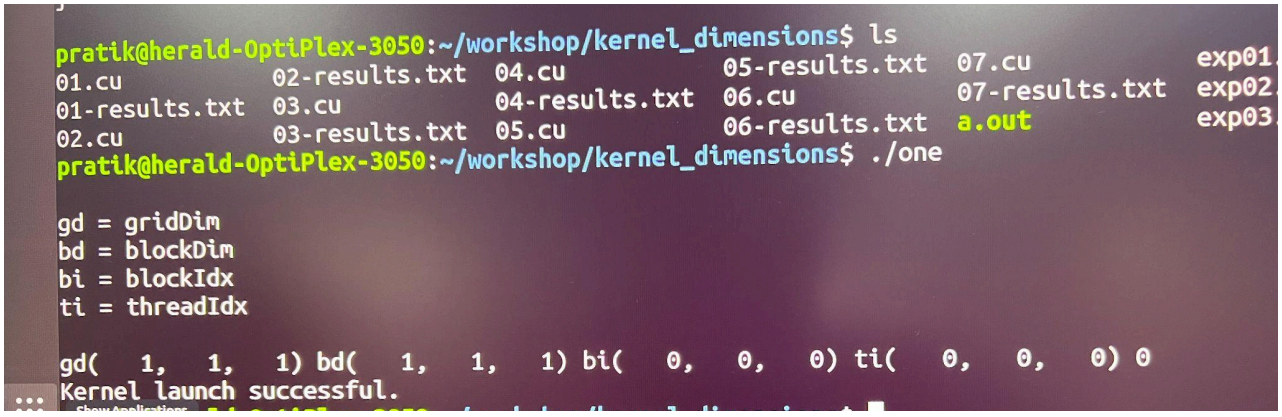
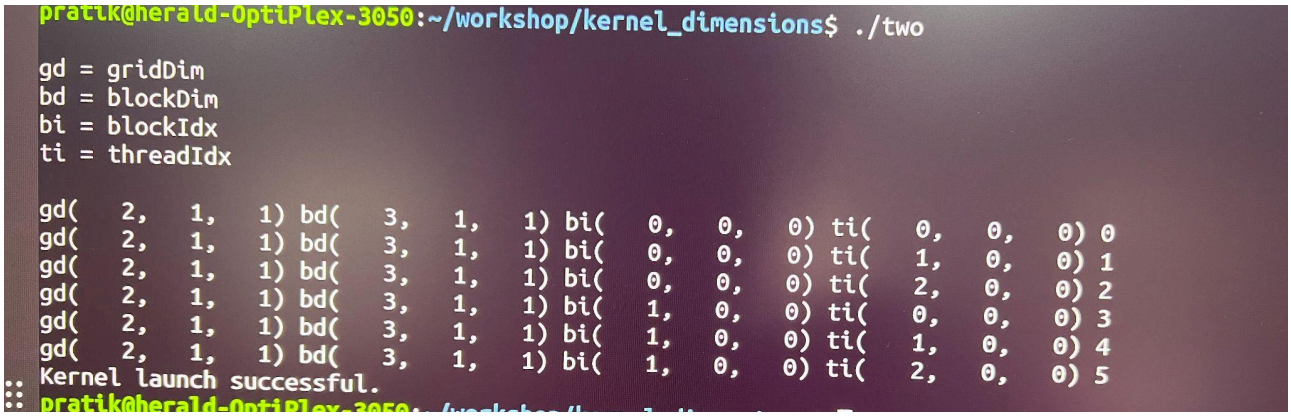
# Pratik Dhimal
# 2407779

