Important information about plagiarism:

While you are encouraged to discuss lecture topics with your peers, assignments must be your own work and plagiarism of any kind will not be tolerated. Submitted assignments will be automatically cross-checked with all other submissions in the class. Anyone who plagiarizes (i.e., copies code or allows their code to be copied by others) will receive a zero (0) on the assignment and be referred to the department chair for more severe disciplinary action.

Introduction

In this assignment, you are going to write a CUDA program to factor an integer.

One of the properties that make a security system secure is that that hackers can only break the system by brute-force search. RSA is such a system that it depends on the fact that factoring a large integer is a difficult task; and there is no known algorithm that can easily factor a large integer. Brute-force search is currently the best-known method.

Requirement:

Your program must be a CUDA program. When executed, it should launch a grid of 256 blocks, with 256 threads in each block. However, you cannot assume this knowledge in your kernel. That is, you must calculate the total number of threads by using `gridDim` and `blockDim`.

You can, however, assume that the input is a valid integer between 3 and 10^{18} (inclusive). The C `int` type is not big enough to hold a value this big, so we must use the `long long` type.

To give you an idea on how the program should work, a sequential version of the program (`sfactor.cu`) is provided. You can compile this program using the following command[1]:

```
nvcc -o sfactor sfactor.cu
```

You should name your program `pfactor.cu`, and you should be able to compile it using the following command:

```
nvcc -o pfactor pfactor.cu
```

For your convenience, a skeleton file has been provided with all supporting code in place.

You can run the sequential and parallel program as follow, respectively:

```
./sfactor 11010010001
```

[1] You should also be able to use gcc to compile this program. However, since nvcc on Linux is backward compatible with gcc, we choose to always use nvcc for consistent.

```
./pfactor 11010010001
```

Use of AI generated contents

Light use of AI generated content is allowed. By light use, it usually means less than 10% of the stuff you submitted. Examples of such content include (but not limited to) co-pilot for code contents and Chat-GPT for non-code contents.

However, such uses must be reported. The report will be used to determine if your use of AI-generated content is light or not. If you are found using unreported AI-generated contents in your submission, the entire assignment will receive a grade of zero, and the incident will be reported to the department chair for further investigation.

In coding assignments, you should add the report at the end of your code in the form of code comments. If applicable, this will not count as fulfilling documentation requirement.

For each item you are reporting, you must include at least the following (repeat for every and all items):

- Date you use the service.
- Name of the service.
- The prompt you used.
- The response generated.
- The part of the response you have used in your submission.

Please note that you must report the use of AI-generated contents even if you have modified the contents before using it in your submission.

Example:

```
# Date: July 1, 2024
# Service: Co-pilot
# Prompt: type in "def convert_int_to_string"
# Response:
#   def convert_int_to_string(number):
#       return str(number)
# Used: the entire response
```

Submission Checklist

- ☐ Can you compile your file on the cluster without any compiler errors and/or warnings?
- ☐ Can you execute your program without any runtime errors?
- ☐ Is your program written in CUDA C?
- ☐ Do you compute the grid dimension in your kernel instead of assuming a fixed grid dimension?

☐ Have you included comments in your program to explain how your program works?

Submission Requirement:

Except for `pfactor.cu`, you should not modify any files provided. You only need to submit the following file:

- Your program file `pfactor.cu`.

Please submit the file to Canvas – due date: Sunday November 9.