UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 8, Examination in Engineering, December 2015

Module Number: EE8246 **Module Name: High Performance Computing**

[Three Hours]

[Answer **all questions**, each question carries equal marks]

Q1. a) Briefly explain *Distribute memory computing.*

[2 marks]

b) State an advantage and a disadvantage of *Distribute memory computing* over *Shared memory computing.*

[2 marks]

c) Message Passing Interface (MPI) is a standardization of a message-passing library interface specification used for distributed memory computing.

Listing 1: MPI program

```
#include <stdio.h>
#include <stdlib.h>
#include "mpi.h"

// N defines how many integers we will send in one MPI message
#define N 2000


int main(int argc, char *argv[]) {
      int rank, size;
      MPI_Status status;
      MPI_Init(&argc, &argv);
      MPI_Comm_size(MPI_COMM_WORLD, &size);
      MPI_Comm_rank(MPI_COMM_WORLD, &rank);

      int *data = malloc(sizeof(int) * N);

      printf("Node %d is trying to send\n", rank);
      MPI_Send(data, N, MPI_INT, (rank + 1) % size,
      1, MPI_COMM_WORLD);
      printf("Node %d sent  successfully\n", rank);
```

```
        MPI_Recv(data, N, MPI_INT, (rank - 1) % size,
        1, MPI_COMM_WORLD,
        &status);
        printf("Node %d received from node \n", rank);

        MPI_Finalize();
        return 0;
}
```

i) Explain the difference between MPI_Ssend and MPI_Bsend using a diagram. [2 mark]

ii) Explain the intended output of the MPI program in Listing 1. [1 mark]

iii) The program in Listing 1 works only with some values of *N*. Explain why the program behaves like that and propose a solution to rectify the error. [3 marks]

Q2. a) Compare Central Processing Unit (CPU) and Graphical Processing Unit (GPU) architectures. [2 marks]

b) Explain the following CUDA functions

i) `cudaMalloc(void ** devPtr, size_t size)`

ii) `cudaMemcpy(void * dst, void * src, size_t count,...)`

[2 marks]

c) Complete the following kernel to add *n* dimensional vectors *a* and *b* into *c* in GPU.

```
__global__ void VectorAdd(int *a, int *b, int *c, int n)
{

}
```

[2 marks]

d) Write a C++ program which runs on CPU to initialize two vectors of integers of size *n*, add them together to a new vector and print it to the console. [2 marks]

e) Write a C++/CUDA program to initialize two vectors of integers of size *n*, add them together to a new vector and print it to the console. Here the addition of two vectors should be done using GPU. [2 marks]

Q3. a) What is the difference between *Thread* and a *Process* in computer programming.

[2 marks]

b) Explain each section of the program in Listing 2

[2 marks]

Listing 2: Thread programing using C#

```
class Program
{
  static void ThreadMethod()
  {
     for (int i = 0; i < 10; i++)
     {
        WriteLine($"Thread functioin on  {i}");
        Thread.Sleep(50);
     }
  }

  static void Main(string[] args)
  {
     Thread t = new Thread(((s) =>
     {
        for (int i = 0; i < (int)s; i++)
        {
            WriteLine($"Thread functioin on  {i}");
            Thread.Sleep(50);
        }
      }));

     t.Start(15);
     Thread.Sleep(100);
     WriteLine("Main thread");
     t.Join();
     WriteLine("thread finished");

  }
}
```

c) What is the output of the program in Listing 2?

[2 marks]

d) Explain asynchronous programming using a diagram?

[2 marks]

e) Method in Listing 3 show C# code for reading a web page for given URL. Convert this to asynchronous method.

[2 marks]

Listing 3: C# code for reading a webpage

```
private static string DownloadPage(string url)
{
    WebRequest requst = WebRequest.Create(url);
    WebResponse response = requst.GetResponse();
    var reader = new StreamReader(response.GetResponseStream());
    return reader.ReadToEnd();
}
```

Q4. a) Explain the following terms in OpenMP.

i) *dynamic* scheduling

ii) *shared* clauses

iii) *private* clause.

iv) *firstprivate* clause.

[2 marks]

b) Explain the race condition that can happen in shared memory programing.

[2 marks]

c) Listing 4 shows a C++ program to add an array.

Listing 4: C++ program to add an array

```
int main()
{
    int A[10000];
    int i;

    for (i = 0; i < 1000; i++)
    {
        A[i] = i*i;
    }


    long sum = 0;

    for (i = 0; i < 1000; i++)
    {
      sum += A[i];
    }
```

```
        printf("sum is %d\n", sum);
        return 0;
    }
```

i) Explain perfectly parallel loop and a reduction loop.

[1 mark]

ii) Parallelize the program in Listing 4 using OpenMP without using reduction clause.

[3 marks]

iii) Parallelize the program in Listing 4 using OpenMP with reduction clause.

[2 marks]

Q5. a) Jacobi iteration method can be used to solve a linear system in the form of $Ax = b$. Jacobi iteration can be given by the following formula.

$$x_i^k = \frac{1}{a_{i,i}} \left[b_i - \sum_{j \neq i} a_{i,j} x_j^{k-1} \right]$$

i) Write a sequential program to solve a linear system of size N using the Jacobi iteration method.

[2 marks]

ii) Parallelize the program using OpenMP.

[3 marks]

b) MPI, the Message Passing Interface, is commonly used for high-performance computing on clusters. One of the standard example programs distributed with MPI concerns the computation of Π from the recurrence relation

$$\int_0^1 \frac{4}{1+x^2} \mathrm{d}x$$

i) Describe a strategy for splitting this problem up into one that can be used with MPI.

[2 marks]

ii) Write an MPI program in commented pseudo-code that implements your approach. Marks will be awarded for the algorithm and the correct use of MPI calls.

[3 marks]