Tutorial-2 GPU Memory Access

Question 1

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 2

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

```
1.  unsigned int tid = threadIdx .x;
2.  unsigned int i = blockIdx .x* blockDim .x + threadIdx .x;
3.  __shared__ float sdata[64];
4.  sdata [ tid ] = (i < n) ? g_idata [i] : 0;
5.  __syncthreads ();
6.  for (unsigned int s=blockDim.x/2; s>0; s»=1)
7.  {       if (tid < s)
8.            sdata [ tid ] += sdata [tid + s];
9.        __syncthreads () ;
10. }
```

Number of write accesses :

A. 10
B. 5
C. 4
D. 20

Answer: C

Solution:

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

Question 3

You are performing matrix multiplication on a GPU for two matrices A and B containing integers, both sized 4096×4096, resulting in a matrix C that is also of the same size. The GPU has a warp size of 32.

The multiplication kernel uses tiling with size 32×32 and thread block dimensions also as 32 × 32. The GPU has 96 KB of shared memory per multiprocessor (SM). If each SM can support a maximum of 2048 threads, calculate the percentage of shared memory utilization per SM.

A. 15.57%
B. 16.67%
C. 20.0%
D. 18.33%

Answer:B

Solution:

Tile size per block: 32×32

Threads per block=32×32=1024

Each block requires two shared memory tiles for matrices A and B:

Matrix A tile: 32×32×4=4096 bytes

Matrix B tile: 32×32×4=4096 bytes

Total shared memory per block=4096+4096=8192 bytes=8 KB

The maximum number of threads per SM is 2048, so the number of thread blocks that can fit = 2048 / 1024

=2 blocks

Shared memory used by 2 blocks=2×8 KB=16 KB

Memory Usage Percentage=(16 KB/96 KB)×100=16.67%

Question 4

Consider an array of size N is stored sequentially in global memory and each thread swaps one element from the beginning with one element from the end of the array, the following kernel is launched with 256 threads per block and 4 blocks. In total, how many warp READ accesses (i.e., global memory read accesses made by all warps) will occur when performing the array reversal given that each warp consists of 32 threads ? (Assume no effect of Cache subsystem and N=1024)

```
__global__ void reverseArray(int *arr, int N) {
        int idx = threadIdx.x + blockIdx.x * blockDim.x;
        if (idx < N / 2) {
                int temp = arr[idx];
                arr[idx] = arr[N - 1 - idx];
               arr[N - 1 - idx] = temp;
        }
}
```

A. 16 warp accesses
B. 32 warp accesses
C. 64 warp accesses
D. 128 warp accesses

Answer:B

Solution:

Each warp accesses 64 elements during the reversal operation (32 from the start and 32 from the end of the array). As there are 1024 elements in the array and each warp processes 64 elements, the number of warps required to reverse the array is:

1024/64=16 warps
Each warp performs 4 memory accesses:
2 read accesses (1 from the beginning and 1 from the end).
So, the total number of warp accesses is the number of warps multiplied by the number of accesses per warp:
16 warps×2 accesses per warp=32 warp accesses.

Question 5
Consider the following kernel code snippet.

```
#define BDIMX 32
#define BDIMY 32
__global__ void setRowReadCol(int *out)
{
__shared__ int tile[BDIMY][BDIMX];
unsigned int row_idx = threadIdx.y * blockDim.x + threadIdx.x;
unsigned int col_idx = threadIdx.x * blockDim.y + threadIdx.y;
tile[row_idx] = row_idx;
__syncthreads();
out[row_idx] = tile[col_idx];
}
```

The kernel is executed on a GPU architecture where the number of shared memory banks is 16 and the bank width is 4 bytes. The kernel is launched with the following configuration.
dim3 block (BDIMX, BDIMY);
dim3 grid (1,1);
Assuming the size of an integer is 4 bytes and wrap size 16: match and pair the following.

i. Number of shared loads per warp request	a. 1
ii. Number of shared stores per warp request	b. 16
iii. Number of global loads per warp request	c. 32
iv. Number of global stores per warp request	d. 0

A. i → b, ii->b, iii->a, iv->a
B. i → b, ii->a, iii->d, iv->a
C. i → c, ii->c, iii->a, iv->a
D. i → a, ii->a, iii->b, iv->b

Answer: B

Question 6

Consider the following code snippet executing on a GPU architecture where the number of shared memory banks is 32 and the bank width is 4 bytes.

```
#define SZ 32
__global__ void setRowReadCol(float *out) {
__shared__ int tile[SZ][SZ];
unsigned int gid = threadIdx.x * blockDim.y + threadIdx.y;
tile[threadIdx.x][threadIdx.y] = gid;
__syncthreads();
int tidx = _______ ;
int tidy = ________;
out[gid] = tile[tidx][tidy];
}
```

The kernel is launched with parameters <<<(32,32),(32,32)>>>. Consider the last shared load operation. The accesses for this operation depends on the values of tidx and tidy. State whether the following statement is true or false.

"For tidx=threadIdx.y, tidy=threadIdx.x, shared load transactions are bank conflict free"

A. True
B. False

Answer: A

Solution:

Since, the size of the shared tile is 32x32 and the number of banks is 32, each column of the tile gets mapped to a single bank. Now threads are packed into warps as follows.

warp 0: (0,0) ,(0,1),.....,(0,31)

warp 1: (1,0) ,(1,1),.....,(1,31)

and so on.

The first element of each pair is the threadIdx.y value while the second element is the threadIdx.x value. Thus for any warp, the values of the threadIdx.x change for a fixed value of threadIdx.y. Given this, shared memory access expressions of the form tile[threadIdx.y][threadIdx.x] would be bank conflict free, since threads in any warp access elements from different columns.

Question 7

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 8
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 9
For an effective Thread-level Coarsening across z axis for a 3D kernel with launch parameter <<<(16,16,16),(32,16,16)>>>, coarsening factor 2 and target platform with warp size 16, the minimum and maximum bound for stride length are
Options:
A. 8, 4
B. 4, 8
C. 1, 4
D. not feasible
Answer: D

Solution:

`Max stride length ≤ (Threads per block in z/ Coarsening factor)=16/2=8`

`Min stride length ≥ wrap size(16)`

Here, the maximum stride length (8) is less than the minimum required stride length

Question 10:

Match column 1 with the correct option in column 2

a) Global Memory	i) Shared only within a single SM
b) Shared Memory	ii) Shared across all SMs
c) Local Memory	iii) Private to a wrap
d) Constant Memory	iv) Private to a thread
	v) Shared across group of SMs but not among all SMs

Options:

A. a-ii,b-i,c-iv,d-iii
B. a-ii,b-i,c-iv,d-ii
C. a-ii,b-ii,c-iv,d-v
D. a-ii,b-i,c-v,d-iii

Answer: B

Solution:

The memory shared among all SMs is Global Memory and Constant Memory.Shared memory is shared among all the threads within a single SM while local memory is private to a thread.