High Performance Computing with GPUs
Exercise Sheet 2

Shoaib Ahmed Siddiqui

Part a: Information about the used device

Number of streaming multi-processors: **28**
Number of ALUs per streaming multi-processor: **128**
Peak bandwidth = (Memory clock * Bus width / (Bits per byte)) * 2
Memory clock = 5.5 GHz
Bus width = 352 bit
Bits per byte = 8
Peak bandwidth = (5.5 GHz * 352 Bit / 8) * 2
Peak bandwidth = 484 **GB/s**

The calculated peak memory bandwidth perfectly matches the memory bandwidth mentioned on the manufacturer's website (Fig. 1).

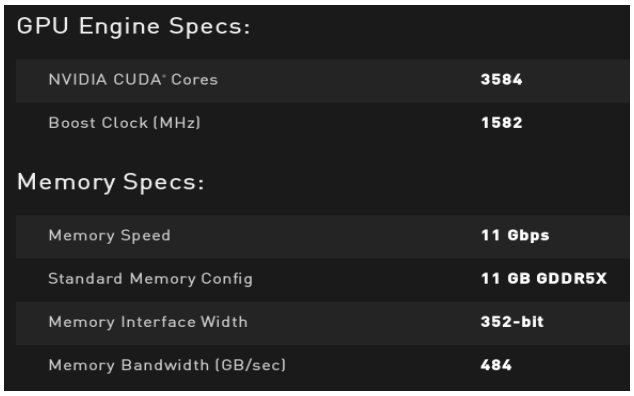


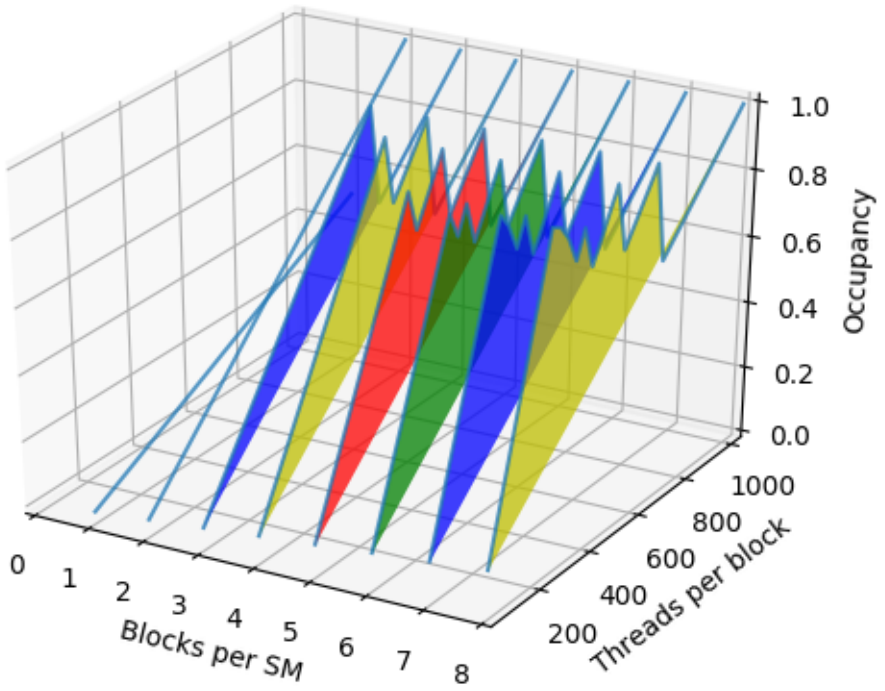
Fig. 1. Official configuration of the card mentioned on the website

