CUDA Code Profiling Exercise

Questions

1. Basic Matrix Multiplication kernel runtime: 0.0081 ms
2. HtoD memcpy = 0.019 ms, DtoH memcpy = 0.006 ms,

   Duration of memory transfer = 0.019 + 0.006 = 0.025 ms
3. Achieved occupancy: 16.46%

   Occupancy is the number of active warps per SM divided by the maximum number of warps per SM 48 warps.
4. The main performance bottleneck comes from the low achieved occupancy.

   There are 48 warps per SM, but on average only 7.9 warps are active per SM i.e. $7.9 \times 32 \approx 252$ threads are active per SM (out of the max 1536 threads per multiprocessor). So several threads are idle in the SM while waiting for the high latency global memory accesses.
5. Tiling uses shared memory which would reduce the latency of memory accesses and improve the throughput of the multiplication and addition operations. This doesn't match the bottleneck of the basic matrix multiplication but the bottle makes the latency of memory accesses very visible.
6. Tiling kernel runtime: 0.007 74 ms Runtime improvement: 0.0081 ms − 0.007 74 ms = 0.36 µs or 4.35%
7. HtoD memcpy = 0.013 ms, DtoH memcpy = 0.004 ms

   Duration of memory transfer = 0.013 + 0.004 = 0.017 ms

   Difference in memory transfer time: 0.025 ms − 0.017 ms = 0.008 ms or ∼ 30%
8. Achieved occupancy: 16.52%

   Difference in achieved occupancy: 16.52% − 16.46% = 0.06% or ∼ 0.36% Difference
9. There was no significant overall improvement in the performance because the data set was too small to see the benefits of tiling. Furthermore the difference in the memory transfer is too small (in the order of µs) for there to be a significant difference.
10. To improve the performance of the matrix multiplication, we can implement memory coalescing by loading the B matrix (from $A \times B$) from global memory to shared memory by inputing into the shared memory via column-major order. Accessing the `B_s` matrix in consecutive DRAM locations allows the data to be delivered in bursts which will be faster than the non-coalesced memory access.