8 BONUS: GPU Programming [50 points]

An inexperienced CUDA programmer is trying to optimize her first GPU kernel for performance. After writing the first version of the kernel, she wants to find the best *execution configuration* (i.e., grid size and block size[1]).

As she assigns one thread per input element, calculating the grid size (i.e., total number of blocks) is trivial. For N input elements, the grid size is $\lceil \frac{N}{block\_size} \rceil$, where $block\_size$ is the number of threads per block. So, the challenging part will be to figure out what is the block size that produces the best performance. She will try 5 different block sizes (64, 128, 256, 512, and 1024 threads).

She has learned that a general recommendation for kernel optimization is to maximize the *occupancy* of the GPU cores, i.e., Streaming Multiprocessors (SMs). Occupancy is defined as the ratio of active threads to the maximum possible number of active threads per SM.

In order to calculate the occupancy, it is necessary to take the available SM resources into account. She knows that in each SM of her GPU:

- The total *scratchpad memory* or *shared memory* is 16 KB.
- The total *number of 4-byte registers* is 16384.

In her first version of the kernel code, each thread needs 2 4-byte elements in shared memory for its private use. In addition, each block needs 10 4-byte elements in shared memory for communication across threads.

She has also learned that she can obtain the number of registers that each thread needs by using a special compiler flag. This way, she finds that each thread in the first version of the kernel uses 9 registers.

(a) [15 points] After reasoning some time about the amount of shared memory that her code needs, she decides to first test a block size of 128 threads. Why do you think she chose that number? Show your work.

She calculated the maximum number of threads that an SM can hold according to the shared memory usage. 128 threads per block results in that maximum.

Explanation:
Given the shared memory needs of her code, each block uses $2 \times block\_size + 10$ 4-byte elements, that is, $4 \times (2 \times block\_size + 10)$ bytes.
Since the shared memory available per SM is 16 KB, the number of blocks and the number of threads that each SM can allocate is as follows:

$block\_size$	Blocks/SM	Threads/SM
64	29	1856
128	15	**1920**
256	7	1792
512	3	1536
1024	1	1024

She decides to test a block size of 128 threads because this achieves the highest number of active threads per SM (i.e., the highest occupancy).

[1] We use NVIDIA terminology in this question.

However, after testing other block sizes, she finds out that using 256 threads per block provides higher performance than using 128. She does not understand why, so she looks for some information in the documentation of her GPU that can lead her to an explanation. There, she finds two more SM hardware constraints. In each SM:

- The *maximum number of blocks* is 8.
- The *maximum number of threads* is 2048.

Take into account these new constraints when answering parts (b), (c), and (d).

(b) [10 points] Can you explain why using 256 threads per block perform better than 128? Show your work.

The limitation in the maximum number of blocks per SM makes that the configuration with the highest occupancy is 256 threads per block.

Explanation:
This is the corrected table after taking the limitation in the maximum number of blocks into account:

block_size	Blocks/SM	Threads/SM
64	8	512
128	8	1024
256	7	**1792**
512	3	1536
1024	1	1024

(c) [15 points] What is the occupancy limitation due to register usage, if any? Explain and show your work.

There is *no* occupancy limitation due to the register usage.

Explanation:
The register usage depends on the block size. As she knows the number of registers per thread (9), she can calculate the total register needs:

block_size	Blocks/SM	Threads/SM	Registers/block	Registers/SM
64	8	512	576	4608
128	8	1024	1152	9216
256	7	1792	2304	16128
512	3	1536	4608	13824
1024	1	1024	9216	9216

In all cases, the total register usage is lower than 16384. (NOTE: The solution is correct if only calculated for 256 threads.)

(d) [10 points] The performance obtained by the first kernel version does not fulfill the acceleration needs. Thus, the programmer writes a second kernel version that reduces the number of instructions at the expense of using one more register per thread. What would be the highest occupancy for the second kernel? For what block size(s)?

The highest occupancy will be $\frac{1536}{2048} = 0.75$. It can be obtained with blocks of 256 or 512 threads.

Explanation:
The number of registers per thread is 10 in the second kernel. The configuration with 256 threads per block for the second kernel version will be able to allocate one less block per SM (6) than for the first kernel version (7):

block_size	Blocks/SM	Threads/SM	Registers/block	Registers/SM
64	8	512	640	5120
128	8	1024	1280	10240
256	6	**1536**	2560	15360
512	3	**1536**	5120	15360
1024	1	1024	10240	10240