CS6963: Parallel Programming for GPUs
Midterm Exam
April 4, 2011

Instructions:
This is an in-class, open-note exam. Please use the paper provided to submit your responses. You can include additional paper if needed. The goal of the exam is to reinforce your understanding of issues we have studied in class.

CS6963: Parallel Programming for GPUs
Midterm Exam
April 4, 2011

I. Definitions (10 points)

Provide a very brief definition of the following terms:

a. Temporal locality

b. SIMT

c. Scoreboarding

d. Memory latency

e. Local memory (Nvidia architecture definition)

II. Short answer (20 points)

Provide a very brief answer to the following four questions.

a. Describe how you can exploit spatial reuse in optimizing for memory bandwidth on a GPU. (Partial credit: what are the memory bandwidth optimizations we studied?)

b. Given examples we have seen of control flow in GPU kernels, describe ONE way to reduce divergent branches for ONE of the following: consider tree-structured reductions, even-odd computations, or boundary conditions.

c. Regarding floating point support in GPUs, how does the architecture permit trading off precision and performance?

d. What happens if two threads assigned to different blocks write to the same memory location in global memory?

III. Problem Solving (60 points)

In this set of three questions, you will be asked to provide code solutions or explanations related to code solutions to solve particular problems. This portion of the exam may take too much time if you write out the CUDA solution in detail. I will accept responses that sketch the solution, without necessarily writing out the code or worrying about correct syntax. Just be sure you have conveyed the intent and issues you are addressing in your solution.

a. **Data placement in the memory hierarchy**

Given the following CUDA code, for each data structure in the thread program, what is the most appropriate portion of the memory hierarchy to place that data (constant, global, shared, or registers) and why. Feel free to give multiple answers for some data in cases where there are multiple possibilities that are all appropriate. In each case, explain how you would modify the CUDA code to use that level of the memory hierarchy.

```
#define N 512
float a[N], b[N], c[N][N], d[N], e[N];

__global compute(float *a, float *b, float *c, float *d, float *e) {
float temp;
int index = blockIdx.x*blockDim.x + threadIdx.x

for (j =0; j<N; j++) {
   temp = (c[index][j] + b[j])*d[e[index]];
   a[index] = a[index] – temp;
}


}
```

b. **Tiling for constant memory**

The following sequential computation uses a large read-only array A and multiplies another input by it to produce an updated result. Suppose you want to place A in constant memory, which is limited to 64 Kbytes of storage. Sketch the CUDA code that would place this data in constant memory and correctly synchronize the host code. You will need to think about how to decompose the computation such that there are no data races and array A is reused in constant memory. This may require a transformation to the loop order.

```
#define N 512
float A[512][512];
float B[512][512];

for (k=0; k<N; k++) {
   for (j=0; j<N; j++) {
      for (i=0; i<N; i++) {
           B[i][j] += A[j][k];
      }
   }
}
```

c. **Parallel partitioning and synchronization (Jacobi)**
Without writing out the CUDA code, consider a CUDA mapping of the sequential Jacobi code below. Answer should be in three parts, providing opportunities for partial credit: (i) where are the data dependences in this computation? (ii) how would you partition the computation across threads and blocks? (iii) how would you add synchronization to avoid race conditions?

```
#define TS 100
float a[1024][1024];

for (k=0; k<TS; k++) {
   for (j=1; j<1023; j++) {
      for (i=1; i<1023; i++) {
         l[i][j] = (a[i+1][j] + a[i-1][j] + a[i][j+1] + a[i][j-1])/4;
      }
   }
   for (j=1; j<1023; j++) {
      for (i=1; i<1023; i++) {
         a[i][j] = l[i][j];
     }
  }
}
```

(Brief) Essay Question (10 points)
Pick one of the following three topics and write a very brief essay about it, no more than 3 sentences.

a. We talked about sparse matrix computations with respect to linear algebra, graph coloring and program analysis. Describe a sparse matrix representation that is appropriate for a GPU implementation of one of these applications and explain why it is well suited.
b. We talked about how to map tree-structured computations to GPUs. Briefly describe features of this mapping that would yield an efficient GPU implementation.
c. We talked about dynamic scheduling on a GPU. Describe a specific strategy for dynamic scheduling (static task list, dynamic task list, wait-free synchronization) and when it would be appropriate to use it.