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

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

POLITECNICO
MILANO 1863

January 28, 2025 - **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 a tensor core is, which operation it performs, and why it has been added to the recent GPU architectures.

Question 2
Explain the streaming multiprocessor occupancy index: formula, why it is relevant, and what it is used for.

Question 3
Let's assume to run the following kernel on a Maxwell (or more recent) architecture and to size the grid with a single block of 32 threads; which is the efficiency of global load and store operations of the following CUDA kernel? Motivate the answer.

```
__global__ void vsumKernel(char* a, char* b){
  int i = blockIdx.x * blockDim.x + threadIdx.x;
  b[(i+2)%blockDim.x] = a[(i*2)%blockDim.x];
}
```

Question 4
For each of the two following device characterizations, evaluate (and motivate) whether the kernel reported below is compute-bound or memory-bound:

1. Peak FLOPS=100 GFLOPS, peak memory bandwidth=250 GB/second
2. Peak FLOPS=150 GFLOPS, peak memory bandwidth=100 GB/second

```
__global__ void foo(float *in1, float *in2, float *in3, float *output){
  const int i = blockIdx.x*blockDim.x + threadIdx.x;
  const float a = in1[i];
  const float b = in2[i];
  const float c = in3[i];
  output[i] = ((a-b-c)/a + (c-a-b)/c + (b-a-c)/c) * 3;
}
```

Question 5
Is it possible to parallelize the following snippet of code using OpenACC pragmas? Motivate the answer.

```
int sum=0;
for(int i = 0; i < N; i++)
  sum += A[i];
```

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

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

POLITECNICO
MILANO 1863

January 28, 2025 - **SECOND PART OF THE EXAM**

Surname:	Name:	Personal Code:

Question	**1**	**2**	**3**	**OVERALL**
Max score	**5**	**5**	**6**	**16**
Score				

Instructions:

- This second part of the exam is "open book". The students are allowed to use any material and notes.
- The students are allowed to use the laptop and the tablet. No extra devices (e.g., phones) are allowed. Please, shut down and store not allowed electronic devices.
- Students are not allowed to communicate with any other one or use Internet.
- Students can write in pen or pencil, any color, but avoid writing in red.
- Students can also use the laptop to code the test solution. In this case, please pay attention to the instructor's instructions to submit the test solution.
- Any violation of the above rules will lead to the invalidation of the test.
- **Duration: 1 hour and 15 minutes**

The source code can be downloaded from the course page on WeBeep

Question 1
Implement a basic CUDA kernel function to accelerate the compute-intensive function in the following C program.

Question 2
Modify the main function to execute the CUDA kernel function defined in the former question. Set the block size to 32 (for each used dimension).

Question 3
Implement a new CUDA kernel function to accelerate the compute-intensive function in the following C program by using dynamic parallelism. Specify any change (possibly) applied to the main function.

```
/*
* The following program elaborates a 2D matrix of integers where each row contains a
* variable number of elements. In particular, the kernel function receives in input
* the matrix and sum to each element a value obtained by multiplying the row number
* by the number of elements in the row.
* For instance, if we consider the following matrix with 4 rows:
* 1 2 3 4
* 1 2
* 1 1 1 1 1
* 2
* The kernel function will compute a new matrix as follows:
* 1 2 3 4
* 3 4
* 11 11 11 11 11
* 5
*
* The described matrix is represented in the program as follows:
* - An integer array A contains all the elements of the matrix in a linearized way
* - An integer variable NUMOFELEMS containing the overall number of elements in the matrix
* - An integer variable ROWS contains the number of rows in the matrix
* - An integer array COLOFFSETS contains the indexes in A where each matrix row starts
* - An integer array COLS contains the length of each row of the matrix
*
* Thus, the matrix above is modeled in the program as:
* A = [1 2 3 4 1 2 1 1 1 1 1 2]
* NUMOFELEMS = 12
* ROWS = 4
* COLOFFSETS = [0, 4, 6, 11]
* COLS = [4, 2, 5, 1]
*/

#include<stdio.h>
#include<stdlib.h>

void printM(int* a, int rows, int* colOffsets, int *cols);

//kernel functions: sum to each element of the matrix a coefficient obtained
//by multiplying the row number by the row index
void elaborateMatrix(int* a, int rows, int* colOffsets, int *cols, int* b){
  int i, j, coeff;
  for(i=0; i<rows; i++){
    coeff = i*cols[i];
    for(j=0; j<cols[i]; j++)
      *(b + colOffsets[i] + j) = *(a + colOffsets[i] + j) + coeff;
  }
}

int main(int argc, char *argv[]){
  int *a, *b; //input and output matrices
  int rows; //number of rows
  int *cols; //contains the number of columns per each row
  int *colOffsets; //contains the indexes in the data array where each matrix row starts
  int numOfElems; //overall number of elements
  int i, j;

  //generate the matrix

  //call the kernel function
  elaborateMatrix(a, rows, colOffsets, cols, b);

  //print results

  //release memory
  free(a);
  free(b);
  free(cols);
  free(colOffsets);

  return 0;
}
```