Graphics Processing Units (GPUs): Architecture and Programming
Homework Assignment #2
(total: 30 points)

1. [5 points] No, 32 threads can be divided into two blocks (hence two warps) vs being assigned to one block (hence one warp).
#warps = #blocks * #warps per block = #blocks * ceil(t/#blocks/32).

---

2. [2 points]

- To amortize the cost of fetching/decoding instructions
- To reduce hardware needed for non-computational purposes.

3.
a. [1] Four warps will have branch divergence. Each if-condition corresponds to a thread in a different warp, causing a branch divergence.
[1] Those affected warps are number: 0, 1,2, and 6.
[1] We know deterministically that the first 32 threads are in warp 0, then threads from 32 to 63 are in warp 1, and so on. So, from the thread ID we can determine the warp number.

b. [1] No, we may get different values because there is a race condition in writing to shared variable sum.
[2] The values that sum can have are: 40, 70, 200, 110, 240, 270, and 310.

4.
a) [1 point] Each block has 256 threads. And we cannot have a fraction of a block so we need:
ceil (3000/256) = 12
This means we will have 12*256 = 3072

b) [2 points: 1 for the answer and 1 for the explanation] If warp size = 32 threads and if we need to add an if-condition to ensure no thread will be calculating anything outside the 3000 threads, how many warps will have thread divergence? And how did you reach that?

Only one.
We have 72 threads above 3000 based on the solution a) above. This includes 2 warps outside the 3000 where all the conditions will the false, so no thread/branch divergence for them. Then the third warp will have 8 threads outside 3000 and 24 inside the 3000 borders. So, only this warp will have branch divergence.

5

a. [2] How many threads are there in total?

VECTOR_N blocks * ELEMENT_N threads / block = 256*1024 threads in total

b. [2] How many threads are there in a warp?

Current GPUs assign 32 threads to a warp

c. [2] How many threads are there in a block?

ELEMENT_N = 256 threads (second configuration argument of the kernel launch on line 10)

d. [2] How many global memory loads and stores are done for each thread?

Every thread performs two global memory loads on line 21 (one each from A and B). Every thread also writes one value on line 30. Therefore, each thread performs 3 global memory accesses in total.

e. [3] How many accesses to shared memory are done for each block?

On line 21, 256 writes (one for each thread) to shared memory occur. When a thread executes line 27, it performs two shared memory reads, adds the two values together, and then performs a shared memory write to store the result.
Given the loop structure, on the first iteration, 128 threads will execute the statement, then 64 on the following statement, and so on.
Finally, each thread reads element 0 on line 30, accounting for another 256 accesses.
Therefore, the total number of accesses is
256 + (128 + 64 + 32 + 16 + 8 + 4 + 2 + 1)*3 + 256 = 256 + 255*3 + 256 = 1277 accesses.

f. [3] How many iterations of the for loop (Line 23) will have branch divergence? Show your derivation.

As described in the solution to (e) above, on the first iteration, 128 threads will execute statement 27, while 128 will not. On the second iteration, 64 threads will execute statement 27, while 192 will not, and so on. Furthermore, the threads executing statement 27 on each iteration are threads with indexes of 0 to stride-1: a contiguous set. Divergence will only occur in iterations where stride is not a multiple of the warp size, or 32. We can see that there are five such iterations, specifically the last five iterations of the loop.