CSCI-UA.0480-051: Parallel Computing
Midterm Exam (Mar 10th, 2022)
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, assignments section, at the beginning of the Mar 10th lecture.**
- **You have up to 23 hours and 55 minutes from the beginning of the Mar 10th lecture to submit on Brightspace (in the assignments section).**
- **You are allowed only one submission, unlike assignments and labs.**
- **Your answers must be very focused. You may be penalized for wrong answers and for putting irrelevant information in your answers.**
- **You must upload a pdf file.**
- **Your answer sheet must have a cover page (as indicated below) and one problem answer per page (e.g. problem 1 in separate page, problem 2 in another separate page, etc).**
- **This exam has 3 problems totaling 100 points.**
- **The very first page of your answer is the cover page and must contain:**
  - **You Last Name**
  - **Your First Name**
  - **Your NetID**
  - **Copy and paste the honor code showed in the rectangle at the bottom of this page.**

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

- You may use the textbook, slides, 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 G-Chat, Messenger, E-mail, 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 or communicating with another student 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. [7] Suppose we have a core with pipelining but no superscalar or hyperthreading capabilities. Will this core benefit from branch prediction? Justify in no more than two lines.

[2] Yes, it will.
[5] Without branch prediction, whenever a condition branch is encountered, the fetch phase won't know which next instruction to fetch, introducing bubbles (i.e. empty phases) in the pipeline.

b. [8] Suppose we have an eight-core processor. Fill-in the following table for each scenario.

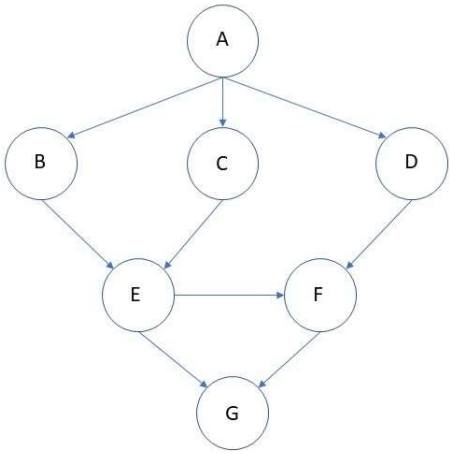
If each one of the 8 cores is	**The maximum number of *processes* that can execute on the whole processor is:**	**The maximum number of *threads* that can execute on the whole processor is:**
Only pipeline	8	8
Superscalar with no hyperthreading and each core has four execution units	8	8
Four-way hyperthreading but without branch prediction	32	32
Four-way hyperthreading with branch prediction	32	32

c. [10] "By having more variables shared among processes, we increase the chance of false sharing and hence performance will go down due to coherence overhead". Is this statement true or false? Justify.

[3] This statement is false.
[7] There cannot be any shared variables among processes because processes do not share virtual address space.

Problem 2

Assume we have the following task flow graph where every node is a task and an arrow from a task to another means dependencies. For example, task B cannot start before task A is done.



The following table shows the time taken by each task in nano seconds.

Task	Time Taken
A	10
B	5
C	5
D	10
E	5
F	10
G	10

a. [10] Can we have more than one span in the same DAG? If yes, cite all spans in the above DAG. If no, justify why not in no more than two lines.

[4] Yes, we can.
[6] In this DAG we have the following spans, each of which takes 40 nano seconds:

- A→B→E→F→G
- A→C→E→F→G
- A→D→F→G

b. [15] What is the minimum number of cores needed to get the maximum speedup? What is the speedup we get using the number of cores you specified? Specify, what will be the tasks that each core will execute to get the speedup you calculated. There may be several ways of assigning tasks to cores. Cite one of them that gives the needed speedup.

- [2] We need two cores.
- [8]The speedup = T_1/T_2 = 55/45 = 1.22

- [5] One core can execute: A, B, C the other: D, E, F, G Another solutions are also possible. The main thing is that B and C must be in one core while D in another.

c. [5] With the assignment of tasks to cores that you did in part b above, are we facing load imbalance? Why?

[2] Yes, we may face that.
[3] Because E, F, and G are dependent on each other. So, they may either be assigned to one core while the other is idle. Or they are distributed between the two cores but, due to the dependencies, one core will be executing a task while the other is waiting.

d. [5] Given the speedup you calculated in b above, what is the efficiency? Show all steps

Efficiency = speedup/#cores = 1.22/2 = 0.61 → That is, 61% efficiency.

e. [15] Suppose we execute the DAG on a single core (i.e. sequentially) that has a clock speed of 4GHz, and that each one of the tasks (A, B, C, D, E, F, and G) consists of 100 machine language instructions. What is the CPI? Show all your steps to get the whole credit.

CPI = (total number of cycles) / (total number of instructions) [3]
= (total time in seconds/ cycle time) / (#tasks * #instructions per task) [7]
= $(55*10^{-9} * freq) / (7*100)$ [5 points from this step to end]
= $(55*10^{-9} *4*10^{9})/700$
= 220/700
= 0.314
If you show different but correct steps, it is fine too.

Problem 3

Suppose that MPI_COMM_WORLD consists of three processes: 0,1, and 2. Also suppose the following code is executed (my_rank contains the rank of the executing process):

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

a. [9] Fill in the table below with the values of x, y, and z for each process after all processes are done executing their code.

	Process 0	Process 1	Process 2
x	4	6	8
y	8	8	2
z	8	4	8

b. [6] What will happen if we remove the "**break;**" that exists at the end of "**case 1:**"? Explain all consequences.

[2] If "break;" is removed from process1, the execution of process 1 will continue to process 2.
[4] This means process 1 will call two extra MPI_Bcast() that no other processes will call, causing process 1 to be blocked forever (i.e. deadlock).

c. [10] Is there a possibility that process 0 and process 1 both execute on the same core and at the same time? Justify your answer.

Yes
If this core is at least two-way hyperthreading.