UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Science and Technology

EXAMINATION QUESTION PAPER

Exam in:	INF-3201 Parallel Programming
Date:	Friday, 2018-11-30
Time:	09:00 - 13:00
Place:	Adm. building, Aud. Max.
Approved aids:	None / Digital exam
Type of sheets (squares/lines):	Digital exam
Number of pages incl. cover page:	4
Contact person during the exam:	John Markus Bjørndalen
Phone:	+47 90148307
	Visit round around 10:30-11:30

NB! It is not allowed to submit scratch paper along with the answer sheets. If you do submit scratch paper, it will not be evaluated.

PO Box 6050 Langnes, NO-9037 Tromsø / +47 77 64 40 00 / postmottak@uit.no / uit.no

NB: for questions that ask for pseudocode, you can, if you prefer to do so, use languages such as Python, Java, C and Go instead. There is no requirement for remembering exact parameter lists or exact names of API calls, but please indicate important information that must be passed to function calls.

Please make sure you read the questions fully before answering. There may be hints or suggestions that could make it simpler for you to answer.

You are free to answer in Norwegian or English, or a combination of the two.

Good luck!

1) Parallel computers (10%)

a) *What is Amdahl's law?*

b) *What are Flynn's classifications along the instruction and data dimensions? List all four (4) and describe them with a sentence.*

2) Parallel execution (10%)

Given the following code:

```
for (i=2; i<6; i++) {
    x = i - 2 * i + i * i;
    a[i] = a[x];
}
```

Listing 1: Sequential loop.

a) *What are Bernstein's conditions? Give a short description.*

b) *Can you run the iterations in parallel without causing problems? Explain why.*

Hint: Bernstein's conditions are useful for analysing the problem.

3) Collective/group operations (10%)

a) *Explain the term group (or collective) operations.* A short description is enough.

Explain the following operations:

b) Broadcast

c) Reduce

d) Gather

e) Scatter

4) Inspecting a sequential program (15%)

The following two functions are part of a sequential program that computes two things:

1. The Mandelbrot set
2. A simple checksum of the computations

```
int roadMap[HEIGHT][WIDTH];

int solve(double x, double y)
{
    complex z = {0.0, 0.0};
    complex c = {x, y};
    int itt = 0;
    for (itt = 0; (itt < MAX_ITERATIONS) && (complex_magn2(z) <= 4.0); itt++) {
        z = complex_add(complex_squared(z), c);
    }
    return itt;
}

int CreateMap()
{
    int crc = 0;
    //Loops over pixels
    for (int y=0; y<HEIGHT; y++) {
        for (int x=0; x<WIDTH; x++) {
            // Store the number of iterations for this pixel
            int c = solve(translate_x(x), translate_y(y));
            roadMap[y][x] = c;
            crc += c;
        }
    }
    return crc;
}
```

Listing 2: Mandelbrot Code

NB: the functions that are not shown can be ignored as they are not important for the questions in the rest of the exam.

a) *What is dynamic task assignment and static task assignment (or partitioning)?*

b) *When can we expect dynamic task assignment to perform better than static task assignment?*

c) *When can we expect static task assignment to perform better than dynamic task assignment?*

d) *Which task assignment method do you expect to perform better for this program? Why?* NB: you only need to give a short "rule of thumb" explanation, not a deep analysis.

5) MPI (20%)

We are using the code from Listing 2 for this part.

a) *On which type of parallel computing resources or computers do you typically run MPI programs?*

b) *Write a rough outline of an MPI program with a master/worker architecture.* You can use pseudocode and the master and worker functions can be empty (they don't need to communicate with each other) as we're just looking for the structure here.

c) *Write pseudocode and a short explanation of how CreateMap can be parallelized with MPI.* You don't need to write an optimal program: just write a first approach to, for instance, a program with static task assignment. The main points are how you break up the `CreateMap` function (you don't need the entire program), what communication you need and how you use the MPI functions.

6) OpenMP (15%)

We are using the code from Listing 2 for this part.

a) *On which type of parallel computing resources or computers do you typically run OpenMP programs?*

b) *What is a race condition? Can you see a potential for a race condition when parallelizing* `CreateMap`*?*

c) *Parallelize* `CreateMap` *using OpenMP.* You don't need to remember exact syntax as long as you explain what you are doing and why it is safe.

7) CUDA (20%)

We are using the code from Listing 2 for this part.

a) *On which type of parallel computing resources or computers do you typically run CUDA programs?*

b) *What is a kernel?*

c) *How are the terms warps, thread blocks and grids related?*

d) *Write an outline of a typical CUDA program.* You can use pseudocode and the kernel does not have to do any actual work, but write the outline as if you need to run some computations on an array and write back results to a different array. An example could be adding two matrices or vectors ($A = B + C$).

e) *Write a CUDA version of* `CreateMap`. You can ignore the computation of `crc` here (just remove it from the code). You only need to focus on `CreateMap` and how to invoke and use it.

f) *Why is computing* `crc` *a challenge with CUDA?*