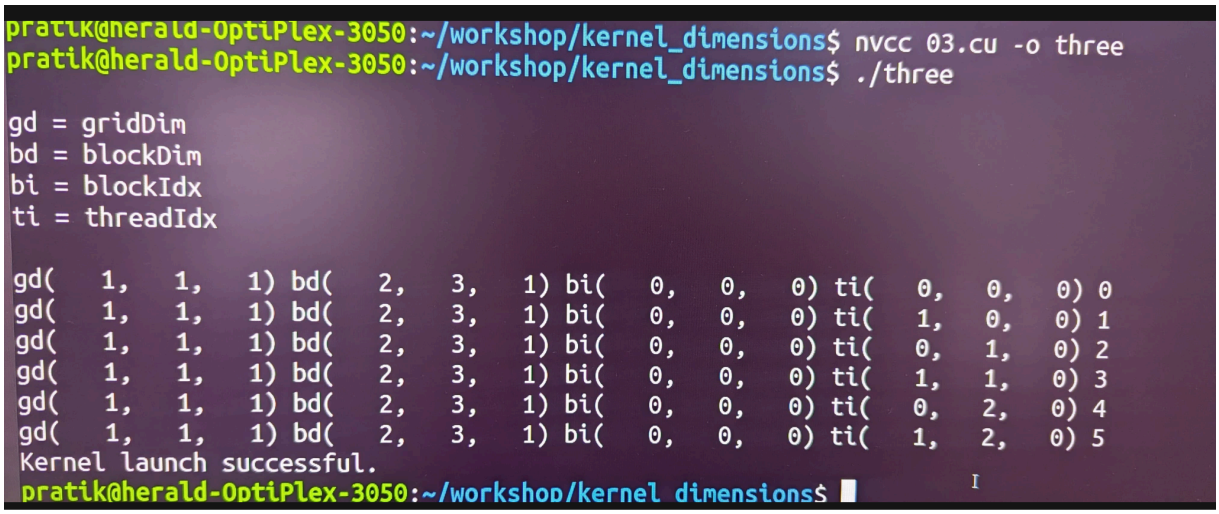
1. In this programmer has launched a kernel with kernel <<<1,1>>> which means 1 block and 1 thread. Also the dimension will (1,1,1) for both the grid and block. Hence, **the number of threads will be 1**. And the output will be the configuration as shown below.