Peak performance = Clock Rate$_{(SM)}$ * 2_Ops * Number of SM * Cores per SM

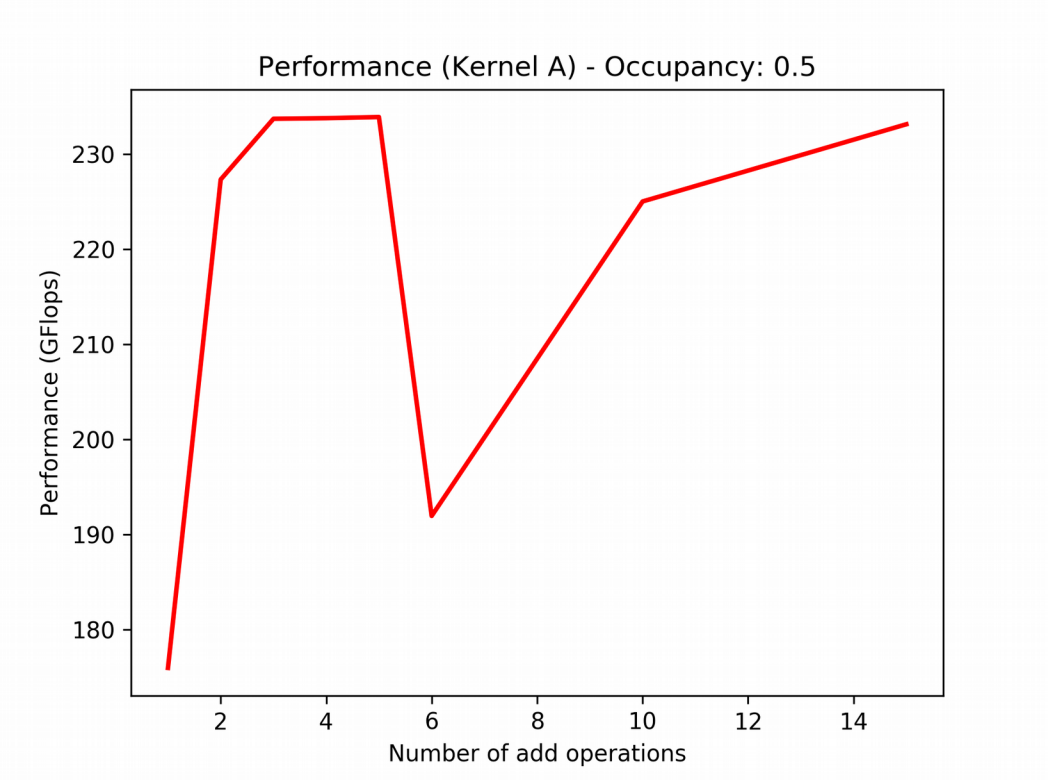
Clock Rate$_{(SM)}$ = 1.62 GHz
Number of SM = 28
Cores per SM = 128
Peak performance = 1.62 GHz * 2 * 28 * 128
Peak performance = 11612.16 GFlops

