GPU Architectures
and Programming
Assignment- Week 7
TYPE OF QUESTION: MCQ/MSQ

Number of questions: 10 **Total mark: 10 X 1 = 10**

MCQ/MSQ Question

Question 1:

Consider an array **A** where A[i]=1 if i is divisible by 2 and A[i] = 0 otherwise. The array has 1024 elements. Let us consider a reduction which computes the addition of all the elements of the array. What is the final value of this reduction operation?

A. 1024
B. 512
C. 2048
D. 256

Answer: B

Solution:
Given that A[i]=0 or 1 based on divisibility of 2, half the elements in the range [0,1023] would be 0 while half the elements would be 1. So the total sum would be 512*1=512.

Question 2:

Consider the same array **A** with 1024 elements where A[i]=1 if i is divisible by 2 and A[i] = 0 otherwise. Let us consider the most naive reduction kernel but with the XOR operation as the reduction operation in question. What is the final value of this XOR reduction operation after applying it on A?

A. 0
B. 1

Answer: A

Solution: A contains an even number of 0s and even number of 1. Applying a XOR over the entire sequence would reduce it to 0.

Question 3:

Consider the following reduction kernel code snippet.

```
__global__ void reduce ( int * g_idata , int * g_odata , unsigned int n ) {
extern __shared__ int sdata [];
unsigned int tid = threadIdx . x ;
unsigned int i = blockIdx . x * blockDim . x + threadIdx . x ;
sdata [ tid ] = ( i < n ) ? g_idata [ i ] : 0;
__syncthreads () ;
for(unsigned int s=1; s < blockDim.x; s *= 2)
{
int index = 2 * s * tid;
if (index < blockDim.x)
sdata [ index ] += sdata [ index + s ];
__syncthreads () ;
}
if ( tid == 0)
g_odata [ blockIdx . x ] = sdata [0];
}
```

Which of the following options is correct?

A. The kernel suffers only from high divergence.

B. The kernel suffers only from shared memory bank conflicts.

C. The kernel suffers from high divergence as well as memory bank conflicts.

D. None of the above

Answer: B

Solution: Refer to slides for Reduction 2 kernel

Question 4:

Consider the reduction kernels taught in the lecture. Which among them completely unrolls the loop using the template parameter?
A. Reduction 2
B. Reduction 3
C. Reduction 4
D. Reduction 6

Answer: D

Question 5:

In a CUDA reduction kernel, if the input array size is n=1024 and threads per block are 32, how many total threads are required in the second kernel invocation, assuming two elements are processed per thread in the first iteration?
A. 32
B. 64
C. 128
D. 256

Answer: D

Solution:
Input size: n=1024
Threads per block: 3
Elements processed per thread in the first kernel: 2

Step 1: Reduction after the first kernel invocation

Each thread processes 2 elements in the first kernel invocation. Therefore, the total number of threads required is:

Threads needed after the first kernel=n/2=1024/2=512

Step 2: Reduction after the second kernel invocation

In the second kernel invocation, each thread will again process 2 elements. Therefore, the total number of threads required is:

Threads needed after the second kernel=512/2=256

Question 6:

In a parallel reduction kernel, the size of the input array is 210, and each block has 64 threads. Assuming each thread processes two elements in the first iteration, how many blocks are required for the first kernel invocation?

A. 4
B. 2
C. 8
D. 16

Answer: B

Solution:

Input size: n=210
Threads per block: 64
Elements processed per thread: 2

Step 1: Total number of threads required

Since each thread processes 2 elements, the total number of threads required is:

Threads required=n/2=210/2=105

Step 2: Number of threads per block

Each block has 64 threads. The number of blocks required is:

Blocks required=⌈ Threads required / Threads per block⌉ =⌈ 105/64⌉ =⌈ 1.64⌉ =2

Question 7

Which of the following statements is correct?

A. Inclusive scan of an array generates a new array where each element j is the sum of all elements up to and excluding j.
B. Exclusive scan of an array generates a new array where each element j is the sum of all elements including j
C. A Bitonic Sequence is a sequence of numbers which is first strictly decreasing then after a point strictly increasing.
D. In the most naive version of the reduction operation, half of the threads are idle on first loop iteration while executing the kernel.

Answer: D

Solution: Refer slides of lecture.

Question 8:

In the given code snippet for reduction using Algorithm Cascading, there are some mistakes. Identify the line number(s) for the probable error(s). Multiple choices can be possible for this question.

```
unsigned int tid = threadIdx .x;
unsigned int i = blockIdx .x * 2 * ( blockDim .x) + threadIdx .x;
unsigned int gridSize = blockSize*2*gridDim.x;
sdata[tid] = 0;
while (i < n) {
    sdata[tid] += g_idata[i] + g_idata[i+gridSize];
    i ++;
}
```

A) 3
B) 4
C) 7
D) 8

Answer: C,D

Solution:

```
sdata[tid] += g_idata[i] + g_idata[i+blockSize];
i += gridsize
```

Question 9:

For the following code snippet for reduction, find the total number of memory read accesses between line number 6 and 10 per block. The blockDim.x is 64 and warp size is 16.

Kernel

- unsigned int tid = threadIdx .x;
- unsigned int i = blockIdx .x* blockDim .x + threadIdx .x;
- __shared__ float sdata[64];
- sdata [tid] = (i < n) ? g_idata [i] : 0;
- __syncthreads ();
- for (unsigned int s=blockDim.x/2; s>0; s»=1)
- { if (tid < s)
- sdata [tid] += sdata [tid + s];
- __syncthreads () ;
- }

Number of write accesses :

A) 10

B) 5

C) 4

D) 20

Answer: C

QUESTION 10:

Referring to the above code snippet find the total number of memory read accesses between line number 6 and 10 per block as described in question 9.

Number of read accesses :
A)10
B) 20
C) 23
D) 30

Answer: B

Solution of question 9 and 10:

In each block, Total number of warps will be 64/16=4.

In each warp, total number of write accesses = 1

In each warp, for each possible value of s, number of read accesses :

32-> 2

16->2

8->1

4,2,1->0

In each warp, total number of read accesses = **2+2+1=5**

Total number of read accesses per block is 5*4=20

Total number of write accesses per block is 1*4=4

**********END**********