Intensive Computation

Prof. A. Massini

June 18, 2019

Part B

Student's Name

_____

***Matricola* number**

_____

Exercise 1 (4 points)	
Exercise 2 (4 points)	
Exercise 3 (4 points)	
Question 1 (3 points)	
Question 2 (3 points)	
Exercise 4 (3 points)	
Exercise 5 (3 points)	
Exercise 6 (3 points)	
Exercise 7 (2 points)	
Exercise 8 (3 points)	
Total (32 points)	

Exercise 1 (4 points) - GPU & CUDA

1) You need to write a kernel that operates on a matrix of size **5760x1600**. You would like to assign one thread to each matrix element. You would like your thread blocks to use the maximum number of threads per block possible on your device, having compute capability 3.0.

a) How would you select the dimensions of a **2D grid** and **2D blocks** for your kernel?

b) What is the best choice for grid and block dimensions with respect to the number of idle threads?

Technical specifications	Compute capability (version)									
	1.0	**1.1**	**1.2**	**1.3**	**2.x**	**3.0**	**3.5**	**3.7**	**5.0**	**5.2**
Maximum dimensionality of grid of thread blocks	2				3					
Maximum x-dimension of a grid of thread blocks	65535					2^{31}-1				
Maximum y-, or z-dimension of a grid of thread blocks	65535									
Maximum dimensionality of thread block	3									
Maximum x- or y-dimension of a block	512				1024					
Maximum z-dimension of a block	64									
Maximum number of threads per block	512				1024					
Warp size	32									
Maximum number of resident blocks per multiprocessor	8					16			32	
Maximum number of resident warps per multiprocessor	24		32		48	64				
Maximum number of resident threads per multiprocessor	768		1024		1536	2048				
Technical specifications	**1.0**	**1.1**	**1.2**	**1.3**	**2.x**	**3.0**	**3.5**	**3.7**	**5.0**	**5.2**
	Compute capability (version)									

2) Select the expression for mapping the thread indices to data, if you use each thread to calculate one output element of a matrix addition using **2D grid** and **2D blocks**:

a) index = blockIdx.x * blockDim.x * blockDim.y + threadIdx.y * blockDim.x + threadIdx.x;

b) index = (blockIdx.x + blockIdx.y * gridDim.x) * (blockDim.x * blockDim.y) + (threadIdx.y * blockDim.x) + threadIdx.x;

c) index = (blockIdx.y * gridDim.x + blockIdx.x;)* blockDim.x + threadIdx.x;

Exercise 2 (4 points) – Interconnection Networks – CLOS

Design a Clos network of size 60 x 60, using modules 4x4. In the first stage, only 4 inputs per module are allowed.

Consider both cases, strictly non-blocking and rearrangeable network.

Compare the cost of the crossbar 60 x 60 and the Clos network, strictly non-blocking and rearrangeable, designed in the previous point, giving the gain in both cases.

Exercise 3 (4 points) – Interconnection networks

Draw a **Butterfly network** of size 8 and briefly explain how the self-routing algorithm works. Show the connection between input 1 and output 5 and between input 0 and output 2.

Draw a **Cube network** of size 8 and briefly explain how the routing algorithm works. Show the connection between input 1 and output 5 and between input 0 and output 2.

Question 1 (3 points)

Briefly explain the main features of multistage interconnection networks.

Question 2 (3 points)

Briefly describe class of SIMD architectures, also describing the main features of the CRAY-1

Exercises 4 (3 points) Amdhal Law

The following measurements are recorded with respect to the different instruction classes for the instruction set running a given set of benchmark programs:

Instruction Type	Instruction Count (millions)	Cycles per Instruction
Arithmetic and logic	8	7
Load and store	6	3
Branch	6	6
Others	10	5

Assume that "*Arithmetic and logic*" instructions can be modified so that they take 5 cycle per instruction. Compute the speedup obtained by introducing this enhancement using the **Amdhal law**.

Then compute the speedup obtained if also "*Branch*" instructions are modified so that they take 5 cycle per instruction (that is the speedup obtained with both enhancements).

Exercises 5 (3 points) Performance equation

Suppose we have made the following measurements, where we are considering Arithmetic and logic instructions - A&L - and the subset of only integer multiplications and Divisions - MD:

Frequency of A&L operations = 45%
Average CPI of A&L operations = 3
Average CPI of other instructions = 2.2
Frequency of MD = 10%
CPI of FPM = 7

Assume that the two design alternatives are to decrease the CPI of A&L to 2.5 or to decrease the average CPI of all MD operations to 3.4.
Compare these two design alternatives using the processor performance equation.

Exercise 6 (3 points) – Number representation

Given the values A= 01 01 00 00 00 11 -12 and B= 00 00 00 10 11 10 convert them in decimal and show the execution of operation A+B. Verify the value of the result.

Exercise 7 (2 points) - N**umber representation**

- Determine the moduli set for representing values in the number range [0; 719] in the **residue number system** using:
    - a moduli set consisting of **3 moduli** and a moduli set consisting of **4 moduli.**
- Compare the two different choices and the conventional binary system with respect to the number of bits necessary for representing the range [0; 719].
- Represent the 600 using both selected moduli sets.

Exercise 8 (3 points) – Arithmetic circuit time and area

Draw a pipelined adder for binary values consisting of four bits. Compute the **time** (propagation delay) to obtain the first addition and **area** required by the whole circuit.