ASSIGNMENT NO – 2

1- For the naive reduction kernel, how many steps execute without divergence? How many steps execute with divergence?

Ans: -

Block Size – 512

No of steps: log2(512) = 9

So, there are 9 Steps through execution.

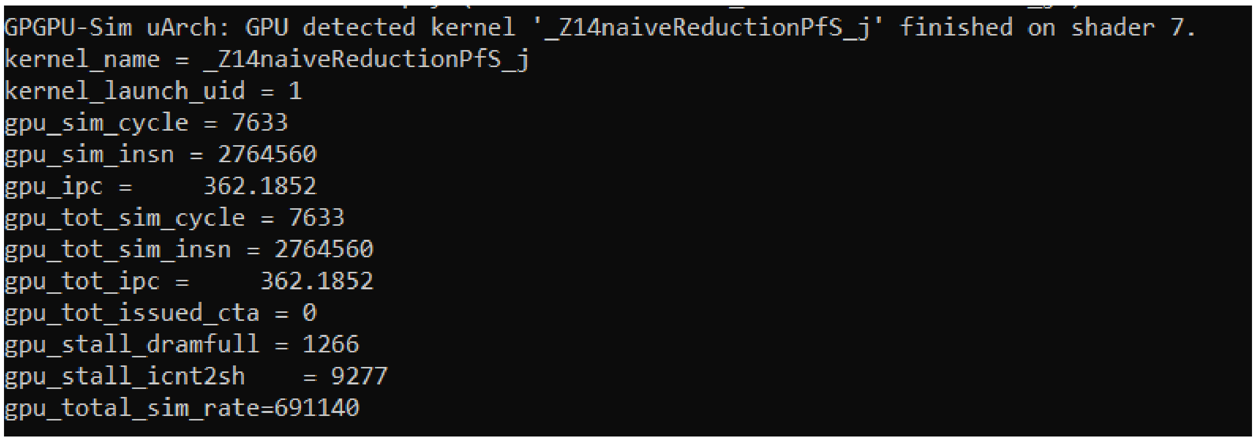
In the first step, all the threads are used to compute the partial sum, so there is no divergence. Half of the threads after the first steps will not have any task, and the other eight steps execute divergently.

2-For the optimized reduction kernel, how many steps execute without divergence? How many steps execute with divergence?

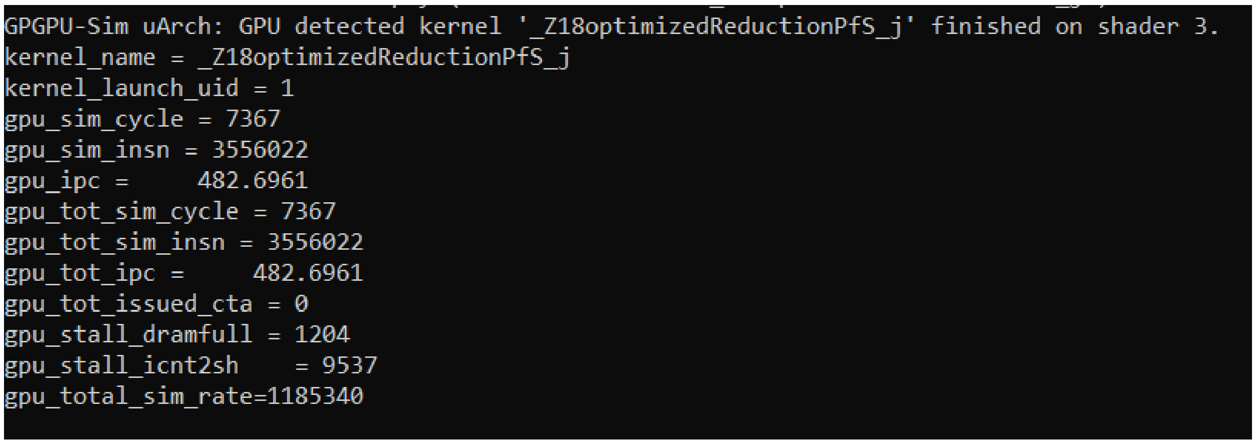
Ans: -

There is no divergence in the first four steps; the number of effective threads in the first four steps is smaller than a warp. In contrast, the last five steps are executed divergently.

