Graphics Processing Units (GPUs): Architecture and Programming

Spring 2022 – Final Exam

Important Notes- READ BEFORE SOLVING THE EXAM

- **If you perceive any ambiguity in any of the questions, state your assumptions clearly and solve the problem based on your assumptions. We will grade both your solutions and your assumptions.**
- **This exam is take-home.**
- **The exam is posted on Brightspace, at 4:55pm EST of Tue May 3rd.**
- **You have up to 23 hours and 55 minutes to submit on Brightspace, similar to assignments submissions.**
- **You are allowed only one submission, unlike assignments and labs.**
- **Upload one pdf file that contains all your answers.**
- **This exam contains three problems, with a total of 100 points.**
- **Your answer sheet must be organized as follows:**
  - **The very first page of your answer must contain:**
    - **You Last Name**
    - **Your First Name**
    - **Your NetID**
  - **In your answer sheet, answer one problem per page (even if your answer takes one line).**

Problem 1

a. [10 points] All the threads in a warp share the same front-end (i.e. fetch, decode, etc). State two reasons as to why did GPU designers decide to do so instead of having a front end for each thread. That is, one front-end for each SP?

b. [5 points] Why did NVIDIA GPU designers decide to group several SPs in an SM?

c. [15 points] If the warp concept did not exist, what will be the implications on GPU programming? State three implications and, for each one, explain in 1-2 lines.

d. [10 points] Suppose we want to implement vector additions on multi-GPU systems because the two vectors we want to add are huge. Each thread will be responsible for few hundred elements. Which is more beneficial: using the traditional cudamalloc() and cudaMemcpy()? or using unified memory? And why?

Problem 2

a. [10 points] Suppose you are writing a kernel where each thread loads two integers from two different addresses (addr1 and addr1) in global memory, processes each integer, then writes the result back to the same addresses. Which one of the following two scenarios is better? and why [State two reasons]? Assume processing the int takes one instruction only and that addr1 and addr2 and very far from each other. Each thread calculates its own addr1 and addr2 based on each unique ID.

Scenario 1:	**Scenario 2:**
load integer from addr1 and put it into a register	load integer from addr1 and put it into a register
load integer from addr2 and put it into a register	process the first integer
process the first integer	write the new first integer to addr1
process the second integer	load integer from addr2 and put it into a register
write the new first integer to addr1	process the second integer
write the new second integer to addr2	write the new second integer to addr2

b. [10 points] Suppose X is an integer that exists in shared memory. What happens if two threads, from two different blocks, write to X at the same time? There are several scenarios, discuss each one in no more than two lines per scenario.

c. [5 points] Can more than one grid exist in the same GPU at the same time? Explain in 1-2 sentences.

d. [5 points] Even though registers are much faster than shared memory, it is sometimes more beneficial to put data in shared memory. When does this happen?

Problem 3

Suppose we have the following code snippet. The programmer launches the kernel with:
printexample<<<3,2>>>();

```
__global__ printexample(){

        __shared__ int x = 7;
        if(blockIdx.x > 0)
                printf("%d", blockIdx.x);
        else
                printf("x");
        __synchthreads();
        printf("%d", threadIdx.x);
}
```

a. [12 points] For each one of the following statements, specify whether it is a "possible" output or "not possible" output (i.e. can never be printed on the screen no matter how many times we execute the kernel). No need to justify your choice, just write (a. , b. , …) and next to each one ("possible", or "not possible").

a. 770100010001	b. 000100011101	c. 001000107710
d. 777000000111	e. 000000011177	f. 001077010001

b. [6 points] If we remove the __shared__ keyword, which ones of your answers above will change? Justify.

c. [6 points] If we remove the __synchthreads(), which ones of your answers (in part a) above will change? Justify.

d. [6 points] How many warps are created when the kernel is launched? Explain.