Assignment No: - HPC-Group1-4

TITLE	**Parallel Computing Using CUDA**
PROBLEM STATEMENT / DEFINITION	Write a CUDA Program for : 1. Addition of two large vectors 2. Matrix Multiplication using CUDA C
OBJECTIVE	• Learn parallel decomposition of problem. • Learn parallel computing using CUDA.
S/W PACKAGES AND HARDWARE APPARATUS USED	1. Operating System : 64-bit Open source Linux or its derivative 2. Programming Language: C/C++ 3. NVidea GPU 4. CUDA API
REFERENCES	• Jason sanders, Edward Kandrot, "CUDA by Example", Addison-Wesley, ISBN-13: 978-0-13-138768-3 • Shane Cook, "CUDA Programming: A Developer's Guide to Parallel Computing with GPUs", Morgan Kaufmann Publishers Inc. San Francisco, CA, USA 2013 ISBN: 9780124159884
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	1.Date 2.Assignment no. 3. Problem definition 4. Learning objective 5. Learning outcome 6. Related Mathematics 7. Concepts related Theory 8. Test cases 9. Program code with proper documentation. 10. Output of program. 11. Conclusion and applications (the verification and testing of outcomes)

Assignment No: - HPC-Group1-4

Problem statement: Write a CUDA Program for :
1. Addition of two large vectors
2. Matrix Multiplication using CUDA C

- **Aim:**
  Parallel Computing Using CUDA.

- **Prerequisites**
  C/C++ Programming

- **Learning Objectives**
  - Learn parallel decomposition of problem.
  - Learn parallel computing using CUDA.

- **Learning Outcome:**

Students will be able to decompose problem into sub problems, to learn how to use GPUs, to learn to solve sub problem using threads on GPU cores.

- **Theory**

Dividing a computation into smaller computations and assigning them to different processors for parallel execution are the two key steps in the design of parallel algorithms. The process of dividing a computation into smaller parts, some or all of which may potentially be executed in parallel, is called **decomposition**. Tasks are programmer-defined units of computation into which the main computation is subdivided by means of decomposition. Simultaneous execution of multiple tasks is the key to reducing the time required to solve the entire problem. Tasks can be of arbitrary size, but once defined, they are regarded as indivisible units of computation. The tasks into which a problem is decomposed may not all be of the same size.

In addition of two vectors, we have to add ith element from first array with ith element of second array to get ith element of resultant array. We can allocate this each addition to distinct thread. Same thing can be done for the product of two vecors.

There can be three cases for addition of two vectors using CUDA.

1. n blocks and one thread per block.

2. 1 block and n threads in that block.

3. m blocks and n threads per block.

In addition of two matrices, we have to add (i,j)th element from first matrix with (i,j)th element of second matrix to get (i,j)th element of resultant matrix.. We can allocate this each addition to distinct thread.

There can be two cases for addition of two matrices using CUDA.

1. Two dimensional blocks and one thread per block.

2. One block and two dimensional threads in that block.

Same cases can be considered for multiplication of two matrices.

How to run CUDA Program on Remote Machine

1. Open Terminal

2. Get log in to remote system which has GPU and CUDA installed in it.
e.g. ssh student@10.10.15.21

3. Once you get logged in to system, create a cude file with extension .cu and write code in it.

e.g. cat >> sample.cu
Write code here
Press Ctrl+D to come outside the cat command.

4. Compile CUDA program using nvcc command.
e.g. nvcc sample.cu

5. It will create executable file a.out. Rut it.
e.g. ./a.out

When you are compiling using nvcc command, you may get compiler error “nvcc command not found”

In this case, on remote machine, of which you are using GPU,
you have to run following commands:

Open the terminal and type:
gedit ~/.bashrc

This will open .bashrc for editing.

Note: You have check path of CUDA bin folder. Suppose path is /usr/local/cuda-8.0/bin

Add the following to the end of your .bashrc file.
export PATH="$PATH:/usr/local/cuda-8.0/bin"

This sets your PATH variable to the existing PATH plus what you add to the end.
Run following command to reload the configuration.
source ~/.bashrc

- **Test data:**
  Take vector or matrices with different values of elements.

  - **Conclusion :**
    After successfully completing this assignment, student should be able to understand and implement Addition of two large vectors and Matrix Multiplication operations in CUDAC