ST1 PROGRAMMING QUESTIONS

QUESTION 1

Let us consider a 2D Neighborhood Operation which is a spatial transformation operation that takes as input a 2D integer array A of size N x N and produces a 2D integer output array B of size MxM where M<N. This is obtained by considering a 2D window W of size rs x cs and sliding the window in strides of rs across rows and strides of cs across columns. The pooling operation in context takes the sum of the elements in A for each such overlap and produces one element of array B. The window starts by covering the top left block of size rs x cs in A and is moved in strides along rows , keeping in mind not to go beyond the boundaries of A. Assume that N is divisible by rs and cs. Consider the following incomplete code-snippet.

```
__global__ void 2DPool(int* A, int* B, int N, int M, int rs, int cs)
{
     int tidx = blockIdx.x*blockDim.x+threadIdx.x;
     int tidy = blockIdx.y*blockDim.y+threadIdx.y;
     int i = __(a)__;
     int j = __(b)__;
     int sum = 0;
     for(int x=i;x<i+rs;x++)
        for(int y=j;y<j+cs;y++)
            sum+=A[__(c)__];
     B[__(d)__] = sum;
}
```

The kernel is launched with a 2D grid of 2D blocks. The variables (tidx,tidy) represent a (row,column) position in the output matrix B. Choose the correct option for matching and pairing the two columns in the following table.

a)	i) tidx*rs
b)	ii) tidy*cs
c)	iii) tidx*M+tidy
d)	iv) x*N+y
	v) tidx*N+tidy

	vi) x*M+y

Options:

A. a->ii, b->iii c->i,d->iv
B. a->i, b->iv c->ii,d->iii
C. a->ii, b->i c->iii,d->iv
D. a->i, b->ii c->iv,d->iii
E. None of the above

Answer: D

QUESTION 2

Consider a 3D Array A of MxNxP integers. There exists a sum kernel called add(A) which returns the sum of all elements in A. Consider another kernel snippet, the code for which is given below.

```
__global__ void initCube(int* A, int M,int N, int P)
{
       int tidx = blockIdx.x*blockDim.x+threadIdx.x;
       int tidy = blockIdx.y*blockDim.y+threadIdx.y;
       int tidz = blockIdx.z*blockDim.z+threadIdx.z;

      int index = (M*N)*tidx + N*tidy + tidz;
      A[index]=blockIdx.x ^ blockIdx.y ^ blockIdx.z;
}
```

Given M = 16, N=16 and P=16, what is the output of add(A) after A is processed by the above kernel? The launch parameters for initCube are <<<(2,2,2),(8,8,8)>>>.

A.512
B.256
C.192
D.2048
E.None of the above

Answer: D

QUESTION 3:

Let us consider a 1D Neighborhood Operation, say **reduce()** which is a spatial transformation that takes as input a 1D array **A** of size **N** and produces a 1D output array **B**. This is obtained by considering a 1D window of size **k** and sliding it along A in strides of **k.** Therefore, this operation repeats a total of N/k times. In each instance, the operation takes the average of the elements in A for the window of length k and produces one element of array **B.** The total number of elements computed and stored in B is therefore N/k. Assume N is divisible by k.

For example, consider A=[1 2 3 4 5 6 7 8] where N = 8 and k=2. The output array B is therefore of size 8/2=4 and is [1.5 3.5 5.5 7.5]. The first entry of B is B[0] = avg (A[0], A[1]) = 1.5. Similarly for the rest. Now consider the following incomplete code-snippet where the reduction operation is computing maximum.

```
__global__ void reduce(int* A, int* B,  int N, int k)
{
   int tid = (blockIdx.y * gridDim.x + blockIdx.x) * blockDim.x +
threadIdx.x;
     int bound = _____(a)______ ;
     int blockSize = ____(b)________;
    for(int s=1;s<blockSize;s*=2)
    {
       if(tid%2*s == 0 && tid+s<bound)
          A[tid] = max(A[tid],A[tid+s]) ;
      --syncthreads() ;
     }
     if(threadIdx.x == 0)
          B[___(c)______ ] = A[____(d)_______ ];
}
```

The kernel is launched with a 2D grid of 1D blocks where each block is responsible for reducing k consecutive elements to one element. In other words, the window of size k moving in strides of k has k= blockDim.x.