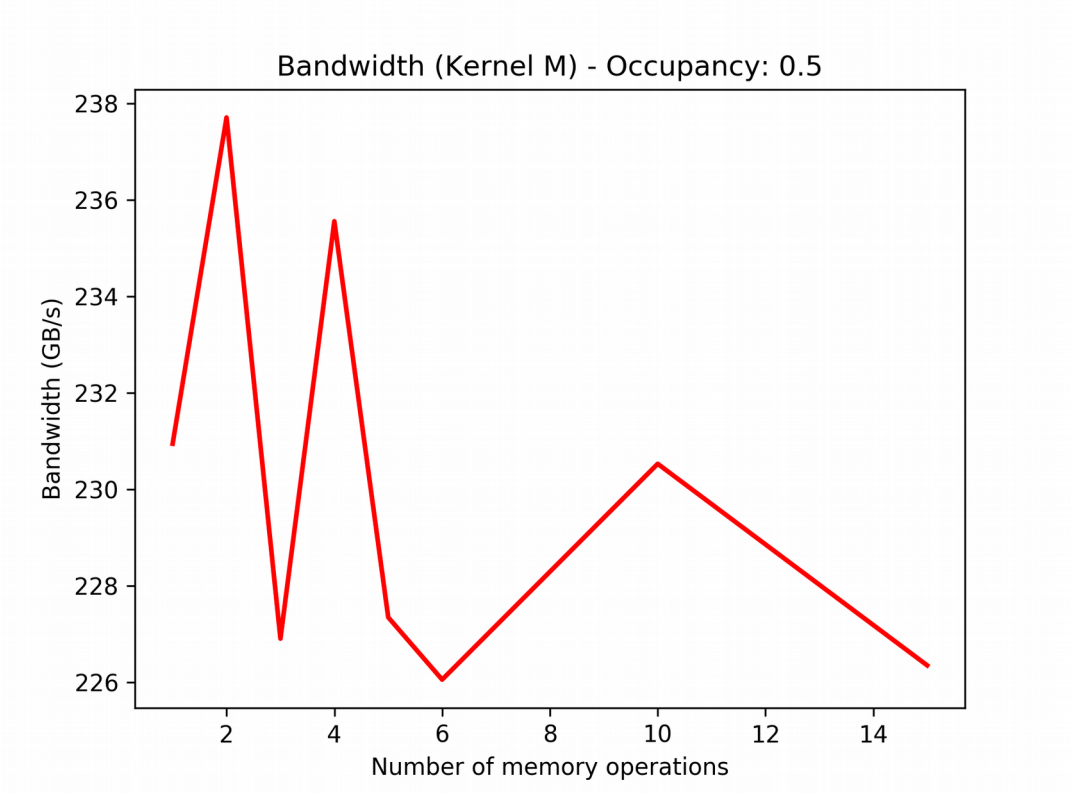
Part b: Performance and Occupancy

1. Fixed number of operations. Change the occupancy of the card.