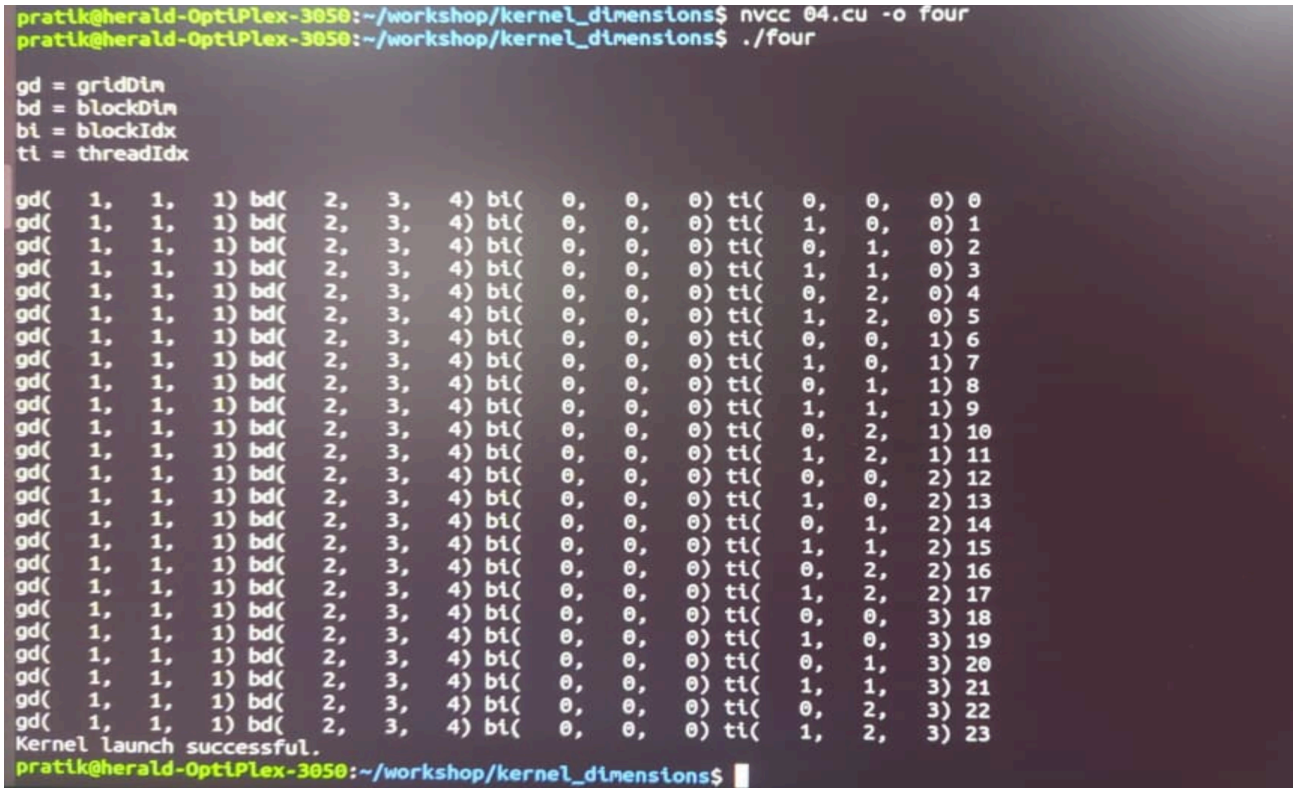

2. Similar to 1, but here we have used kernel <<<2,3>>>. Hence the block and thread all be in x direction. **So there will be 6 threads, 3 threads for each 2 blocks**.



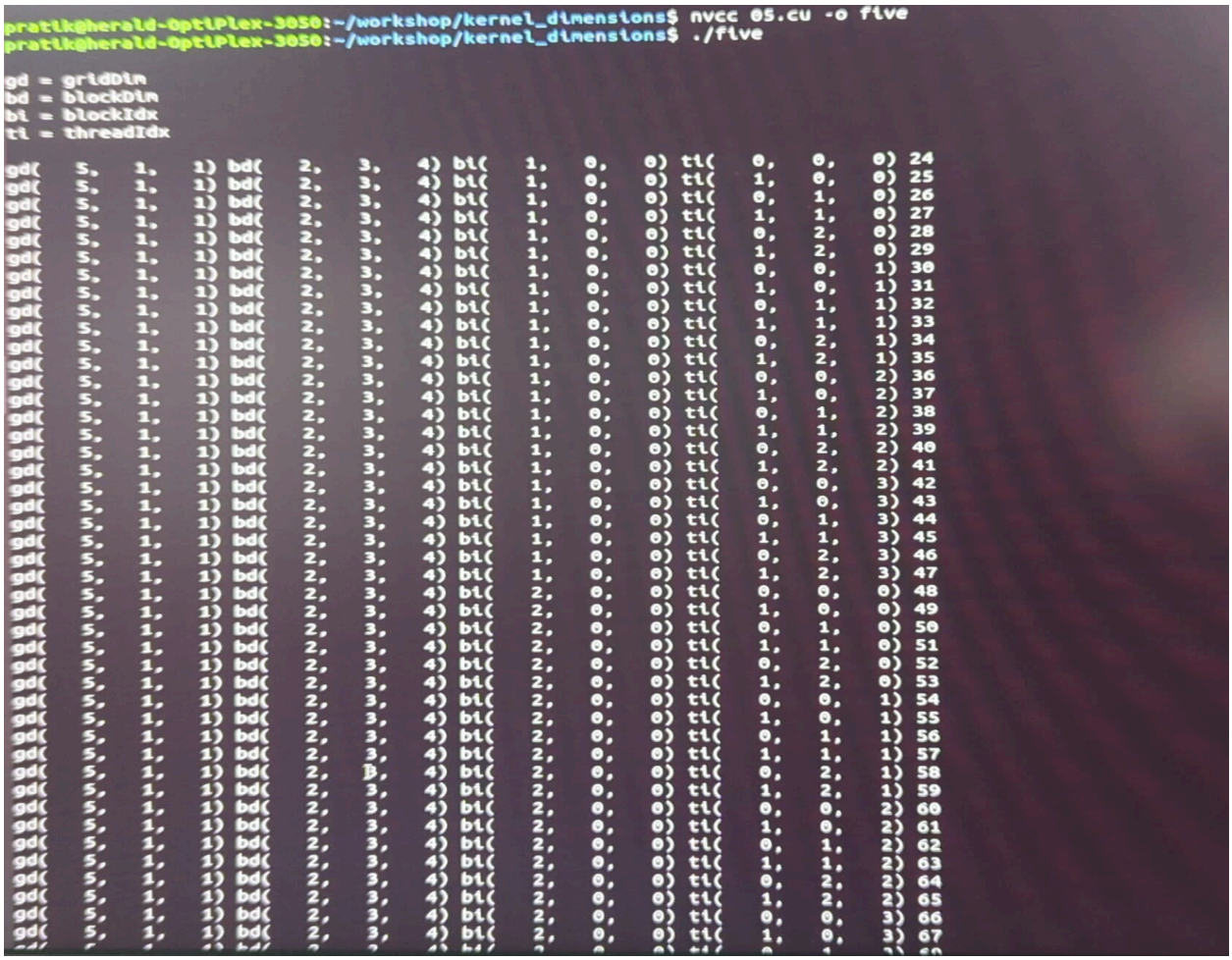
3. Here we have used dim3 bd(2,3) for block dim, and the grid dimension is default as (1,1,1). So we launch the kernel as kernel<<<1,bd>>>. Hence, we have **1 block in x direction** and because of the dimension of the block **we'll have 2 threads in x direction and 3 threads in y direction**. Hence totaling 6 threads in one block and as we have one block **we'll have 6 threads in total**.