Choose the correct option for matching and pairing the two columns in the following table.

a)	i) blockDim.x
b)	ii) (1+(blockIdx.y * gridDim.x + blockIdx.x)) * blockDim.x
c)	iii) blockIdx.y * gridDim.x + blockIdx.x
d)	iv) (blockIdx.y * gridDim.x + blockIdx.x) * blockDim.x

Options:

A. a->i, b-> ii, c-> iv, d-> iii

B. a->iii, b-> iv, c-> i, d-> ii

C. a->ii, b-> i, c-> iii, d-> iv

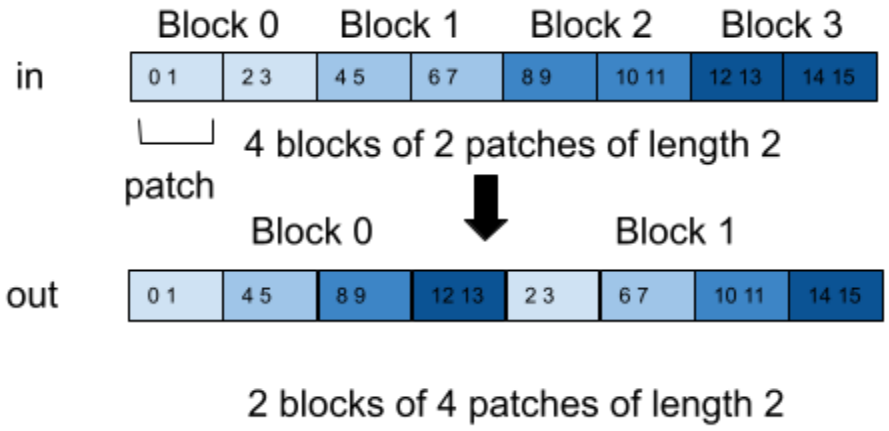
D. a->iv, b-> iii, c-> ii, d-> i

E. None of the above

Answer: c

QUESTION 4

Consider a 1D kernel reorder which operates on the buffer **in** which is of dimension N x B x P, reorders the elements and produces the buffer **out** which is of dimension B x N x P. The reorder operation sets out[b][n][p]=in[n][b][p] as illustrated in the figure below.



The corresponding incomplete kernel code snippet is depicted below.

```
__global__ void reorder(float *in, float *out, int B, int N, int P)
{
      int id = blockIdx.x*blockDim.x+threadIdx.x;
      int n =  _____(a)______ ;
      int patch= _____(b)______ ;
      int b=patch/P;
      int p=patch%P;
      int input_ptr= _____(c)______ ;
      int output_ptr= _____(d)______ ;
      out[output_ptr]=in[input_ptr];


}
```

a)	i) `id%(B*P)`
b)	ii) `id/(B*P)`
c)	iii) `p*B*N+n*N+p`
d)	iv) `n*N*P+b*B+p`
	v) `b*N*P+n*P+p`
	vi) `n*B*P+b*P+p`

Answer: A. a - ii, b-i, c-vi, d-v
B. a - i, b-ii, c-v, d-iv
C. a - ii, b-i, c-iv, d-vi
D. a - i, b-ii, c-vi, d-iv

Answer A

Solution

```
__global__ void reorder(float *in, float *out, int B, int N, int P)
{
      int id = blockIdx.x*blockDim.x+threadIdx.x;
      int n = id/(B*P);
```

```
        int patch=id%(B*P);
        int b=patch/P;
        int p=patch%P;
        int input_ptr=n*B*P+b*P+p;
        int output_ptr=b*N*P+n*P+p;
        out[output_ptr]=in[input_ptr];


}
```

QUESTION 5

A 2D image can be represented by a 2D array of pixel intensity values where each value is a floating point number. Let us consider a smoothening operation which takes an image **img_src** of dimensions **NxN** and creates a new image **img_dst** of dimensions NxN**.** Any pixel value in **img_dst[i][j]** is computed as the average of the immediate neighbours of **img_src[i[j]**

