GPUs and Heterogeneous Systems – A.Y. 2023-24

Scuola di Ingegneria Industriale e dell'Informazione
Prof. Antonio Miele

POLITECNICO
MILANO 1863

July 19, 2024 - **FIRST PART OF THE EXAM**

Surname:	Name:	Personal Code:

Question	**1**	**2**	**3**	**4**	**5**	**OVERALL**
Max score	**3**	**3**	**3**	**3**	**3**	**15**
Score						

Instructions:

- This first part of the exam is "closed book". The students are not allowed to consult any course material and notes.
- No extra devices (e.g., phones, iPad) are allowed. Please, shut down and store any electronic device.
- Students are not allowed to communicate with any other ones.
- Students can write in pen or pencil, any color, but avoid writing in red.
- Any violation of the above rules will lead to the invalidation of the test.
- **Duration: 30 minutes**

Question 1

Briefly explain what the graphics pipeline is.

The graphics pipeline is the framework comprising all compute steps required for 3D rendering, i.e., for transforming a model of a 3D scene into a 2D representation on a screen.

Question 2

Complete the following OpenACC pragmas to optimize data transfer between the host and the device. In particular, 1) specify between brackets () the appropriate list of arrays (index range for each array is always `[0:N]`), and 2) delete unnecessary clauses. Motivate your answer.

```
void foo(int *A, int *B){
  int C[N];
#pragma acc data copy(A[0:N]) copyin(     ) copyout(     ) create(C[0:N])
{
  #pragma acc parallel loop copy(     ) copyin(B[0:N]) copyout(     ) create(     )
  for(int i = 0; i < N; i++)
    C[i] = A[i] + B[i];
  #pragma acc parallel loop copy(     ) copyin(     ) copyout(     ) create(     )
  for(int i = 0; i < N; i++)
    A[i] = C[i] * A[i];
}
}
```

A is taken as input in both the loops, and it represents the final result.
B is taken in input only in the first loop.
C is used to save the intermediate results of the first loop, then used in the second loop.

Question 3

Simulate the following parallel reduction kernel (based on the algorithm optimizing memory access efficiency) and show the `data` array's content at each loop iteration. Consider a grid with 2 blocks, each block with 4 threads; the initial content of the `data` array is: `[0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]`

```
#define STRIDE_FACTOR 2

__global__ void reduce(double* data) {
  int i = threadIdx.x;
```

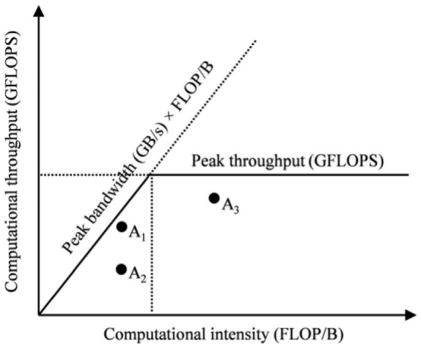
```
  int base_i = blockDim.x * blockIdx.x * STRIDE_FACTOR;
  for (int stride = blockDim.x; stride >= 1; stride /= STRIDE_FACTOR) {
    if (i < stride) {
      data[base_i + i] += data[base_i + i + stride];
    }
    __syncthreads();
  }
}
```

```
Input:       [ 0, 1, 2, 3, 0, 1, 2, 3,  0, 1, 2, 3, 0, 1, 2, 3]
Iteration 1: [ 0, 2, 4, 6, 0, 1, 2, 3,  0, 2, 4, 6, 0, 1, 2, 3]
Iteration 2: [ 4, 8, 4, 6, 0, 1, 2, 3,  4, 8, 4, 6, 0, 1, 2, 3]
Iteration 2: [12, 8, 4, 6, 0, 1, 2, 3, 12, 8, 4, 6, 0, 1, 2, 3]
```

Question 4

Draw a basic example of the roofline model. Then, add to the plot the following 3 points representing:

A. A memory-bound application efficiently using resources
B. A memory-bound application not efficiently using resources
C. A compute-bound application efficiently using resources



A1: A memory-bound application efficiently using resources
A2: A memory-bound application not efficiently using resources
A3: A compute-bound application efficiently using resources

Question 5

Briefly describe the CUDA memory model (organize the answer in a table); for each component, specify the name, which data is stored, type of access (read/write or read-only), and scope.

	Type of data	Type of access	Scope
Registers	Automatic variables (no array or struct)	Device: read/write	1 Thread
Local Memory	"large" automatic variable	Device: read/write	1 Thread
Global Memory	All data exchanged with the host and among all application's kernels	Device: read/write Host: read/write	All threads + Host
Shared Memory	Data exchanged within a block	Device: read/write	1 Block
Constant Memory	Constants used by all kernels	Device: read Host: read/write	All threads + Host
Texture Memory	2D constant data used by all kernels	Device: read Host: read/write	All threads + Host