4. Here we have used dim3 bd(2,3,4) for block dim, and the grid dimension is default as (1,1,1). So we launch the kernel as kernel<<<1,bd>>>. Hence, we have 1 block in x direction and because of the dimension of the block we'll have **2 threads in x direction, 3 threads in y direction,and 4 threads in z direction.** Hence totaling **24 threads in one block** and as we have one block **we'll have 24 threads in total**.



5. Here we have used dim3 bd(2,3,4) for block dim, and the grid dimension is  set to (5,1,1). So we launch the kernel as kernel<<<5,bd>>>. Now, we have **5 blocks in x direction** and because of the dimension of the block **we'll have 2 threads in x direction, 3 threads in y direction,and 4 threads in z direction**. Hence totaling **24 threads in one block and as we have one 5 blocks we'll have 120 threads in total**.



7. Here are the understanding i got after testing various dimensions:
    - When we use print statements in cuda it cannot directly print the result in the host terminal so it has a fixed sized print buffer to hold any print statement which is later printed in the terminal after the device is synchronized. **So, printing a lot of information can lead to buffer overflow**
    - Threads execute in warps of 32, so for better efficiency we need to make **total threads per block multiple of 32**.
    - More blocks offers better SM utilization for better load balancing.
    - For example, for a total of 4096 threads: 16 blocks with 256 threads each will give better performance than a setup of 8 blocks with 512 threads. Although, in both the cases the threads are exactly multiple of 32 but the number of blocks take the weight here for performance.
    - Making 2D 3D thread will not give better performance, we need to follow the above steps for better performance no matter if we use 1-D,2-D, or even 3-D dimensions. (**NOT 100% sure**)

Hence, among the various configurations of the kernel launch. The one with bd(5, 3, 4) gd(2, 3, 4) is better as it has 60 threads per block which **gives better wraps per block**. Also it has more blocks which promotes better SM utilization.