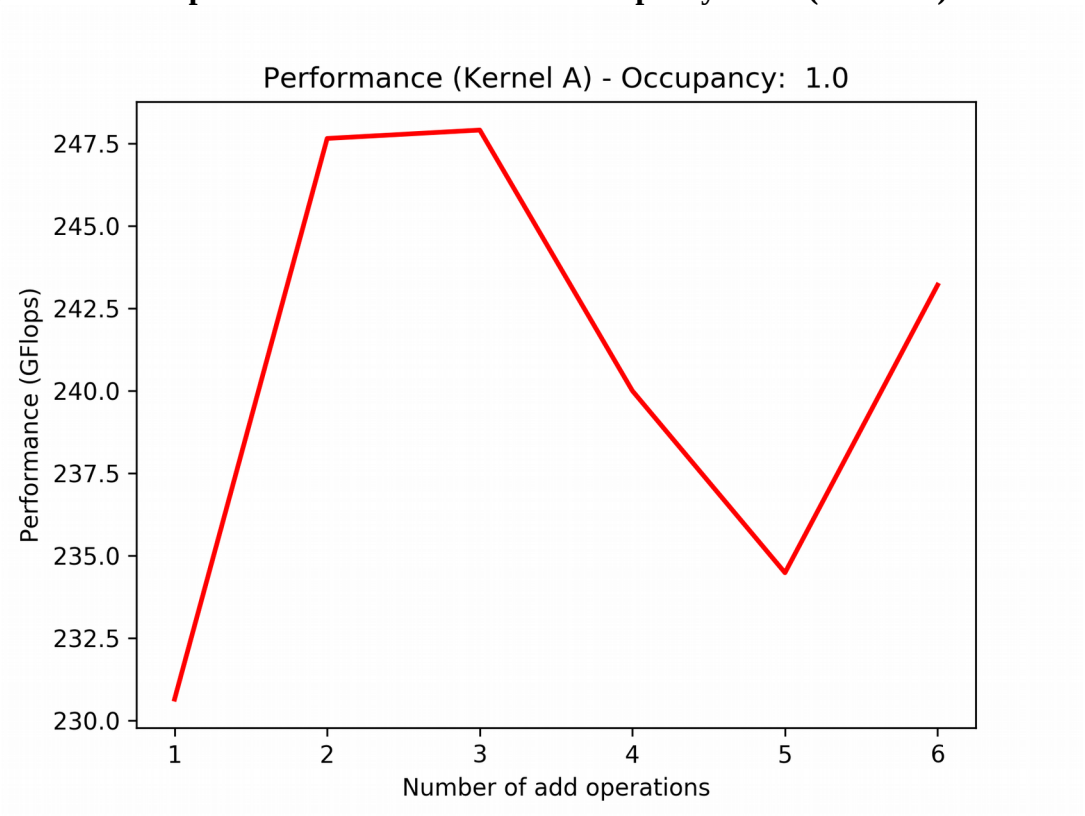


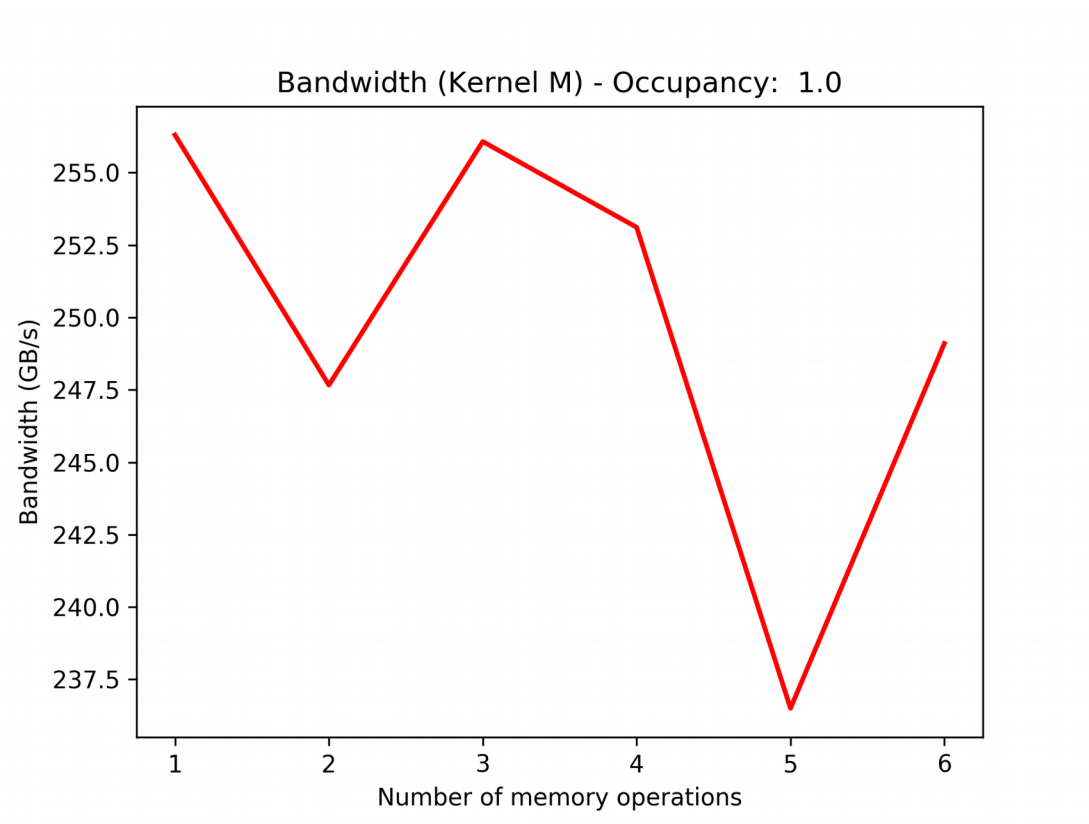
2. Change number of operations in the kernel with occupancy of 0.5 (A and M)



