Parallel Computing
Homework Assignment 2 Solutions
(Total of 40 points)

1. [3 points, 1 for no and 2 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.

2. [3 points, 1 for no and 2 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.

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 $4 = 1/(0.4 + 0.6/P)$ → Cannot be done. So, even with infinite number of cores we will get 1/0.4 = 2.5

4. [3 points, 1 for yes and 2 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 is taking much longer time than the others, for example it is taking as much as the total of the other three, then the other three cores will finish much earlier and stay idle. In which case, two cores may be enough.

6. [3 points: 1 for answer and 2 for justification]
The more the synchronization point the more sensitive (i.e. will be easily affected) your program will be to load balance.
This is because each time you reach a synchronization point, you are slowing done some processes/threads.

7. [3 points, 1 for 'No' and 2 for justification]
No, higher efficiency does not imply higher speedup. And speedup is what we are looking for.
A single core will always have an efficiency of 100% yet its speedup is 1. With more cores, we can get higher speedup while the efficiency may decrease.

8. [7 points, 1 for no and 2 for each item]
No, we cannot make this assumption. Because:

- One thread can get a cache miss when accessing a data and the other thread does not.

- One thread can get page fault when accessing a data and the other thread does not.
- Memory response time is not uniform (Non-Uniform Memory Access or NUMA as we saw in class) and same for the shared cache (non-uniform cache access or NUCA).

9.

a. [3] Although you may think we need 4 processes, but on a deeper thinking, you will find that 2 processes are enough to get as much speedup as 4 processes but better efficiency. Process 1 will work on S1, P1, P2, and P3; while process 2 will work on P4. This will take 2050ms. Then process 1 will work on S2, P5, P7, and P8; while process 2 works on P6. The total time for that part is 550ms.
Finally, process 1 will do S2. This makes the total execution time = 2050 + 550 + 50 = 2650ms

b. [3] Speedup(p) = T1/Tp where P is the number of processes.
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. [3]
Span is the longest path of dependence in the graph
Span = $T\infty$ = 2650

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

d. [3] Parallelism T1/ $T\infty$ = 3250/2650 = 1.23
This is why increasing the number of processes beyond 2 does not yield any advantages.