C1

./dp1 1000000 1000

N: 1000000 T: 0.000000000 sec B: 35772925764.192 GB/sec F: 8943231441.048 GFLOP/sec

./dp1 300000000 20

N: 300000000 T: 0.000000001 sec B: 3120761904.762 GB/sec F: 780190476.190 GFLOP/sec

C2:

./dp2 1000000 1000

N: 1000000 T: 0.000000000 sec B: 62534351145.038 GB/sec F: 15633587786.260 GFLOP/sec

./dp2 300000000 20

N: 300000000 T: 0.000000000 sec B: 4990050761.421 GB/sec F: 1247512690.355 GFLOP/sec

C3:

(base) [ck3411@log-1 High-Performance-Machine-Learning]$ ./dp3 1000000 1000

N: 1000000 , R: 1000

N: 1000000 T: 0.000 sec B: 252.893 GB/sec F: 63.223 GFLOP/sec

(base) [ck3411@log-1 High-Performance-Machine-Learning]$ ./dp3 300000000 20

N: 300000000 , R: 20

N: 300000000 T: 0.017 sec B: 137.577 GB/sec F: 34.394 GFLOP/sec

C4:

N: 1000000 T: 0.290642359 sec B: 1431.31235 GB/sec F: 3.44065 GFLOPS/sec

N: 300000000 T: 0.4978157681 sec B: 819.580306902 GB/sec F: 2.00877526201 GFLOPS/sec

C5:

python3 dp5.py

N: 1000000 T: 5.777567625045777e-05 sec B: 7200261.892160959 GB/sec F: 17308.32185615615 GFLOPS/sec

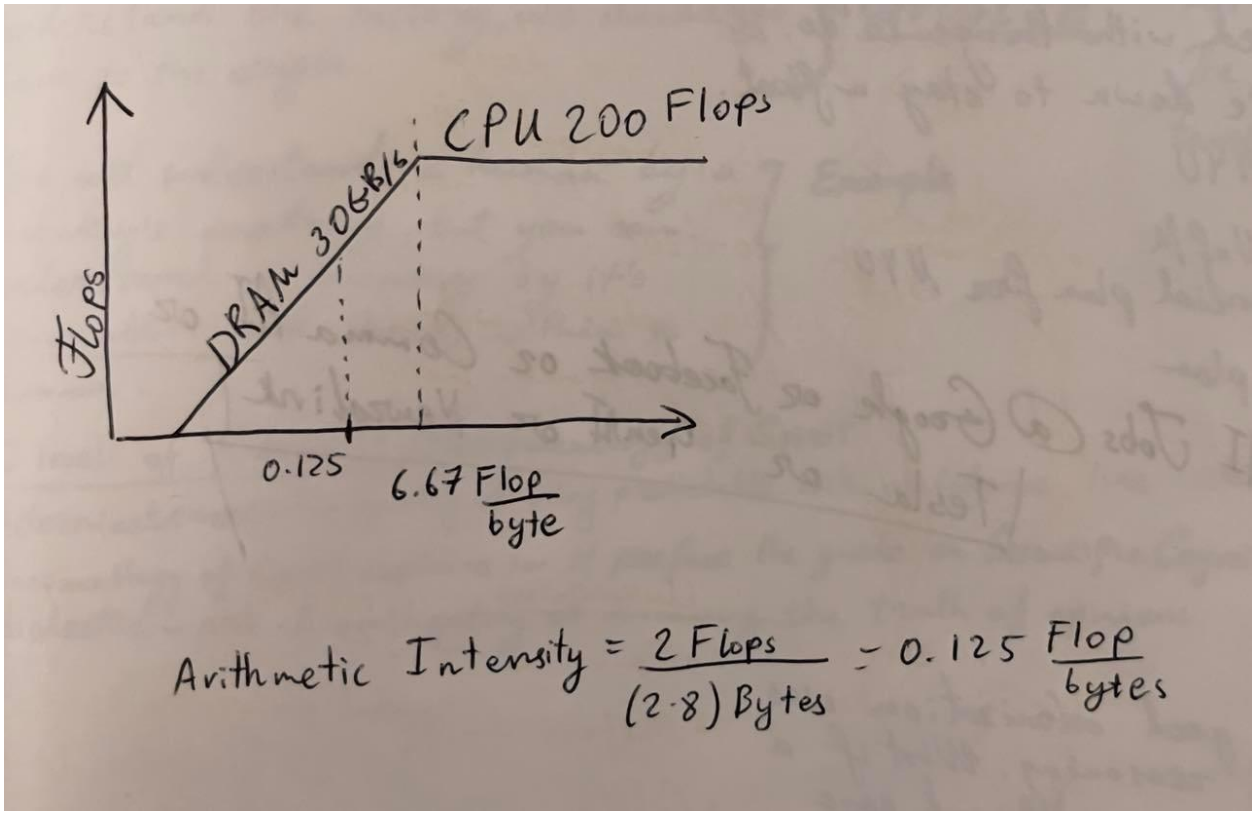
R: 1000

N: 300000000 T: 0.020117679913528263 sec B: 124069.97281637574 GB/sec F: 298.2451269624417 GFLOPS/sec

R: 20

Q1: Explain the consequence of only using the second half of the measurements for the computation of the mean.
A1: Depending on the CPU hardware the consequences of only using the second half of the measurements for computation time will vary. For example, the hardware might be caching some of the floating point numbers and not the others so measuring time on the second half could result in computation time when all the values are in the registers. This will make the computation more standardized as compared to the other computations where the numbers were not cached and causing slower computation.

Q2 (6 points): Draw a roofline model based on 200 GFLOPS and 30 GB/s. Add a vertical line for the arithmetic intensity. Draw points for the 10 measurements for the average results for the microbenchmarks.
A2:



Q3 (5 points): Using the N=300000000 simple loop as the baseline, explain the difference in performance for the other 5 measurements in the C variants.
A3: C1 and C2 achieved the highest performances since the computation was very efficient. C is a low level language that does not have as many layers of abstractions, so if a code is written efficiently in C it will perform better than C4 where the program was written in python script. We also saw that C3 was slower compared to C1 and C2, this could be because the intel library is

not written as efficiently as our code. Since Python is a high level language written using C and C++, it's performance will not be as efficient as the lower level languages. This is a tradeoff between ease of use and efficiency. On the other hand we saw that using a library based dotproduct function in python gave better performance, this is because the library itself is written in a low level language.

Q4 (6 points): Check the result of the dot product computations against the analytically calculated result. Explain your findings. (Hint: Floating point operations are not exact.)

A4: The computation results were not exact because it is often impossible to represent irrational numbers like pi and sqrt(3) precisely! Type float only has around 7 decimal places but irrational numbers have infinitely many that need to be exactly accounted for in correct calculations. Numbers like pi that are infinite series are rounded when put in floating point numbers format, the errors caused by rounding add up causing the inexact results.