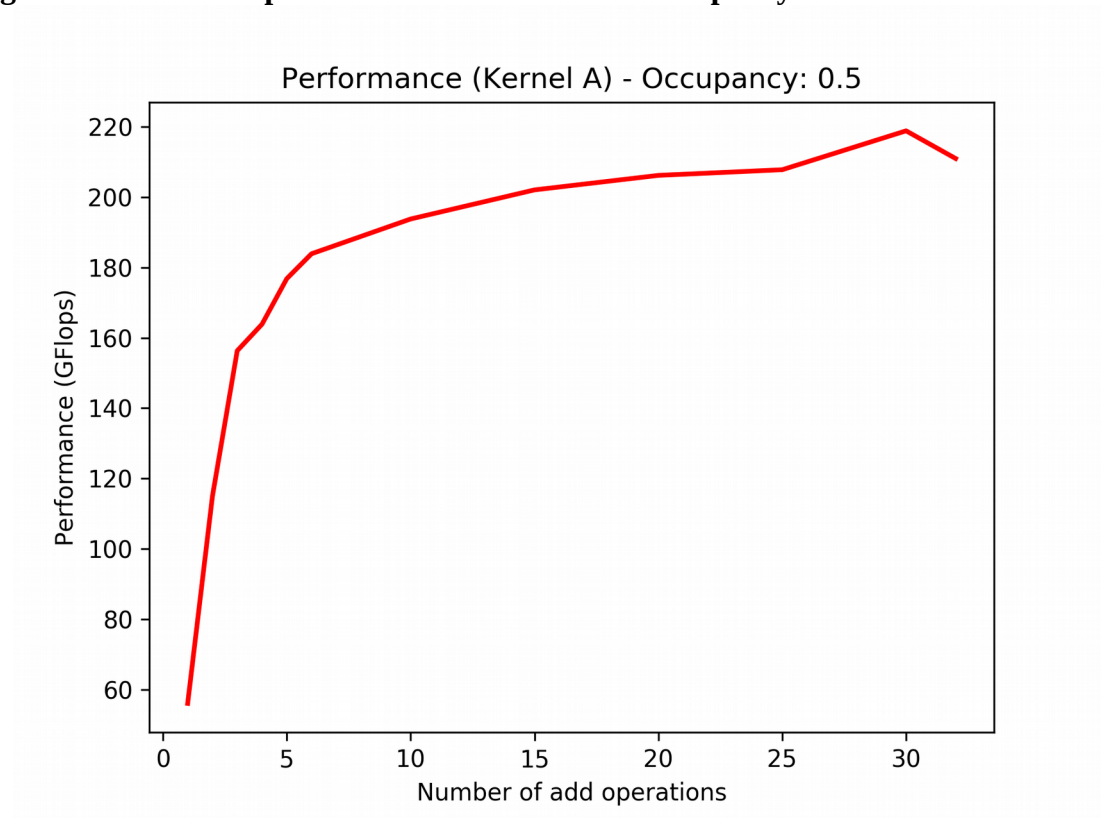


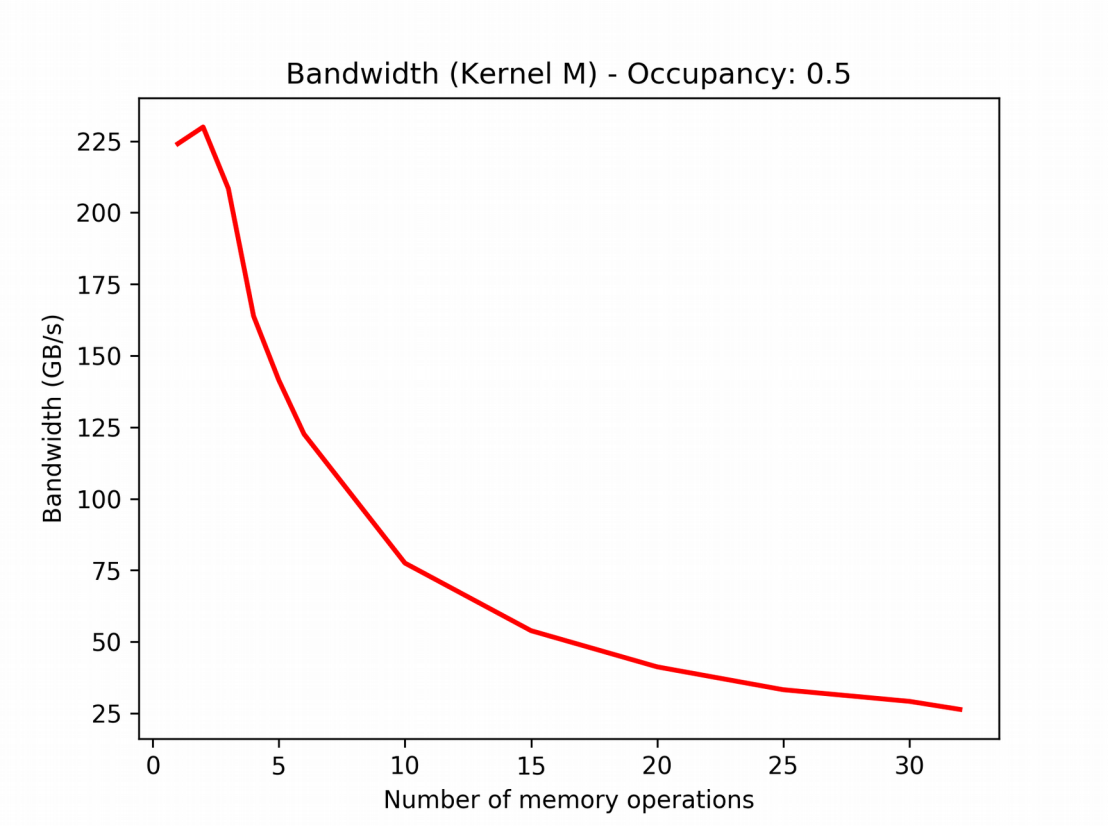
3. Change number of operations in the kernel with occupancy of 1.0 (A and M)





4. Change number of add operations in the kernel with occupancy of 0.5





The bandwidth and the performance obtained was always significantly lower than the peak bandwidth and performance. I tried changing the number of operation via adding a loop as well as directly adding instructions as the loop itself adds some overhead but the outcome was almost the same which might be due to the optimization performed by the compiler.
The copy-pasted identical instructions resulted in a dependency in the computation of the values which can might have resulted in drop of performance.