**Vector Addition**

**How many floating operations are being performed in your vector add kernel? EXPLAIN.**

*out = in1 + in2 is the only only floating point operation being done in the VecAdd kernel*

**How many global memory reads are being performed by your kernel? EXPLAIN.**

*Global memory reads are associated with copying from Host to Device (cudaMemcpyHostToDevice). As the only data we are copying from Device to host is Input1 and Input2, there are 2 global memory reads.*

**How many global memory writes are being performed by your kernel? EXPLAIN.**

*Global Memory writes are associated with copying from Device to Host (cudaMemcpyDeviceToHost). As there is only one copy from Device to Host (ImageOutput) there is only 1 Global Memory write.*

**Color Space Conversion**

**How many floating operations are being performed in your color conversion kernel? EXPLAIN.**

*Only one: outImg[grayidx] = (0.21\*r + 0.71\*g + 0.07\*b); is the only one using a floating point operation. All other operations are memory reads.*

**Which format would be more efficient for color conversion: a 2D matrix where each entry is an RGB value or a 3D matrix where each slice in the Z axis represents a color.**

*A 2d matrix would be more efficient because 3 (RGB) doesn’t make a nice number when you divide it. ie. the dim for DimBlock is 16,16 and that makes 256. However, 9\*9\*3 = 243 which is about the closes you can get, which means there would be threads are not being utilized.*

**How many global memory reads are being performed by your kernel? EXPLAIN.**

*There is only one memory read (cudaMemcpyHostToDevice): deviceInputImageData*

**How many global memory writes are being performed by your kernel? EXPLAIN.**

*There is only one image that is being written from Global memory (cudaMemcpyDeviceToHost): &deviceOutputImageData*