For example, let us consider the following 4x4 img_src

1.00	2.00	3.00	4.00
5.00	6.00	7.00	8.00
9.00	10.00	11.00	12.00
13.00	14.00	15.00	16.00

After applying the smoothening operation, we obtain the following 4x4 img_dst

3.50	4.00	5.00	5.50
5.50	6.00	7.00	7.50
9.50	10.00	11.00	11.50
11.50	12.00	13.00	13.50

One can note that img_dst[1][1] = (img_src[0][0] + img_src[1][0] + img_src[2][0] + img_src[1][0] + img_src[1][1] + img_src[2][1] + img_src[0][1] + img_src[1][2] + img_src[2][2]) / 9.

For elements residing at the boundaries, consider taking the average of the immediate neighbours which are present in img_src. Thus, img_dst[0][0] = (img_src[0][0] + img_src[1][0] + img_src[1][0] + img_src[1][1])/4.

As you can understand, this operation is highly parallelizable. We construct a CUDA kernel called smoothen that operates on NxN images and uses a 1D Grid of 1D Blocks <<<(N,1,1),

(N,1,1)>>>. Each thread is responsible for computing the required average of one position in the destination image img_dst.

Consider the following incomplete version of the CUDA kernel.

```
__global__ void smoothen(float *img_src, float *img_dst, int N)
{
        int i,j,num_neighbours,u,v,ne_i,ne_j;
        i = ______(a)______;
        j = ______(b)_______;


        num_neighbours=0;
        for(u=-1;u<=1;u++)
        {
           for(v=-1;v<=1;v++)
           {
                ne_i=i+u;
                ne_j=j+v;
                if(ne_i>=0 && ne_i<N)
                  if(ne_j>=0 && ne_j<N)
                  {
                        img_dst[___(c)___]+=img_src[___(d)____];
                        num_neighbours++;
                   }

               }
        }
        img_dst[i*N+j]/=num_neighbours;
}
```

a)	i) (blockIdx.x*blockDim.x+threadIdx.x)%N;
b)	ii)(blockIdx.x*blockDim.x+threadIdx.x)/N
c)	iii) i*N+j
d)	iv) ne_i*N+ne_j

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

Answer A

Solution

```
__global__ void smoothen(float *img_src, float *img_dst, int N)
{
        int i,j,num_neighbours,u,v,ne_i,ne_j;
        i = (blockIdx.x*blockDim.x+threadIdx.x)/N;
        j = (blockIdx.x*blockDim.x+threadIdx.x)%N;


        num_neighbours=0;
        for(u=-1;u<=1;u++)
        {
           for(v=-1;v<=1;v++)
           {
                ne_i=i+u;
                ne_j=j+v;
                if(ne_i>=0 && ne_i<N)
                  if(ne_j>=0 && ne_j<N)
                  {
                        img_dst[i*N+j]+=img_src[ne_i*N+ne_j];
                        num_neighbours++;
                   }

               }
        }
        img_dst[i*N+j]/=num_neighbours;
}
```

ST1 MULTIDIMENSIONAL MAPPING QUESTIONS (NUMERICAL)

QUESTION 1:

Consider a 1D CUDA kernel that computes on a 1D array A of size 2048 with launch parameters <<<(32),(64)>>> . Each CUDA thread operates only on one data point. On what data point will the thread with threadIdx.x=20, blockIdx.x=20 operate on. Assume that thread ids along the x-dimension represent rows and thread ids along the y-dimension represent columns. Choose the correct option.

A. A[200]
B. A[40]
C. A[440]
D. A[660]
E. None of the above

Solution: D

QUESTION 2:

Consider a 2D CUDA kernel that computes on a 2D matrix M of size 2048x2048 with launch parameters <<<(32,32),(64,64)>>> . Each CUDA thread operates only on one data point. On what data point will the thread with the threadIdx.x=0, threadIdx.y=10, blockIdx.x=10 and =blockIdx.y=10 operate on. Assume that thread ids along the x-dimension represent rows and thread ids along the y-dimension represent columns. Choose the correct option.

