

# GPU and Heterogeneous Systems – A.Y. 2021-22

Scuola di Ingegneria Industriale e dell'Informazione

Instructor: Prof. Antonio Miele



January 27, 2023 – FIRST PART OF THE EXAM

Surname:	Name:	Person Code:
----------	-------	--------------

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

Instructions:

- **Duration: 40 minutes**
- 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.

## Question 1

Explain which are the three features of the graphics pipeline that have been parallelized in the GPU architecture.

## Question 2

Draw the Gantt chart of the execution of the various functions in the following three cases.

(a) Assume that `foo` execution is shorter than `cpuFoo` one.

```
foo<<<blocks, threads>>>();  
cpuFoo();  
foo<<<blocks, threads>>>();  
cudaEventRecord(event1);  
cudaEventSynchronize(event1);  
cpuFoo();
```

(b) Assume that `foo` execution is longer than `goo` one.

```
cudaStreamCreate(&stream1);  
cudaStreamCreate(&stream2);  
foo<<<blocks, threads, 0, stream1>>>();  
cudaEventRecord(event1, stream1);  
goo<<<blocks, threads, 0, stream2>>>();  
cudaStreamWaitEvent(stream2, event1);  
foo<<<blocks, threads, 0, stream2>>>();
```

(c) Assume that `foo` execution is longer than `goo` one.

```
cudaStreamCreate(&stream1);  
cudaStreamCreate(&stream2);  
foo<<<blocks, threads, 0, stream1>>>();  
goo<<<blocks, threads, 0, stream2>>>();  
foo<<<blocks, threads>>>();  
foo<<<blocks, threads, 0, stream2>>>();
```

## Question 3

Briefly describe CUDA memory model; for each component specify name, type of usage, type of access (read/write or read only) and scope.

## Question 4

Describe the main strategies to accelerate the Smith-Waterman algorithm in CUDA.

### Question 5

In the following snippet of code, how many times will `foo()` and `goo()` be executed? Motivate the answer.

```
#pragma acc parallel num_gangs(16)
{
    #pragma acc loop gang
    for (int i=0; i<32; i++) {
        goo(i);
    }
    foo();
}
```

# GPU and Heterogeneous Systems – A.Y. 2021-22

Scuola di Ingegneria Industriale e dell'Informazione

Instructor: Prof. Antonio Miele



January 27, 2023 – **SECOND PART OF THE EXAM**

Surname:	Name:	Personal Code:
----------	-------	----------------

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

Instructions:

- **Duration: 1 hour and 15 minutes**
- 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.

## Question 1

Implement two CUDA functions to accelerate the `bodyForce()` and `updateBodyPositions()` functions.

## Question 2

Modify the main function to execute the simulation on the GPU by calling previously defined kernels.

## Question 3

Describe how you would modify the `bodyForce()` to use the shared memory.

The source code can be downloaded from: <https://miele.faculty.polimi.it/nbody.c>

```

/* In physics, the n-body simulation is the simulation of the motion of a set of M
different objects due to the gravitational interaction among them. The following
program implement an N-body simulation. Each object to be simulated is modeled
by means of the Body_t struct that contains fields for the position and the speed
in a 3D space; data of N different objects is stored in an Body_t array.
The simulation starts by computing randomly the initial position and speed of each
object. Then a number of time steps are simulated; at each time step the bodyForce()
function computes the new speed of each object due to the gravitational interaction
and updateBodyPositions() functions computes the new position of each object.
*/

```

```

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>

#define SOFTENING 1e-9f

typedef struct {
    float x, y, z, vx, vy, vz;
} Body_t;

double get_time();
void randomizeBodies(float *data, int n);
void bodyForce(Body_t *p, float dt, int n);
void updateBodyPositions(Body_t *p, float dt, int n);

void randomizeBodies(float *data, int n) {
    for (int i = 0; i < n; i++) {
        data[i] = 2.0f * (rand() / (float)RAND_MAX) - 1.0f;
    }
}

void bodyForce(Body_t *p, float dt, int n) {
    for (int i = 0; i < n; i++) {
        float Fx = 0.0f, Fy = 0.0f, Fz = 0.0f;

        for (int j = 0; j < n; j++) {
            float dx = p[j].x - p[i].x;
            float dy = p[j].y - p[i].y;
            float dz = p[j].z - p[i].z;
            float distSqr = dx*dx + dy*dy + dz*dz + SOFTENING;
            float invDist = 1.0f / sqrtf(distSqr);
            float invDist3 = invDist * invDist * invDist;

            Fx += dx * invDist3;
            Fy += dy * invDist3;
            Fz += dz * invDist3;
        }

        p[i].vx += dt*Fx;
        p[i].vy += dt*Fy;
        p[i].vz += dt*Fz;
    }
}

void updateBodyPositions(Body_t *p, float dt, int n) {
    for (int i = 0 ; i < n; i++) {
        p[i].x += p[i].vx*dt;
        p[i].y += p[i].vy*dt;
        p[i].z += p[i].vz*dt;
    }
}

```

```

int main(const int argc, const char** argv) {
    int nBodies = 30000;
    if (argc > 1) nBodies = atoi(argv[1]);

    const float dt = 0.01f; // time step
    const int nIters = 10; // simulation iterations

    double cpu_start, cpu_end;

    Body_t *p = (Body_t*)malloc(nBodies*sizeof(Body_t));

    randomizeBodies((float*)p, 6*nBodies); // Init position / speed data

    cpu_start = get_time();
    for (int iter = 1; iter <= nIters; iter++) {
        bodyForce(p, dt, nBodies); // compute interbody forces
        updateBodyPositions(p, dt, nBodies); // integrate position
    }
    cpu_end = get_time();
    double avgTime = (cpu_end - cpu_start) / nIters;

    printf("%d Bodies: average %0.3f Billion Interactions / second\n", nBodies, 1e-9 *
nBodies * nBodies / avgTime);
    free(p);

    return 0;
}

// function to get the time of day in seconds
double get_time() {
    struct timeval tv;
    gettimeofday(&tv, NULL);
    return tv.tv_sec + tv.tv_usec * 1e-6;
}

```