****

**Comsats University Islamabad, Lahore campus**

**ASSIGNMENT#3(Lab)**

**RAMSHA KHAN**

**(SP23-BCS-112)**

**Section: C**

**Course: PDC**

**Instructor’s name:Akhzar Nazir**

**Due date: 02-10-25**

import cupy as cp

import numpy as np

import time

N = 1024

THREADS\_PER\_BLOCK = 32

BLOCKS\_N\_32 = (N // THREADS\_PER\_BLOCK, 1, 1)

THREADS\_32 = (THREADS\_PER\_BLOCK, 1, 1)

CUDA\_KERNELS = r'''

extern "C" \_\_global\_\_ void kernel\_indexing(int N) {

    int i = blockIdx.x \* blockDim.x + threadIdx.x;

    if (i == 0 || i == N - 1 || i == N/2) {

        printf("Thread %d / %d | BlockIdx=%d | ThreadIdx=%d\n",

               i, N, blockIdx.x, threadIdx.x);

    }

    if (blockIdx.x == 0 && threadIdx.x == 0) {

        printf("Launch Metadata: GridDim=%d | BlockDim=%d\n",

               gridDim.x, blockDim.x);

    }

}

''';

k\_index = cp.RawKernel(CUDA\_KERNELS, 'kernel\_indexing')

print("Step 1: Memory Allocation and Initialization")

A\_host = np.arange(N, dtype=np.int32)

B\_host = 2 \* np.arange(N, dtype=np.int32)

print("First 5 elements of A:", A\_host[:5])

print("First 5 elements of B:", B\_host[:5])

A\_device = cp.asarray(A\_host)

B\_device = cp.asarray(B\_host)

print("Data copied to GPU")

print("Step 2: Serial Execution on Default Stream")

C\_device = A\_device + B\_device

print("Kernel1 executed: C = A + B")

D\_device = C\_device \* C\_device

print("Kernel2 executed: D = C \* C")

D\_serial = D\_device.get()

print("First 5 elements of D:", D\_serial[:5])

expected = (A\_host[4] + B\_host[4]) \*\* 2

print("Validation check for D[4]: expected", expected, "got", D\_serial[4])

print("Step 3: Parallel Execution with CUDA Streams")

stream1, stream2 = cp.cuda.Stream(), cp.cuda.Stream()

event = cp.cuda.Event()

D\_device.fill(0)

with stream1:

    C\_stream = A\_device + B\_device

    stream1.record(event)

    print("Kernel1 launched on Stream1")

stream2.wait\_event(event)

with stream2:

    D\_stream = C\_stream \* C\_stream

    print("Kernel2 launched on Stream2")

D\_stream\_host = D\_stream.get(stream=stream2)

stream2.synchronize()

print("First 5 elements of streamed D:", D\_stream\_host[:5])

print("Step 4: Synchronization Scenarios")

start\_time = time.time()

D\_device = (A\_device + B\_device) \* (A\_device + B\_device)

cp.cuda.Device(0).synchronize()

sync\_time = time.time() - start\_time

print("With explicit synchronization, CPU blocked for", round(sync\_time, 5), "seconds")

D\_device.fill(0)

D\_device = (A\_device + B\_device) \* (A\_device + B\_device)

time.sleep(0.001)

D\_nosync = D\_device.get()

print("With implicit synchronization, .get() waited automatically")

print("Step 5: Thread Hierarchy Visualization")

print("Configuration 1: <<<1, N>>> (1 block, N threads)")

k\_index((1, 1, 1), (N, 1, 1), (N,))

cp.cuda.Device(0).synchronize()

print("Configuration 2: <<<N/32, 32>>> (multiple blocks)")

k\_index(BLOCKS\_N\_32, THREADS\_32, (N,))

cp.cuda.Device(0).synchronize()

print("Done")

**output:**

