1. Multicore Programming

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?

(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?

2. Data Parallelism and Task Parallelism

Multicore programming typically exploits data parallelism and task parallelism.

(a) [8%] What is data parallelism? What should the control thread and computational threads do to exploit data parallelism in multicore programming?

Under what situation (data characteristic) it may not be possible to exploit data parallelism in multicore programming? Why?

(b) [8%] What is task parallelism? What should the control thread and computational threads do to exploit task parallelism in multicore programming?

Under what situation (task characteristic) it may not be possible exploit task parallelism in multicore programming? Why?

(c) [8%] Explain why it is easier to exploit data parallelism (than task parallelism) in multicore programming.

What is the problem in exploiting task parallelism in multicore programming? Why?

3. Threading Building Blocks (TBB)

Threading Building Blocks (TBB) is Intel's technology for multicore programming homogeneous processor cores.

(a) [8%] How does TBB exploit data parallelism? What should the programmer indicate in the program and what does the TBB runtime do to exploit data parallelism?

How is the data parallel computation specified in TBB.

(b) [8%] How does TBB exploit task parallelism? What should the programmer indicate in the program and what does the TBB runtime do to exploit ~~data~~ task parallelism?

How is the task parallel computation specified in TBB.

(c) [8%] How does the TBB scheduler improve the performance of a TBB program?

How does the TBB scheduler scale a TBB program to run on different multicore CPUs with different number of processor cores?

4. Multicore Program Design

The following pseudo code abstracts the multi-species game of life computation from the assignment.

```
get the number of species (num_species);
set the initial status of each cell;

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

   display the current screen image using OpenGL;

   for (int i = 0; i < num_species; ++i) do {
      update the status of each cell;
   }

}
```

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

Using appropriate pseudo code, illustrate the computation performed by the control thread in this multicore implementation.

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

(b) [7%] Consider another multicore implementation of the given computation in which the cell array is divided into multiple partitions and a computation thread performs the required computation of one of the partition for all the species.

Using appropriate pseudo code, illustrate the computation performed by each of the computation threads in this multicore implementation.

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

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

Explain the problem in using the implementation you did not select. Why is this problem important in TBB?

(d) [7%] Suppose the required computation (displaying the screen image and updating the cell status for all the species) takes much more than the available time ($1/30^{th}$ of a second).

Suggest a way to further speed up the multicore implementation selected in (c) in this case, explaining how your suggestion reduces the time by exploiting more parallelism.

Should your suggested solution be implemented within the control thread or within the computation threads? Why?