**Name: ROMEO SARKAR**

**Adm. No.: 20JE0814**

**Lab Exercise 2.1 Write a PyCUDA program to demonstrate the followings:**

**1. Allocate host and device memories for three matrices A, B, C**

**2. Transfer data of matrices A, B from host to device**

**3. Performance Matrix and Matrix multiplication**

**import pycuda.driver as cuda**

**import pycuda.autoinit**

**from pycuda.compiler import SourceModule**

**import numpy**

**TILE\_WIDTH = 2;**

**MATRIX\_LEN = 8**

**mat\_a = numpy.random.randn(MATRIX\_LEN, MATRIX\_LEN).astype(numpy.float32)**

**mat\_b = numpy.random.randn(MATRIX\_LEN, MATRIX\_LEN).astype(numpy.float32)**

**mat\_c = numpy.empty\_like(mat\_a)**

**dev\_a = cuda.mem\_alloc(mat\_a.nbytes)**

**cuda.memcpy\_htod(dev\_a, mat\_a)**

**dev\_b = cuda.mem\_alloc(mat\_b.nbytes)**

**cuda.memcpy\_htod(dev\_b, mat\_b)**

**dev\_c = cuda.mem\_alloc(mat\_c.nbytes)**

**cuda.memcpy\_htod(dev\_c, mat\_c)**

**source\_module = SourceModule("""**

**\_\_global\_\_ void tiledMatrixMulKernel(float \*mat1, float \*mat2, float \*mat3, int width) {**

**\_\_shared\_\_ float mds[2][2];**

**\_\_shared\_\_ float nds[2][2];**

**int bx = blockIdx.x;**

**int by = blockIdx.y;**

**int tx = threadIdx.x;**

**int ty = threadIdx.y;**

**int row = by \* 2 + ty;**

**int col = bx \* 2 + tx;**

**float pvalue = 0;**

**for(int ph = 0; ph < width / 2; ph++) {**

**mds[ty][tx] = mat1[row \* width + ph \* {0} + tx];**

**nds[ty][tx] = mat2[(ph \* 2 + ty) \* width + col];**

**\_\_syncthreads();**

**for(int k = 0; k < 2; k++) {**

**pvalue += mds[ty][k] \* nds[k][tx];**

**}**

**\_\_syncthreads();**

**}**

**mat3[row \* width + col] = pvalue;**

**}**

**""")**

**tiled\_matrix\_multiplication\_function = source\_module.get\_function("tiledMatrixMulKernel")**

**tiled\_matrix\_multiplication\_function(dev\_a, dev\_b, dev\_c, MATRIX\_LEN, block=(1, 1, 1), grid=(MATRIX\_LEN, MATRIX\_LEN, 1))**

**cuda.memcpy\_dtoh(mat\_c, dev\_c)**

**print("Matrix A:")**

**print(mat\_a)**

**print("Matrix B:")**

**print(mat\_b)**

**print("Product Matrix:")**

**print(mat\_c)**

**docker run --runtime=nvidia -v $HOME:$HOME -ti bryankp/pycuda:latest bash**