Parallel Computer Architecture
Final Exam
Gaussian Elimination

Prof. Naga Kandasamy
ECE Department
Drexel University

March 12, 2018

This problem, worth 25 points, is due March 24, 2018, by 11:59 pm via BBLearn. You may work on this problem in a team of up to two people. One submission per group will suffice. Please submit original work. Solutions copied from other students or from online sources will result in a grade of zero for the entire exam. You may discuss high-level approaches to solve the problem with Shihao or me, but we will not help you with the code itself.

Consider the problem of solving a system of linear equations of the form

$$\begin{array}{ccccc} a_{0,0}x_0 & + a_{0,1}x_1 & + \cdots & + a_{0,n-1}x_{n-1} & = b_0, \\ a_{1,0}x_0 & + a_{1,1}x_1 & + \cdots & + a_{1,n-1}x_{n-1} & = b_1, \\ . & . & & . & . \\ . & . & & . & . \\ a_{n-1,0}x_0 & + a_{n-1,1}x_1 & + \cdots & + a_{n-1,n-1}x_{n-1} & = b_{n-1}. \end{array}$$

In matrix notation, the above system is written as $Ax = b$ where A is a dense $n \times n$ matrix of coefficients such that $A[i, j] = a_{i,j}$, b is an $n \times 1$ vector $[b_0, b_1, \ldots, b_{n-1}]^T$, and x is the desired solution vector $[x_0, x_1, \ldots, x_{n-1}]^T$. From here on, we will denote the matrix elements $a_{i,j}$ and x_i by $A[i, j]$ and $x[i]$, respectively. A system of equations $Ax = b$ is usually solved in two stages. First, through a set of algebraic manipulations, the original system of equations is reduced to an upper triangular system of the form

We write the above system as $Ux = y$, where U is an upper-triangular matrix, that is, one where the subdiagonal entries are zero and all principal diagonal entries are equal to one. More formally, $U[i, j] = 0$ if $i > j$, otherwise $U[i, j] = u_{i,j}$, and furthermore, $U[i, i] = 1$ for $0 \leq i < n$. In the second stage of solving a system of linear equations, the upper-triangular system is solved for the variables in reverse order, from $x[n - 1]$ to $x[0]$ using a procedure called back-substitution.

$$
\begin{array}{lllll}
x_0 & + u_{0,1}x_1 & + u_{0,2}x_2 & + \cdots & + u_{0,n-1}x_{n-1} = y_0, \\
 & x_1 & + u_{1,2}x_2 & + \cdots & + u_{1,n-1}x_{n-1} = y_1, \\
 & & & & \quad \vdots \qquad\qquad \vdots \\
 & & & & x_{n-1} = y_{n-1}.
\end{array}
$$

A serial implementation of a simple Gaussian elimination algorithm is shown below. The algorithm converts the system of linear equations $Ax = b$ into a unit upper-triangular system $Ux = y$. We assume that the matrix u shares storage with A and overwrites the upper-triangular portion of A. So, the element $A[k, j]$ computed in line 5 of the code is actually $U[k, j]$. Similarly, the element $A[k, k]$ that is equated to 1 in line 8 is $U[k, k]$. Also, we assume that $A[k, k] \neq 0$ when it is used as a divisor in lines 5 and 7. So, our implementation is numerically unstable, though it should not be a concern for this assignment. For k ranging from 0 to $n - 1$, the Gaussian elimination procedure systematically eliminates the variable $x[k]$ from equations $k + 1$ to $n - 1$ so that the matrix of coefficients becomes upper-triangular. In the k^{th} iteration of the outer loop (line 3), an appropriate multiple of the k^{th} equation is subtracted from each of the equations $k + 1$ to $n - 1$.

```
 1: procedure GAUSS_ELIMINATE(A, b, y)
 2: int i, j, k;
 3: for k := 0 to n − 1 do
 4:    for j := k + 1 to n − 1 do
 5:       A[k, j] := A[k, j]/A[k, k];     /* Division step. */
 6:    end for
 7:    y[k] := b[k]/A[k, k];
 8:    A[k, k] := 1;
 9:    for i := k + 1 to n − 1 do
10:       for j := k + 1 to n − 1 do
11:          A[i, j] := A[i, j] - A[i, k] × A[k, j];     /* Elimination step. */
12:       end for
13:       b[i] := b[i] − A[i, k] × y[k];
14:       A[i, k] := 0;
15:    end for
16: end for
```

Develop a parallel formulation of `GAUSS_ELIMINATE` for the GPU. Complete the functionality of Gaussian elimination on the GPU by editing the `gauss_eliminate_on_device()` function in `gauss_eliminate.cu`. You may develop additional kernels and host-side code as needed. The program given to you accepts no arguments. The upper-diagonal matrix generated by the GPU is compared against the CPU result and if the solutions match within a certain tolerance, the application will print out "Test PASSED" to the screen before exiting.

For maximum points, parallelize both the division and elimination steps using two separate kernels, and optimize the performance of your GPU code via efficient use of the memory hierarchy. Try to ensure that your accesses to the matrix elements in global memory are coalesced. You may have to use double precision operations on the GPU to pass the acceptance test with respect to the

reference result. Upload all of the files needed to run your code as a single zip file on BBLearn. Also, provide a short report describing how you designed your kernel, using code or pseudocode to help the discussion, and the amount of speedup obtained over the serial version for the following matrix sizes: 512×512, 1024×1024, and 2048×2048. Ignore the overhead due to CPU/GPU communication when reporting speedup.