3- Which kernel performed better? Use profiling statistics to support your claim.



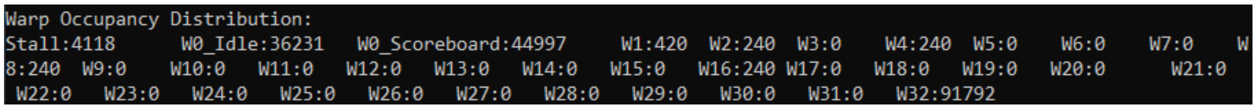
```
GPGPU-Sim uArch: GPU detected kernel '_Z14naiveReductionPfS_j' finished on shader 7.
kernel_name = _Z14naiveReductionPfS_j
kernel_launch_uid = 1
gpu_sim_cycle = 7633
gpu_sim_insn = 2764560
gpu_ipc =     362.1852
gpu_tot_sim_cycle = 7633
gpu_tot_sim_insn = 2764560
gpu_tot_ipc =     362.1852
gpu_tot_issued_cta = 0
gpu_stall_dramfull = 1266
gpu_stall_icnt2sh    = 9277
gpu_total_sim_rate=691140
```



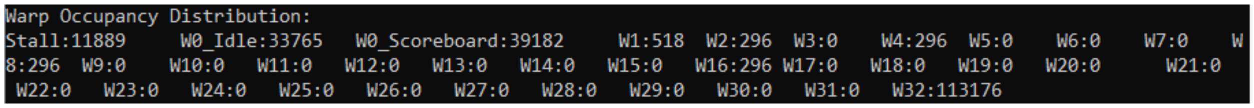
```
GPGPU-Sim uArch: GPU detected kernel '_Z18optimizedReductionPfS_j' finished on shader 3.
kernel_name = _Z18optimizedReductionPfS_j
kernel_launch_uid = 1
gpu_sim_cycle = 7367
gpu_sim_insn = 3556022
gpu_ipc =     482.6961
gpu_tot_sim_cycle = 7367
gpu_tot_sim_insn = 3556022
gpu_tot_ipc =     482.6961
gpu_tot_issued_cta = 0
gpu_stall_dramfull = 1204
gpu_stall_icnt2sh    = 9537
gpu_total_sim_rate=1185340
```

Here, naïve reduction kernel has high value in attribute (gpu_sim_cycle, gpu_tot_sim_cycle) and has lower values in other attributes as compare to optimized reduction. Hence, a reduction kernel that is optimized performs better than a reduction kernel that is naive.

4- How does the warp occupancy distribution compare between the two Reduction implementations?



```
Warp Occupancy Distribution:
Stall:4118      W0_Idle:36231   W0_Scoreboard:44997     W1:420  W2:240  W3:0    W4:240  W5:0    W6:0    W7:0    W
8:240  W9:0    W10:0   W11:0   W12:0   W13:0   W14:0   W15:0   W16:240 W17:0   W18:0   W19:0   W20:0      W21:0
 W22:0   W23:0   W24:0   W25:0   W26:0   W27:0   W28:0   W29:0   W30:0   W31:0   W32:91792
```



```
Warp Occupancy Distribution:
Stall:11889     W0_Idle:33765   W0_Scoreboard:39182     W1:518  W2:296  W3:0    W4:296  W5:0    W6:0    W7:0    W
8:296  W9:0    W10:0   W11:0   W12:0   W13:0   W14:0   W15:0   W16:296 W17:0   W18:0   W19:0   W20:0      W21:0
 W22:0   W23:0   W24:0   W25:0   W26:0   W27:0   W28:0   W29:0   W30:0   W31:0   W32:113176
```

Warp occupancy distributions of naive reduction kernels have fewer stalled warps than the warp occupancy distributions of the optimized reduction kernels, meaning optimized reduction kernels are more efficient.

5- Why do GPGPUs suffer from warp divergence?

Ans- In SIMD, GPGPUs perform better than CPUs because data processing and scheduling take up a much more significant portion of their time than data transmission and scheduling. In SMID, data is processed in the same code. Several ways will result in serialization, which lowers the performance of GPGPUs and causes them to experience warp divergence.