CS217 – LAB4

Shahriyar Valielahi Roshan – 861241935

1- You can find related statistics here. I values are reported for execution of the code without any argument as an input (Default 1000000 4096). Please take a look at attached screen shots.
getHisto:

Metric	Value
Duration	205.9µs
Grid Size	977
Block Size	1024
Registers/Thread	11
Shared Memory/Block	16KiB
Occupancy	94.2%
Warp Execution Efficiency	93.2%
Global Memory Load Efficiency	100%
Global Memory Store Efficiency	**N/A**
Warp Non-predicted Execution Efficiency	87.9
Issued Control Flow Instructions	816491
Executed Control Flow Instructions	31392
Control Flow Ratio	**0.04%**

Convert_32_8:

Metric	Value
Duration	27.584µs
Grid Size	977
Block Size	1024
Registers/Thread	8
Shared Memory/Block	0KiB
Occupancy	**52.5%**
Warp Execution Efficiency	100%
Global Memory Load Efficiency	100%
Global Memory Store Efficiency	**50%**
Warp Non-predicted Execution Efficiency	99.8
Issued Control Flow Instructions	31392
Executed Control Flow Instructions	31392
Control Flow Ratio	100%

Low global memory store Efficiency is due to the poor alignment or access pattern.

2- We have limitations one size of m, and n. The problem on m is the limitation on a total number of blocks that a device can handle, we do not have infinitive number of blocks, and number of blocks is proportional to the block size, the size of input. We can make blocks bigger to overcome this problem but when we use the biggest possible blocks size, and the input size lead to a problem that number of blocks exceed the maximum number of blocks, we have only one solution to divide the problem into smaller problems, and have kind of hierarchy.
And the limitation on the bin size is that we have limited shared memory available. We should use dynamic shared memory, or we should divide each step into several steps that is going to be done by more threads rather than one. For example thread one calculate [0-1023] histo and so one, and we should somehow merge results together.