

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

Scuola di Ingegneria Industriale e dell'Informazione

Instructor: Prof. Antonio Miele



September 6, 2022 – **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 why NVIDIA GPUs do not provide any support for grid-level synchronization.

Question 2

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

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

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

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

Question 3

Explain why OpenCL mainly adopts the just-in-time (JIT) compilation for the kernels to execute and which is the main exception where the JIT approach cannot be used.

Question 4

Describe the data structures used in a CUDA implementation of the graph search application and how computation is parallelized on them.

Question 5

Explain why for an NVIDIA GPU data organization in a structure of arrays may be more efficient for memory accesses than in an array of structures.

Structure of arrays:

```
typedef struct {  
    float x[N];  
    float y[N];  
} innerArray_t;  
innerArray_t mySoA;
```

Array of structures:

```
typedef struct {  
    float x;  
    float y;  
} innerStruct_t;  
innerStruct_t myAoS[N];
```

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

Scuola di Ingegneria Industriale e dell'Informazione

Instructor: Prof. Antonio Miele



September 6, 2022 – **SECOND PART OF THE EXAM**

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

Question	1	2	3	OVERALL
Max score	5,5	5,5	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 basic CUDA kernel functions to accelerate the compute-intensive functions in the following C program.

Question 2

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

Question 3

Implement a new CUDA kernel function to accelerate the first compute-intensive function (`findString`) by using shared memory only on the text array. Specify if any change has to be applied in the main function.

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

```

/*
 * This program takes in input a text and a string, being two arrays of char values, and
 * computes how many times the string appears in the text. In particular:
 * - function findString receives in input the text and the string and saves in each
 *   position i of a third vector called match, 1 if an occurrence of the string has been
 *   found in the text starting the index i, 0 otherwise.
 * - function countMatches receives the vector match in input and count the number of values
 *   equal to 1 (i.e., it counts the number of occurrences of the string in the text).
 * - the main function receives as arguments the size of text and string and (for the sake
 *   of brevity) generates randomly the content of the two vectors, invokes the two
 *   functions above and prints the result on the screen.
 */

#include <stdio.h>
#include <stdlib.h>
#define MAXVAL 2

void findString(char* text, int textDim, char* str, int strDim, char* match);
int countMatches(char *match, int num);

//kernel function 1: identify strings in the text
void findString(char* text, int textDim, char* str, int strDim, char* match) {
    int i, j, ok;
    for(i=0; i<textDim-strDim+1; i++){
        for(j=0, ok=1; j<strDim && ok; j++){
            if(text[i+j]!=str[j])
                ok=0;
            match[i] = ok;
        }
    }
}

//kernel function 2: count matches
int countMatches(char *match, int num) {
    int i, count;
    for(i=0, count=0; i<num; i++){
        count+=match[i];
    }
    return count;
}

int main(int argc, char **argv) {
    char *text, *str, *match;
    int count, textDim, strDim, i;

    //read arguments
    if(argc!=3){
        printf("Please specify sizes of the two input vectors\n");
        return 0;
    }
    textDim=atoi(argv[1]);
    strDim=atoi(argv[2]);

    //allocate memory for the three vectors
    text = (char*) malloc(sizeof(char) * (textDim));
    str = (char*) malloc(sizeof(char) * (strDim));
    match = (char*) malloc(sizeof(char) * (textDim-strDim+1));

    //initialize input vectors (code omitted for the sake of space)

    //execute on CPU
    findString(text, textDim, str, strDim, match);
    count = countMatches(match, textDim-strDim+1);

    //print results (code omitted for the sake of space)

    free(text);
    free(str);
    free(match);

    return 0;
}

```