Hint: Solution of (a):
P1: read 110 read miss satisfied by P0's cache
P3: read 110 read miss is satisfied by memory
Implementation 1: $\{N_{cache(P1)} + N_{writeback(P0)}\} + \{N_{memory}\}$ = 150 stall cycles.
Implementation 2: $\{N_{cache(P1)} + N_{writeback(P0)}\} + \{N_{memory}\}$ = 240 stall cycles.]