# 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 below.
  - In your answer sheet, answer one problem per page. The exam has eight main problems, each one must be answered in a separate page.
- This exam consists of 8 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

10 Suppose we have a core with only superscalar execution (i.e. no pipelining or hyperthreading). Will this core benefit from having a larger instruction cache? Justify your answer in 1-2 lines.

10 Can multiple threads within a single process share the same stack? If yes, explain how. If not, explain why not.

10 Can several processes, each with multiple threads, be executed on a distributed memory machine? If yes, explain how. If not, explain why not. Assume each node of the distributed machine has multiple CPUs.

6 If we have an eight-way superscalar core, how many instruction decoders might be beneficial to get the best performance? Justify.

# Problem 2

Suppose we have the following DAG representing tasks and dependencies:
A / B C / / D E F G / H I / J
Execution times (in milliseconds) for each task on CPU types X and Y:

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

10 What is the minimum number of CPUs of each type to achieve the highest speedup over sequential execution on CPU type X? Show task assignments and calculate the speedup.

10 Repeat the above, using CPU type Y.

10 If you could add a single dependency arrow to the DAG, which one would you add and why? Justify your choice based on potential performance improvements.

# Problem 3

MPI_COMM_WORLD has processes 0, 1, and 2. The following code runs after MPI initialization:

```
int x, y, z;
switch(my_rank) {
case 0:
x=2; y=3; z=1;
MPI_Send(&x, 1, MPI_INT, 1, 1, MPI_COMM_WORLD);
MPI_Recv(&y, 1, MPI_INT, 2, 2, MPI_COMM_WORLD, &status);
MPI_Send(&z, 1, MPI_INT, 2, 3, MPI_COMM_WORLD);
break;
case 1:
x=4; y=5; z=6;
MPI_Recv(&x, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, &status);
MPI_Send(&y, 1, MPI_INT, 2, 4, MPI_COMM_WORLD);
break;
case 2:
x=7; y=8; z=9;
```

```
MPI_Recv(&z, 1, MPI_INT, 0, 3, MPI_COMM_WORLD, &status);
MPI_Send(&y, 1, MPI_INT, 0, 2, MPI_COMM_WORLD);
MPI_Recv(&y, 1, MPI_INT, 1, 4, MPI_COMM_WORLD, &status);
break;
}
```

9 What are the final values of x, y, and z for each process?

| | P0 | P1 | P2 |
|---|---|---|---|
| x | | | |
| y | | | |
| z | | | |

5 Is there a potential deadlock? If yes, explain why. If not, why not?

5 What happens if we run with `mpiexec -n 1`?

5 What if we remove the `case 0:` statement?

# Problem 4

5 Does an application with significant communication overhead between parallel tasks exhibit good scalability? Explain.

5 If we have two threads with equal computational load but different memory access patterns, does this necessarily lead to performance degradation on a multicore system? Explain.

# Problem 5

Describe three different parallel programming models and their respective strengths and weaknesses. (15 points)

# Problem 6

Explain Amdahl's Law and its implications for parallel program performance. Provide a numerical example to illustrate its impact. (15 points)

# Problem 7

Consider a parallel algorithm for matrix multiplication. Describe how you would parallelize this algorithm and discuss the potential challenges and performance considerations. (15 points)

# Problem 8

Compare and contrast OpenMP and MPI. When would you choose one over the other? (15 points)