YOUR NAME HERE:

FINAL EXAM: CS/ECE/ME/EP 759
NOTE: DO NOT COMMUNICATE WITH ANYBODY DURING THE DURATION OF THE EXAM

Problem 1. (40 points, 2 points for each correct answer. Circle only **one** answer for each question, the one that you think is the best answer). Time Budget: About 40 mins.

1.1. A program spends 60% of its 200 seconds of the CPU execution time in a portion of the code that can be parallelized. If you have 12 threads that accomplish perfect linear scaling with no overhead, after the parallelization the program will run in

   a. 90 seconds
   b. 120 seconds
   c. 60 seconds

1.2. The concept of Translation Lookaside Buffer

   a. Is tied to the concept of cache usage
   b. Is tied to the concept of address translation
   c. Is tied to the concept of register usage

1.3. Which of the following statements is false?

   a. CUDA only works if all threads end up executing the same machine instructions
   b. Not all CUDA threads in a warp should execute the same instructions
   c. There are certain classes of parallel applications where CUDA cannot be used effectively

1.4. In CUDA, data available in the shared memory of a SM is available for read/write to:

   a. All of the blocks specified by the execution configuration at any time during the execution of the kernel
   b. The threads of the block dispatched for execution on the SM while it is executed by that SM
   c. Only to one thread in a block executed on that SM at a particular time

1.5. Which one below is an example of Instruction Level Parallelism

   a. Pipelining
   b. Cache coherence
   c. Non-uniform memory access (NUMA)

1.6. A `__syncthreads()` call in CUDA (select the most encompassing statement that is still true)

   a. Ensures that all threads in a kernel are synchronized
   b. All threads in a block are synchronized
   c. All the threads in a warp are synchronized

1.7. Caches are most useful because
   a. They significantly increase the amount of actual memory that the chip can count on
   b. They significantly reduce the amount of time it takes to access data
   c. Both of the above

1.8. GPU computing is well suited for
   a. Fine grain parallelism
   b. Handling asynchronous parallel algorithms
   c. Applications that rely on recursive algorithms

1.9. The concept of branch prediction is
   a. A technique that is only relevant on CISC architectures
   b. Is an example of Instruction Level Parallelism
   c. Is not used anymore since it became obsolete

1.10. In CUDA, a kernel call is by default
   a. An asynchronous operation
   b. A synchronous operation
   c. It depends: some kernel calls are asynchronous but some are synchronous

1.11. The amount of virtual memory
   a. Is typically smaller than the amount of physical memory
   b. Is typically larger than the amount of physical memory
   c. Must always be exactly equal to the amount of physical memory

1.12. In OpenMP, the more likely cause of lack of scalability is
   a. NUMA aspects
   b. The cache coherence requirement
   c. The lack of enough main memory sockets on the motherboard

1.13. Using Flynn's taxonomy, which of the following computational models is currently the least used in practice?
   a. SIMD (single instruction, multiple data)
   b. MIMD (multiple instruction, multiple data)
   c. MISD (multiple instruction, single data)

1.14. The threads in OpenMP
   a. Are heavier (higher overhead to activate/deactivate) than CUDA threads
   b. Are lighter then CUDA threads
   c. Are about the same

1.15. OpenMP is principally used for
   a. Multi-core single-workstation computing
   b. Multi-node computing over a high-speed network
   c. Coarse-grain parallelism of asynchronous tasks

1.16. An implementation of the MPI standard such as OpenMPI or MPICH

a. Can only be used for distributed memory computers

b. Can only be used for shared memory computers

c. Can be used for either shared or distributed memory computers

1.17. For an architecture to be considered "superscalar":

a. More than one machine instruction could be issued for execution at each clock tick, yet in reality this might not always be the case

b. More than one machine instruction can be issued for execution at each clock tick, and this must always be the case

c. At most one instruction is issued at each clock tick. Yet, due to pipelining, you'll have multiple machine instructions being executed at the same time.

1.18. False sharing in OpenMP is more likely to occur

a. When the cache line is large

b. When the cache line size is small

c. The cache size line doesn't have anything to do with false sharing

1.19. In order for a modern CPU to execute vector instructions, the data being used **must**

a. Be aligned to the vector width

b. Be stored sequentially in memory

c. Have a length that is an integer multiple of 4

1.20. Which of these is the defining characteristic of Non-Uniform Memory Access (NUMA)?

a. Each thread has a different path to main memory

b. Multiple threads accessing the same memory location cause race conditions

c. Each set of caches has a different path to main memory

Problem 2. (10 points). Time Budget: About 10 mins.

Point out any problems you see in the code snippet below. How would you fix any problem and/or improve the performance of this piece of code?

```
#pragma omp parallel for
for(int i=0; i<N; i++) histogram[picture[i]]++;
```

Imagine that `picture[i]` returns the color index of a picture you took. Also, assume that there are only 8 colors, 0 through 7. The code runs OpenMP via 4 threads on a quad-core chip.

Problem 3. (10 points). Time Budget: About 10 mins.

Give an example; i.e., describe an application, where MPI is a better choice than using OpenMP for parallel computing. Explain why this is the case. No code is necessary here.

Problem 4. (10 points). Time Budget: About 10 mins.

a) In MPI, explain the difference between a blocking Send/Receive operation, and the non-blocking counterpart.

b) Why would we need the two blocking/non-blocking flavors of the Send/Receive operation?

Problem 5. (20 points). Time Budget: About 30 mins.

Assume that you use an NVIDIA GPU of compute capability 6.0. For its specifications that are relevant in the context of this discussion, see page 767 of the compiled PDF that contains all lectures. Alternatively, look at slide 6 of lecture 17 (October 13).

a) Suppose that in my code I have a kernel whose execution configuration specified that each CUDA block has 1024 threads in it. How many blocks should I have per SM for 100% occupancy?

b) Suppose that in my code I have a kernel whose execution configuration specified that each CUDA block has 512 threads in it. Up to how much shared memory (in KB) can each block ask for so that I can still run at 100% occupancy?

c) Suppose that in my code I have a kernel whose execution configuration specified that each CUDA block has 512 threads in it. Up to how many 32-bit registers can each thread use to still have 100% occupancy?

d) Suppose that in my code I have a kernel whose execution configuration specified that each CUDA block has 1024 threads in it. Can I get 50 percent occupancy? How or why not? Can I get 33% occupancy? How or why not?

e) (Trickier) Suppose that in my code I have a kernel whose execution configuration specified that each CUDA block has 32 threads in it. Each block uses 5 KB of shared memory. Each threads uses 80 registers. What occupancy will I get?

Problem 6. (10 points). Time Budget: About 20 mins.

Given the **`Point3D`** definition below and assuming that **`sizeof(float)`** = 4 bytes for an x86 processor with a vector width of 16 bytes, answer the questions a) through c) below.

```
struct Point3D {
      float x, y, z;
};
```

a) What is **`sizeof(Point3D)`** with

i) an alignment of 4 bytes

ii) an alignment of 8 bytes

b) How much space (in bytes) is used for padding assuming an alignment of 8 bytes?

i) For an array of structures (AoS) layout

```
Point3D points[11];
```

ii) For a Structure of Arrays (SoA) layout

```
struct Point3D_arrays {
      float x[11], y[11], z[11];
};
```

c) Assuming we can only load data from memory into one vector register at a time, how many memory loads would be required assuming an alignment of 8 bytes?

i) For an array of structures (AoS) layout

```
Point3D points[11];
```

ii) For a Structure of Arrays (SoA) layout

```
struct Point3D_arrays {
      float x[11], y[11], z[11];
};
```