CONCORDIA UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 426, Fall 2017 Instructor: R. Jayakumar

MID-TERM EXAMINATION

Date: Oct. 24, 2017 Time: 19:00–20:15

INSTRUCTIONS

1. Closed book exam.
2. Concise answers will be appreciated.
3. Answer All the questions in the space provided.
4. Write your student ID on the bottom-right corner of Pages 1 to 6.

Please fill in the following
Your Exam will not be marked if these details are missing

Student Name	
Student I.D. No.	
Student Signature	

For Examiner's Use Only

Question	Mark
1	/24
2	/24
3	/24
4	/28
Total	/100

PART A

Answer ALL the Questions in this Part (3×3×8 = 72%)

1. **Multicore Architecture and Programming**

Multicore architectures are MIMD shared memory multiprocessor architectures that execute multithreaded programs.

(a) [8%] What is the major difference between multicore architectures and traditional MIMD shared memory multiprocessor architectures?

How does this difference affect the programming of multicore processors?

(b) [8%] What is the major difference between multicore programming and traditional multithreaded programming?

How does this difference affect multicore programming?

(c) [8%] What is the major difference between homogeneous multicore architectures and heterogeneous multicore architectures?

Student ID: ____________________

Which one (homogeneous multicore architecture or heterogeneous multicore architecture) is easier to program? Why?

Which one (homogeneous multicore architecture or heterogeneous multicore architecture) has better computational efficiency? Why?

2. Multicore Programs

A multicore program typically has one control thread and many computation threads.

(a) [8%] Why should a multicore program have many computational threads? What is achieved by using such multiple computational threads?

Why should the multiple computational threads avoid synchronization among them? What is achieved by avoiding such synchronization?

(b) [8%] Why should a multicore program have only one control thread? What is achieved by using a single control thread?

Why should a multicore program avoid multiple control threads? What is achieved by avoiding such multiple control threads?

Student ID: ____________________

(c) [8%] What is the main problem in efficiently executing a multicore program on different processors with different number of hardware threads? Why?

How can an appropriate thread scheduler efficiently execute a multicore program on different processors with different number of hardware threads?

3. Data Parallelism and Task Parallelism

Multicore programming typically exploits data parallelism and task parallelism.

(a) [8%] Explain the difference between data parallelism and task parallelism pointing out which one is easier to exploit in a multicore program.

Explain the difference between the amounts of computation done within a computation thread implementing data parallelism and another computation thread implementing task parallelism. Why is this difference important?

(b) [8%] Why are CPUs good for task parallelism? Is there any problem in using the CPU for data parallelism? If so, explain the problem; if not, explain why not.

Why are GPUs good for data parallelism? Is there any problem in using the GPU for task parallelism? If so, explain the problem; if not, explain why not.

(c) [8%] Explain why TBB supports both data parallelism and task parallelism whereas CUDA supports only data parallelism

What is the problem in implementing task parallelism in CUDA? Why?

PART B

Answer ALL the Questions in this Part (4×7 = 28%)

4. Multicore Program Design

The following pseudo code abstracts the multi-flock bird simulation computation from the assignment.

```
get the number of flocks, num_flocks;
initialize the screen with the required number of flocks;

at each time point (every 1/30th of a second) do {

   display the current screen image using OpenGL;

   // Computation for each flock
   for (int flock = 0; flock < num_flocks; ++flock) do {

      // Computation for each bird
      for each bird in the flock do {
         update the parameters of the bird;
      }
   }
}
```

(a) [7%] Consider a multicore implementation of the given computation in which one computation thread is used for each flock.

Using appropriate pseudo code, illustrate the computation performed by each of the computation threads in this multicore implementation clearly showing any required synchronization.

Does this implementation exploit data parallelism or task parallelism? Why?

(b) [7%] Consider another multicore implementation of the given computation in which one computation thread is used for each bird.

Using appropriate pseudo code, illustrate the computation performed by the control thread in this multicore implementation clearly showing any required synchronization.

Does this implementation exploit data parallelism or task parallelism? Why?

Student ID: ________________

(c) [7%] In order to implement the given computation using TBB, would you use the (one thread per flock) implementation in (a) or the (one thread per bird) implementation in (b)? Why?

In order to implement the given computation using CUDA, would you use the (one thread per flock) implementation in (a) or the (one thread per bird) implementation in (b)? Why?

(d) [7%] Suppose the required computation (displaying the screen image and updating the parameters for all the birds) takes much more than the available time (1/30th of a second).

Suggest a way to further speed up the multicore implementation in this case, illustrating the control thread in this multicore implementation with appropriate pseudo code clearly showing any required synchronization

Does this implementation exploit data parallelism or task parallelism? Why?

*** ***END OF EXAMINATION*** ***

Student ID: ____________________