MP5.2

(1pt)
1. For the Brent-Kung scan kernel, which of the following data structures has the most similar memory access pattern?
A. Queue
B. Tree
C. Stack
D. LinkedList
E. None of the above.

Solution: (B).
The GPU Reduce-then-Scan strategy utilizes the similar idea of recursion, by calculating the partial sum of a lot of smaller questions as parts of the solution for the upper layer. So if we draw the Brent-Kung process out by hand, we will find its memory access pattern looks like a tree.

(2pts)
2. How many times does a single thread block executing the Brent-Kung work-efficient kernel perform barrier synchronization while producing its portion of the scan array?
A. 2*log(2,BLOCK_SIZE)
B. 2*log(2,BLOCK_SIZE)+1
C. 2*log(2,BLOCK_SIZE)+2
D. 2*log(2,BLOCK_SIZE)+3
E. None of the above.

Solution: (C).
Each thread block takes 2*BLOCK_SIZE inputs and generates 2*BLOCK_SIZE outputs. For reduction part, There are log(2,BLOCK_SIZE) + 1 steps of reductions and log(2, BLOCK_SIZE) + 1 numbers of __syncthreads() are performed; for post scan part, there are log(2, BLOCK_SIZE) steps, and log(2, BLOCK_SIZE) numbers of __syncthreads() are performed. And there is one __syncthreads() performed after post scan and before writing results.
So, there are (2*log(2, BLOCK_SIZE)+2) __syncthreads() performed in total.

(1pt)
3. For the Brent-Kung scan kernel based on reduction trees and inverse reduction trees, assume that we have 64 elements in each section and warp size is 32, how many warps in each block will have control divergence during the inverse tree phase iteration where stride is 1?
A. 4
B. 2

C. 1
D. 0
E. None of the above.

Solution: (C).
We have 64 elements in each section, which implies 64/2 = 32 threads = 1 warp. Stride is 1 implies this is the last step. All threads will perform the addition except the last thread.

(2pts)
4. For the Kogge-Stone scan kernel based on reduction trees, assume that we have 1024 elements in each section and warp size is 32, how many warps in each block will have control divergence during the iteration where stride is 8?
A. 32
B. 8
C. 2
D. 0
E. None of the above.

Solution: (E) or 1
When stride is 8, only the first warp has control divergence.

(2pts)
5. In the hierarchical complete scan approach for a long input array with n float elements, the block size of the first scan kernel is BLOCK_SIZE (n > 2 * BLOCK_SIZE), we use two different arrays in GPU global memory for input array of original input data and output array of final scan result, how many bytes of GPU global memory should you allocate at the minimum? Keep in mind that the hierarchical complete scan approach needs two different scan kernels to do scan computation in a hierarchical manner, so, an additional auxiliary array in GPU global memory is needed to temporarily save some intermediate results. Note that in this problem, it does not matter whether you use Kogge-Stone algorithm or Brent-Kung algorithm. (Hint: is every element in the auxiliary array necessary for the last kernel in the hierarchical scan approach?)
(Assuming element type is float; each float is 4 bytes)
A. 8*n+4*(ceil(n/(float)(BLOCK_SIZE))-1)
B. 8*n+4*ceil(n/(float)(BLOCK_SIZE))
C. 8*n+8*ceil(n/(float)(BLOCK_SIZE))
D. 8*n+8*(ceil(n/(float)(BLOCK_SIZE))-1)
E. 16*n+8*ceil(n/(float)(BLOCK_SIZE))

Solution: (A).

Scan kernels can use the same array in GPU global memory for input and output. We use two arrays in GPU global memory for original input data and final scan output, these two arrays need at least 2*4*n = 8*n bytes.
The first scan kernel can use input array in global memory for both its kernel input and its kernel output, or use input array in global memory for its kernel input and use output array in global memory for its kernel output; so we don't need to allocate more bytes for the first scan kernel.
The second scan kernel requires an auxiliary array with at least ceil(n/(float)(BLOCK_SIZE))-1 float elements since the last block in the first kernel won't contribute to the block-wise additions done by the last kernel. The last kernel can do in-place addition, we don't need to allocate more bytes for it. Therefore, 8*n+4*(ceil(n/(float)(BLOCK_SIZE))-1) bytes need to be allocated in GPU global memory at the minimum.

(1pt)
6. How many float-operations are performed by all the threads of the grid as a result of launching the Kogge-Stone scan kernel and the Brent-Kung scan kernel (using big O notation)? Answer and explain respectively.

Solution:
Kogge-Stone: O(n*log(n)). **(0.5pts)**

For the Kogge-Stone method, we have totally O(log(2,n)) iterations. For every iteration, we have (n-1),(n-2),(n-4),. . . ,(n-n/2) numbers of float additions respectively. (n-1)+(n-2)+(n-3)+. . . +(n-n/2) generates O(n*log(n)).

So, we have O(n*log(n)) float additions in total. And for this kernel float addition is the only float operation, therefore O(n*log(n)) float operations are performed totally.

Brent-Kung: O(n). **(0.5pts)**

For the Brent-Kung method, we have O(log(n)) for iterations of reduction, and O(log(n)) for iterations of post scan. For every iteration, we do n/2,n/4,n/8,. . . ,2,1,(2-1),(4-1),. . . ,(n/4-1),(n/2-1) float additions. n/2+n/4+n/8+. . . +2+1+(2-1)+(4-1)+. . . +(n/4-1)+(n/2-1) generates O(n). And for this kernel float addition is the only float operation, therefore O(n) float operations are performed totally.

(1pt)
7. How many bytes are read from global memory by all the threads of the grid as a result of launching a Brent-Kung Scan kernel(consider only the first kernel in the overall hierarchical approach)? How many bytes are written to global memory? The length of input array is N, the block size is BLOCK_SIZE and we are using float in our implementation. Please give answers separately for reads and writes and explain your answer for full credit. (Assume input array with arbitrary length which need to be distributed into multiple blocks).

Solution:
Global memory read : 4*N bytes. In the first kernel, each block(maybe except the last block) always accepts 2*BLOCK_SIZE numbers of inputs and generates 2*BLOCK_SIZE number of outputs. Each block will always perform global memory read for 2*BLOCK_SIZE times(maybe except the last block), and totally N global memory reads will be performed. So, the total bytes read from global memory is 4*N.

Global memory write: 4*N bytes. In the first kernel, each block(except the last block) always accepts 2*BLOCK_SIZE numbers of inputs and generates 2*BLOCK_SIZE number of outputs. Each block will always perform global memory writes for 2*BLOCK_SIZE times(except the last block), and totally N global memory will be performed. So, the total bytes written to global memory is 4*N.

0.5 pts for each
For memory write, it is okay to include the writes to the auxiliary array.
Give full points as long as the answer includes 4N.