CSCI-UA.0480-003

Parallel Computing

Final Exam

Spring 2016 - May 16th (90 minutes)

NAME: **ID:**

NetID:

- If you have to make assumptions to continue solving a problem, state your assumptions clearly.
- You answer on the question sheet. You can use extra white papers if you want.

1. [1] We know that a block cannot be assigned to an SM until it gets all the resources it needs beforehand. What is the advantage of doing so?

So that scheduling of a warp for execution takes zero cycles.

2. We have seen that if-else may lead to branch divergence in a warp due to lockstep execution of instructions. Now, suppose there is a kernel that has an *if without else*.

a. [2] Can this also lead to **performance loss** in some cases, relative to non-branch divergence? Justify your answer. No need to write code, just explain.

Yes, performance loss relative to when all threads has FALSE. Because if only some threads have true, they will execute the if, then all threads execute the rest of the kernel.

b. [2] Can this also lead to **NO performance loss** in some cases, relative to non-branch divergence? Justify your answer. No need to write code, just explain.

Yes, no performance loss relative to when all threads have TRUE. That is because in both cases (divergence and no-divergence) both the if and the rest of the kernel will be executed.

3. [2] Can we have a race condition among threads belonging to the same warp? Justify your answer.

Yes, if two (or more) threads in the same warp try to modify the same variable.

4. [6] For each variable in the following code: identify the scope of the variable, justify your choice, and for each variable identify potential race condition, if any. You can assume that a, b, c, i, N, and j have been defined somewhere before the parallel block.

```
#pragma omp parallel for private(a,b)
for (i = 0; i < N; i++) {
        int x = 0;
        c--;
        for (j = i; j < N; j++)
                x += func(c, b[j]);
        a[i] = x;
}
```

Variable	**Private/ shared**	**Why?**	**race cond? (Y/N)**	**Why?**
a[]	P	private() clause	N	private
b[]	P	private clause	N	private
c	S	defined before parallel structure	Y	shared and modified
i	P	loop index is private by definition	N	private
j	S	defined before parallel structure		shared and modified
x	P	defined inside the parallel structure	N	private

5. [2]State two shortcomings of Amadahl’s law.
   - Does not take communication and memory access overhead into account.

   - F (the sequential part) is hard to calculate.

6. For the following piece of code (assume very large number of cores):

```
…
int globalvalue = 0;
int main() {
        int numprocs, rank;
        int i = 0;

        MPI_Init(NULL, NULL);
        MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);

        #pragma omp parallel for shared(result) reduction(+:globalvalue)
        for( i = 0; i < 2+rank ; i++)
        {
          globalvalue ++;
          …rest of loop body …
        }

        MPI_Finalize();
 }
```

We execute the above code with: mpirun -n 4 *progname*

a. [2]How many threads we will end up having in the *whole system*? Explain.

4 processes: each process will generate threads = loop iterations.
So: 2 + 3 + 4 + 5 = 14

b. [1] Just before executing MPI_Finalize(), how many instances of *globalvalue* do we have in the system?

4

c. [2] Is there a potential race condition in globalvalue++ ? Justify
No, because reduction operation is used and hence will be handled by OpenMP runtime.