GPU Architectures
and Programming
Assignment- Week 8
TYPE OF QUESTION: MCQ/MSQ

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

MCQ/MSQ Question

Common data set for question 1-2

For the given GPU architecture-

- Streaming Multiprocessors (SM): 20
- Max. active threads per SM: 2048
- Max. thread blocks per SM: 32
- Registers per SM: 64K
- Max Shared Memory per SM: 96KB
- Max. Shared Memory per block: 48KB

The given kernel finds the maximum of a given data set (without any coarsening).

```
__global__ void max ( int * g_idata , int * g_odata ,
unsigned int n){
  __shared__ int sdata [1024];
  unsigned int tid = threadIdx .x;
  unsigned int i = blockIdx .x * ( blockDim .x * 2) +
threadIdx .x;
  sdata [ tid ] = g_idata [i] + g_idata [i+ blockDim .x];
  __syncthreads ();
  for ( unsigned int s= blockDim .x /2; s>0; s > >=1) {
     if ( tid < s)
         sdata [ tid ] =max(sdata [ tid ], sdata [ tid +
s]);
     __syncthreads ();
```

```
    }
    if ( tid == 0)
      g_odata [ blockIdx .x] = sdata [0];
}
```

Question 1:

What is the optimal coarsening factor for thread-level coarsening coarsening for the above program.

a) 4
b) 8
c) 12
d) 16

Answer: b

Question 2:

What is the optimal coarsening factor for block-level coarsening for the above program.

a) 4

b) 16

c) 12

d) 8

Answer: d

Solution of 1 and 2:

Thread-level Coarsening:

Let x be the coarsening factor.
Block size = 1024/x
Active blocks per SM = 2048/(1024/x)=2*x
For thread-level coarsening, shared memory requirement per block is same.
Shared memory requirement per block = 1024*4=4096 byte
Shared memory requirement per SM = (2*x) * 4096 byte
Max allowable shared memory per SM = 96KB.

=> (2*x) * 4096 byte = 96 * 1024 byte
=> x = 12
=>x = 8 ($2^3 < 12 < 2^4$)

Block-level Coarsening:
Let x be the coarsening factor.
Block size = 1024
Active blocks per SM = 2048/1024=2
Shared memory requirement per block = 1024*4*x byte
Shared memory requirement per SM = 2 * (4096*x) byte
Max allowable shared memory per SM = 96KB.
=> 2 * (4096*x) byte = 96 * 1024 byte
=> x = 12

=>x = 8 ($2^3 < 12 < 2^4$)

Question 3

For an effective Thread-level Coarsening across x axis for a 2D kernel with launch parameter

<<<(16,16,1),(64,16,1)>>>, coarsening factor 4 and target platform with warp size 8, the

minimum and maximum bound for stride length are

Options:

a) 1, 4

b) 8, 8

c) 4, 8

d) 8, 16

Ans: d

Solution:

Max stride length ≤ (Number of thread per block in the dimension where coarsening is

applied)/ Coarsening Factor (64/4=16)

Minimum stride length ≥ Warp size to ensure memory coalescing (8)

Question 4:

For an effective Thread-level Coarsening across z axis for a 3D kernel with launch parameter

<<<(16,16,16),(32,16,8)>>>, coarsening factor 2 and target platform with warp size 8, the

minimum and maximum bound for stride length are

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)=8/2=4

Min stride length≥wrap size(8)

Here, the maximum stride length (4) is less than the minimum required stride length (8). So thread-level coarsening along the z-axis is not feasible.

Question 5:

What does "cache pressure" refer to in the context of memory access?

a) Overloading the processor with instructions
b) Overworking the memory cache with excessive or wasteful memory access
c) Using GPU memory inefficiently for computation
d) Reducing memory bandwidth usage

Answer: b

Question 6:

What type of memory access pattern minimizes cache pressure on a GPU?
a) Frequent re-use of the same cache line
b) Random memory accesses
c) Streaming data access with no data re-use
d) Accessing non-coalesced global memory

Answer: C

Question 7:

Which reduction kernel method is likely to experience divergent branching due to modulo arithmetic?
a) Reduce1
b) Reduce2
c) Reduce3
d) Reduce5

Answer: a) Reduce1

Solution:

Refer to the slides

Question 8:

Consider the kernels specified with modified calling parameters:

`kernel1<<< 512, 512 >>>, kernel2<<< 256, 1024 >>>`

What will be kernel launch parameters for a inner thread fused version of kernels 1 and 2?

a) fused_kernel<<<512,512 >>>
b) fused_kernel<<<512,1024 >>>
c) fused_kernel<<<1024,1024 >>>
d) None of the above

Answer: b

Solution:

S_th = max(S_th1,S_th2) = max(512, 1024) = 1024

S_bk = max(S_bk1,S_bk2) = max(512, 256) = 512

Question 9:

Consider the kernels specified with modified calling parameters:

```
kernel1<<< 1024, 512 >>>, kernel2<<< 512, 1024
>>>
```

What will be kernel launch parameters for a inner block fused version of kernels 1 and 2?

a) fused_kernel<<<512,512 >>>
b) fused_kernel<<<512,1024 >>>
c) fused_kernel<<<1024,1024 >>> (correct)
d) fused_kernel<<<1024, 1536>>>

Answer: d

Solution:
S_th = sum(S_th1,S_th2) = 1024+512= 1536
S_bk = max(S_bk1,S_bk2) = max(512, 1024) = 1024

Question 10:

Consider the kernels specified with modified calling parameters:

```
kernel1<<< 512, 512 >>>, kernel2<<< 512, 1024 >>>
```

What will be kernel launch parameters for a inter block fused version of the kernels 1 and 2?

a) fused_kernel<<<512,1024 >>>
b) fused_kernel<<<1024,512 >>>
c) fused_kernel<<<1024,1024 >>> (correct)
d) fused_kernel<<<1024,2048 >>>

Answer: c
Solution:

$S_{th} = \max(S_{th,1}, S_{th,2}) = \max(512,1024) = 1024$
$S_{bk} = \mathrm{sum}(S_{bk,1}, S_{bk,2}) = (512 + 512) = 1024$

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