HW4

1. Sol
2. Considering the impossible results. We can find the restriction of assignment for process order happens to the sequence P1-P2-P3 only when u=1 and v=1. Under this condition, w must be 1. Thus, u=1, v=1, w=0 is the only impossible result.
3. Similarly, the restriction happens when u=1, v=0, and w=1, where x must be 1. So the impossible result is u=1, v=0, w=1, x=0.
4. For the first case, where two instruction in each process is undividable, either one of them should happens before the other. Thus, the result should be either u=1, v=0 or u=0, v=1.

For the second case, just treat them as separate instruction. Therefore this is an additional possible result as u=0, v=0.

1. Sol
2. Considering cache request, for every 3 cycles, there will be 1+30\*0.01 = 1.3 more cycles for tag occupation, holds 1+0.01\*2/4.3 = 24% of the total time;

Considering snooping, for every 5 cycles, there will be 1+0.1\*2=1.2 more cycles for tag occupation, holds 1.2/5 = 24% of the time.

Thus, 24%\*24% = 5.76% of the total time would be lost for contention.

1. If separate tags is used, contention happens only for cache modification, which is (0.02/4.3)\*(0.2/5) = 0.0186% of the time.
2. For a processor reference, contention only need it to suspend. But for a bus snooping, every contention case from other processors will need the sender to wait. Thus, I would give priority to bus snooping.
3. Sol
4. It’s coherent, store atomic and TSO. Because load is logically guaranteed to get the right value though load commit can be reordered before write, and every store is ordered and committed transitively
5. It’s coherent and TSO. What’s different from (a) is that store execution can be replaced.
6. It’s even incoherent. In fact, load may get wrong old answer before write, which means it’s even incoherent.
7. It’s coherent, store atomic. What’s different from (a) is that W-R reorder is not allowed. But unrelated W-R reorder is still allowed.
8. It’s coherent, store atomic and sequentially consistent. What’s different from (a) is that it strictly defines the execution order of load. There is no trick played and it’s therefore sequentially consistent.
9. A. It’s incoherent. Because write is allowed to execute in any order without any guarantee of correctness.

B. It’s coherent, weakly ordered and TSO. What’s different from (b) is that unrelated W-W reorder is allowed.

C. It’s totally incoherent. Without any restriction, there is nothing about correctness is guaranteed.

D. It’s incoherent. Because write is allowed to execute in any order without any guarantee of correctness.

E. It’s incoherent. Because write is allowed to execute in any order without any guarantee of correctness.