Parallel Computing
Homework Assignment 2 Solutions
(Total of 30 points)

1. [2 points, 1 for no and 1 for justification]
No, they cannot.
This is because each process has its own virtual address space and the OS will ensure, though page table entries, that not two processes write to the same physical address.

2. [2 points, 1 for no and 1 for justification]
Yes, they can.
This is because they share the same virtual address space. So, if they write to the same virtual address it is writing to the same physical address.

3. [3 points (1pt for just using Amdahl's law 2 for the conclusion)]
In Amdahl's law Speedup(p) = 1 / (F + (1-F)/P) where F is the sequential fraction.
In the problem at hand $4 = 1/(0.5 + 0.5/P)$ → Cannot be done.
So, even with infinite number of cores we will get speedup of 1/0.5 = 2

4. [2 points, 1 for yes and 1 for justification]
Yes, if these two variables are stored in the same cache block. This is called false sharing.
[Remember: Caches do not understand variables. They only see cache blocks.]

5. [3 points]
If one of the tasks takes as much as the total of the other three, then the other three cores, in case of four cores will finish much earlier and stay idle. In which case, two cores are enough to reach the same speedup.
For example: Task A needs 12 units of time while tasks B, C, and D need 4 units of time each.

6. [2 points: 1 for each]
- If one core is executing a thread that has more computations than threads on other cores, this core can become hot and hence slows its speed down to avoid burning.
- Whenever there is a synchronization point, you are slowing down fast threads due to slower threads.

7. [4 points, 1 for 'No' and 1 for each justification]
No, we still cannot guarantee perfect load balancing.
For the following reasons:
- One thread may have worse memory access pattern and hence will be slower.
- One thread may be doing more communication than the other.
- One thread can have more cache misses or page faults than the other.

8

a. [4] Although you may think we need 4 cores, but on a deeper thinking, you will find that 2 cores are enough to get as much speedup as 4 cores but better efficiency.
Core 1 will work on S1, P1, P2, and P3; while core 2 will work on P4. This will take 2050ms. Then core 1 will work on S2, P5, P7, and P8; while core 2 works on P6. The total time for that part is 550ms.
Finally, core 1 will do S2. This makes the total execution time = 2050 + 550 + 50 = 2650ms

b. [4] Speedup(p) = T1/Tp where P is the number of cores.
T1 = 3250 (the sum of all tasks)
T2 = T4 = T8 = 2650 (as indicated in the previous question)
So
S(2) = S(4) = S(8) = 3250/2650 = 1.23

c. [4: 2 for each]
Span is the longest path of dependence in the graph
Span = T_∞ = 2650

Work is the total amount of execution spent on all instructions
Work = T1 = 3250