ST2 PROGRAMMING QUESTIONS (DIVERGENCE AND WARP SCHEDULING)

QUESTION 1:

Assume there is a 1D input array A of N integers, where each element is of the form A[i]=2*i , i = 0 to N-1. Consider the following 1D kernel code snippet executing on a GPU architecture where the warp size is 8.

```
__global__ void divBranch(int* A, int* B, int M, int N, int k){
     int tid = blockIdx.x * blockDim.x + threadIdx.x;
     if(A[tid]%8) {
       B[tid]+=sqrt(A[tid]);
      if(A[tid]%4) {
               B[tid]+=sqrt(A[tid]); B[tid]+=sqrt(A[tid]);
               B[tid]+=sqrt(A[tid]); B[tid]+=sqrt(A[tid]);
      }
      else {
         B[tid]+=sqrt(A[tid]); B[tid]+=sqrt(A[tid]);
       }
     }
}
```

The above code represents a kernel with nested levels of divergence. During the lifetime of warp '0', what are the total number of square root instructions that are executed.

Options:

A. 18

B. 16

C. 32

D. 26

E. None of the above

Answer: D

QUESTION 2

Assume there is a 1D input array A of N integers, where each element is of the form A[i]=2*i, , i = 0 to N-1. Consider the following kernel code snippet executing on a GPU architecture where the warp size is 32.

```
__global__ void divBranch(int* A, int* B, int M, int N, int k)
{
   int tid = blockIdx.x * blockDim.x + threadIdx.x;
    if(!A[tid]%8)
    {
        B[tid]+=sqrt(A[tid]);
      if(!A[tid]%4)
      {
         B[tid]+=sqrt(A[tid]);B[tid]+=sqrt(A[tid]);
          B[tid]+=sqrt(A[tid]);B[tid]+=sqrt(A[tid]);
       }
       else
       {
        B[tid]+=sqrt(A[tid]);B[tid]+=sqrt(A[tid]);
       }
    }
 }
```

The above code represents a kernel with nested levels of divergence. During the lifetime of warp 0 what are the total number of square root instructions that are executed ?

Options:

A. 18

B. 16

C. 32

D. 40

E. None of the above

Answer: E

QUESTION 3:

Assume a hypothetical GPU architecture that has 12 SMs and 564 SPs per SM. Consider a kernel being executed with the launch parameters <<<(16,16,1),(16,16,1)>>>

The occupancy of a CUDA kernel is equal to the number of active threads executing in an SM divided by the total number of SPs in an SM. Since the number of blocks being scheduled is much greater than the number of SMs, thread blocks will be scheduled in batches. What are the number of batches and the occupancy for the CUDA kernel?

A. 11, 90%
B. 8, 70%
C. 7, 60%
D. 9, 40%
E. None of the above

Answer A

QUESTION 4

Assume a hypothetical GPU architecture that has 6 SMs and 160 SPs per SM. Consider a kernel being executed with launch parameters <<<(8,8,1), (64,1,1)>>> The occupancy of a CUDA kernel is equal to the number of active threads executing in an SM divided by the total number of SPs in an SM. Since the number of blocks being scheduled is much greater than the number of SMs, thread blocks will be scheduled in batches. What are the number of batches and the occupancy for the CUDA kernel?

Options:

A. 8, 40%
B. 6, 80%
C. 6, 40%
D. 8, 80%
E. None of the above

Answer: B

QUESTION 5

Consider the following kernel snippet executing on a hypothetical GPU architecture where the warp size is 8. Assume that the number of threads in a thread block at the time of launching the kernel is 32.

```
__global__ void kernel1(float *A)
{
     int tid = threadIdx.x;
     if(tid%2 != 0)
     {
          if(tid%3 == 0)
          {
               A[tid]++;
          }
     }
     else
     {
       A[tid]--;
     }
}
```

Let us define warp utilization as the percentage of threads active in a given warp. A thread is active in the above program if it is performing an increment or a decrement operation. The warp utilization differs for each warp instance in a block of threads in the above program. What are the minimum and maximum warp utilization factors in the above program, considering an architecture where the warp size is 8 and the thread block size at the time of launching the kernel is 32?

Select the correct option from below

A. min = 12.5% , max = 50%
B. min = 12.5%, max = 25%
C. min = 25% , max = 25%
D. min = 25%, max = 75%

ST2 PROGRAMMING QUESTIONS (MEMORY)

QUESTION 1

Consider the following code snippet executing on a hypothetical GPU architecture where warp size is 8. The global memory access width = |warp|*4 bytes = 32 bytes.

```
__global__ void mem_access(float *A, float *B, int n)
{
      int tidx = blockIdx.x*blockDim.x + threadIdx.x;
      for(int i=0;i<n;i++)
          A[tidx]+=B[A[tidx]];
      __syncthreads();
}
```

Array A is of length 256 and B is of length 264, and each of the elements in the arrays are A[i]=i and B[i]=2. The value of n = 5. Ignoring the effects of caching, the total number of global load transactions for the array B by warp '0' is _____

Answer: 8

Consider the following code snippet executing on a GPU architecture where warp size is 8. The global memory access width = |warp|*4 bytes = 32 bytes.

QUESTION 2

```
__global__ void mem_access(float *A, float *B, int n)
{
      int tidx = blockIdx.x*blockDim.x + threadIdx.x;
      for(int i=0;i<n;i++)
          A[tidx]*=B[A[tidx]];
      __syncthreads();
}
```

Array A is of length 256 and array B is of length 1024, and each of the elements in the arrays are A[i]=i and B[i]=2. The value of n = 3. Ignoring the effects of caching, the total number of global load transactions for array B by warp '0' is _____

Answer 7

QUESTION 3:

Consider the following code snippet executing on a GPU architecture where warp size is 8.

The global memory access width = |warp|*4 bytes = 32 bytes.

```
__global__ void mem_access(float *A, float *B, int n)
{
     int tidx = blockIdx.x*blockDim.x + threadIdx.x;
     for(int i=1;i<n;i=i*2)
         A[tidx]=B[A[tidx]]+i;
     __syncthreads();
}
```

Array A is of length 256 and B is of length 264, and each of the elements in the arrays are A[i]=i and B[i]=i. The value of n = 8. Ignoring the effects of caching, the total number of global load transactions for the array B by warp '0' is _____

Answer: 7

QUESTION 4:

Consider the following code snippet executing on a GPU architecture where warp size is 8.

The global memory access width = |warp|*4 bytes = 32 bytes.

```
__global__ void mem_access(float *A, float *B, int n)
{
     int tidx = blockIdx.x*blockDim.x + threadIdx.x;
     for(int i=2;i<n;i=i*2)
         A[tidx]=B[A[tidx]]*i;
     __syncthreads();
}
```

Array A is of length 256 and B is of length 2048, and each of the elements in the arrays are A[i]=i and B[i]=i. The value of n = 16. Ignoring the effects of caching, the total number of global load transactions for the array B by warp '0' is _____

Answer: 11

QUESTION 5:

Consider the following code snippet executing on a GPU architecture where warp size is 8.

The global memory access width = |warp|*4 bytes = 32 bytes.

```
__global__ void mem_access(float *A, float *B, int n)
{
     int tidx = blockIdx.x*blockDim.x + threadIdx.x;
     for(int i=1;i<n;i++)
         A[tidx]=B[i+A[tidx]];
     __syncthreads();
}
```

Array A is of length 256 and B is of length 2048, and each of the elements in the arrays are A[i]=i and B[i]=i. The value of n = 4. Ignoring the effects of caching, the total number of global load transactions for the array B by warp '0' is _____

Answer: 6