A. M[10][10]
B. M[32][32]
C. M[320][320]
D. M[100][100]
E. None of the above

Answer: C

QUESTION 3:

Consider a kernel processing a 2D matrix of dimensions 1024x1024 where each thread is assigned to perform an operation on a single element of the matrix. The kernel is launched with

the following grid and block configurations: <a,32,2> blocks of <32,b,2>. For a hypothetical GPU architecture where the maximum number of threads in a block is 512, what are the values of a and b? Select the correct option from below.

A. a= 16, b=8
B. a=4, b =32
C. a=32, b=4
D. a=8,b=16
E. None of these

Correct Answer: C

QUESTION 4:

Consider a kernel processing a 1D array of 4096 elements where each thread is assigned to perform an operation on a single element of the array. The kernel is launched with the following grid and block configurations: <16,a,2> blocks of <b,2,4>. Assuming a hypothetical GPU where the maximum number of threads in a block is 256, what are the possible values of a and b. Please select the correct option from below.

A. a=4 b=4
B. a= 1, b= 16
C. a=8,b=2
D. All values of a and b given in options
E. None of the values of a and b given in options

Correct Answer: D

QUESTION 5:

Consider a kernel processing a 1D array of 8192 elements where each thread is assigned to perform an operation on a single element of the array. The kernel is launched with the following grid and block configurations: <16,a,2> blocks of <b,2,4> Assuming a GPU where the maximum number of threads in a block is 1024, what are the possible values of a and b. Please select the correct option from below.

A.a=16,b=2

B.a= 2 b= 16

C.a=8 b=4

D.All of the above

Correct Answer: D

ST1 ARCHITECTURE QUESTIONS (SHORT)

QUESTION 1:

Two processors A and B have clock frequencies of 700 Mhz and 900 Mhz respectively. Suppose A can execute an instruction with an average of 3 cycles and B can execute with an average of 5 cycles. For the execution of the same instruction which processor is faster?

a) A

b) B

c) Both take the same time

d) Insufficient information

Correct Answer: a

QUESTION 2:

When processing an array using a looping operation, items at nearby addresses are referenced from time to time. This is called _____ locality. Fill in the blank.

Select one:

A. Temporal

B. Spatial

C. None of the options

D. Regional

Answer: B

QUESTION 3:

Which of the following statements is true?

1. Cache memory provides the fastest access times in the memory hierarchy.

2. Registers provide the slowest access times in the memory hierarchy.

3. In write-back policy, both cache and main-memory is updated when a cache miss occurs.

4. Increasing cache associativity decreases miss rate up to a point, but increases hit times.

Select one:

A. 2

B. 4

C. 1

D. 3

Answer: B

QUESTION 4:

The computer architecture aimed at reducing the time of execution of instructions is _____

a) CISC

b) RISC

c) ISA

d) GPU

Correct Answer: b

QUESTION 5:

The contention for the usage of a hardware device is called _____and the situation wherein the data of operands are not available is called _____

a) Structural hazard, Data hazard

b) Data hazard, Structural hazard

c) Deadlock, Stock

d) Stock, Deadlock

Correct Answer: a

QUESTION 6:

Pipelining increases CPU instruction _____ and reduces processor's _____.

a) Size, Efficiency

b) Throughput, Cycle time

c) Cycle rate, Cycle time

d) Latency, Efficiency

Correct Answer: b

QUESTION 7:

One of the possible ways to deal with data hazards is _____.

a) Reducing CPI

b) Using branch prediction

c) Increasing number of processors

d) Adding forwarding hardware

Correct Answer: d

QUESTION 8:

Branch decision is inferred in_____ stage.

a) Fetch

b) Decode

c) Mem

d) Write back

Correct Answer: c

QUESTION 9:

Which of the following statements is correct.

A. Compared to CPU, GPUs have larger register file, smaller L1/L2 cache with higher bandwidth
B. Compared to CPU, GPUs have smaller register file, smaller L1/L2 cache with lower bandwidth
C. L1 and L2 cache + Shared memory is private to SMs
D. Cache + Constant memory is unified for all SMs

QUESTION 10

If a data item is referenced by a program in execution, it will again be referenced soon. This is called _____ locality.

A. Temporal
B. Spatial
C. Regional
D. None of the above
E. All of the above

Answer: A