CSCI-UA.0480-051: Parallel Computing
Midterm Exam (March 14th, 2024)

Total: 100 points
Important Notes - READ BEFORE SOLVING THE EXAM

- If you perceive any ambiguity in any of the questions, state your assumptions clearly and solve the problem based on your assumptions. We will grade both your solutions and your assumptions.
- This exam is take-home.
- The exam is posted on Brightspace, at the beginning of the March 14th lecture (2pm EST).
- You have up to 24 hours to submit on Brightspace (i.e. till March 15th 2pm EST), in the same way as you submit an assignment. However, unlike assignments, you can only submit once.
- Your answers must be very focused. You may be penalized for giving wrong answers and for putting irrelevant information in your answers.
- Your answer sheet must be organized as follows:
  - The very first page of your answer must contain only:
    * Your Last Name
    * Your First Name
    * Your NetID
    * Copy and paste the honor code shown in the rectangle at the bottom of this page.
  - In your answer sheet, answer one problem per page. The exam has seven main problems, each one must be answered in a separate page.
- This exam consists of 7 problems, with a total of 100 points.
- Your answers can be typed or written by hand (but with clear handwriting). It is up to you. But you must upload one pdf file containing all your answers.

Honor code (copy and paste to the first page of your exam)

- You may use the textbook, slides, the class recorded lectures, the information in the discussion forums of the class on Brightspace, and any notes you have. But you may not use the internet.
- You may NOT use communication tools to collaborate with other humans. This includes but is not limited to Google-Chat, Messenger, E-mail, etc.
- You cannot use LLMs such as chatGPT, Gemini, Bard, etc.
- Do not try to search for answers on the internet, it will show in your answer, and you will earn an immediate grade of 0.
- Anyone found sharing answers, communicating with another student, searching the internet, or using prohibited tools (as mentioned above) during the exam period will earn an immediate grade of 0.
- "I understand the ground rules and agree to abide by them. I will not share answers or assist another student during this exam, nor will I seek assistance from another student or attempt to view their answers."

Problem 1

a. [10] Suppose we have a core with both pipelining and superscalar execution (e.g., 4-way superscalar). Will this core benefit from having more execution units (e.g., increasing to 8-way superscalar)? Justify your answer in 1-2 lines.

b. [10] What is the difference between a process and a thread? Explain in 1-2 sentences.

c. [10] Can several threads, belonging to different processes, be executed on a shared memory machine? If yes, explain how, in 1-2 lines. If not, explain why not.

d. [6] If we have an eight-way superscalar core, how many instruction fetch units do we need to get the best performance? Justify your answer.

Problem 2

Suppose we have the following DAG that represents different tasks and their dependencies. The tasks are: A, B, C, D, E, F, G, H. A is the starting point. A depends on nothing. B depends on A. C depends on A. D depends on B. E depends on B and C. F depends on C. G depends on E and F. H depends on G. (Imagine a DAG diagram here depicting these dependencies).

The following table shows the execution time of each task if we execute it on a core of type X and if we execute it on core of type Y. Each CPU type is optimized for some type of operations.

Task	CPU type X	CPU type Y
A	2	3
B	4	2
C	6	8
D	3	5
E	5	3
F	2	4
G	7	6
H	1	2

a. [10] What is the minimum number of CPUs of each type that we need to get the highest speedup over sequential execution on CPU of type X? Show which CPU will execute which task(s) and calculate the final speedup.

b. [10] Repeat the problem above but using CPU of type Y.

c. [10] Suppose we can optimize the DAG by parallelizing one task. Which task would you choose and why? How would this change the execution time?

Problem 3

Suppose that MPI_COMM_WORLD consists of the four processes 0, 1, 2, and 3, and suppose the following code is executed after MPI has been initialized:

```
int x, y;
MPI_Status status;
switch(my_rank) {
case 0:
x = 10;
MPI_Send(&x, 1, MPI_INT, 1, 10, MPI_COMM_WORLD);
MPI_Recv(&y, 1, MPI_INT, 2, 20, MPI_COMM_WORLD, &status);
break;
case 1:
MPI_Recv(&x, 1, MPI_INT, 0, 10, MPI_COMM_WORLD, &status);
MPI_Send(&x, 1, MPI_INT, 3, 30, MPI_COMM_WORLD);
break;
case 2:
y = 20;
MPI_Send(&y, 1, MPI_INT, 0, 20, MPI_COMM_WORLD);
break;
case 3:
```

```
MPI_Recv(&y, 1, MPI_INT, 1, 30, MPI_COMM_WORLD, &status);
break;
}
```

a. [9 points] What will be the values of x and y for each of the 4 processes after executing the above code?

	P0	P1	P2	P3
x				
y				

b. [5] Is there a possibility that the communication among the 4 processes executes out of order? If yes, explain the reason. If not, why not?

c. [5] What will happen if we execute the above code with: `mpiexec {n 2`

d. [5] What would happen if the $\texttt{MPI}_S endin process 0 to process 1 use data of 20 instead of 10$?

Problem 4

a. [5] Explain Amdahl's Law and its implications for parallel computing scalability.

b. [5] What is load balancing, and why is it important in parallel computing? Give an example of a situation where load imbalance would significantly reduce performance.

Problem 5

a. [10] Describe the difference between OpenMP and MPI. When would you choose one over the other?

b. [10] Explain the concept of a race condition in parallel programming. Provide a simple code example demonstrating a race condition and explain how to prevent it.

Problem 6

Consider a parallel program that sorts a large array of numbers. The program divides the array into chunks, each assigned to a different processor. Each processor sorts its chunk locally. Then, a merge sort is used to combine the sorted chunks.

a. [10] Discuss potential bottlenecks in this approach.

b. [10] Propose at least two strategies to mitigate the bottlenecks identified in part (a).

Problem 7

a. [10] Explain the concept of false sharing in shared memory programming. How does it affect performance?

b. [10] Describe a technique to mitigate false sharing. Illustrate with a code example (in C or C++).