Final Year B.Tech. (CSE) – VII [2024-25]

**6CS452: High Performance Computing Lab**

Assignment No: 11

**Understanding concepts of CUDA Programming**

**PRN:** 21510038  **Name:** Aniket Raju Ghotkar

**Title:** Understanding concepts of CUDA Programming

**Problem Statement 1:**

**Execute the following program and check the properties of your GPGPU.**

**Code:**

# Step 1: Write the CUDA code

%%writefile cuda\_device\_info.cu

#include <stdio.h>

#include <cuda\_runtime.h>

int main()

{

    int deviceCount;

    cudaGetDeviceCount(&deviceCount);

    printf("Checking device count...\n"); // Debug statement

    if (deviceCount == 0)

    {

        printf("There is no device supporting CUDA\n");

        return 0; // Exit the program

    }

    printf("Device Count: %d\n", deviceCount); // Debug statement

    int dev;

    for (dev = 0; dev < deviceCount; ++dev)

    {

        cudaDeviceProp deviceProp;

        cudaGetDeviceProperties(&deviceProp, dev);

        if (dev == 0)

        {

            if (deviceProp.major < 1)

            {

                printf("There is no device supporting CUDA.\n");

            }

            else if (deviceCount == 1)

            {

                printf("There is 1 device supporting CUDA\n");

            }

            else

            {

                printf("There are %d devices supporting CUDA\n", deviceCount);

            }

        }

        printf("  Device %d: \"%s\"\n", dev, deviceProp.name);

        printf("  Major revision number:                         %d\n", deviceProp.major);

        printf("  Minor revision number:                         %d\n", deviceProp.minor);

        printf("  Total amount of global memory:                 %ld bytes\n", deviceProp.totalGlobalMem);

        printf("  Total amount of constant memory:               %ld bytes\n", deviceProp.totalConstMem);

        printf("  Total amount of shared memory per block:       %ld bytes\n", deviceProp.sharedMemPerBlock);

        printf("  Total number of registers available per block: %d\n", deviceProp.regsPerBlock);

        printf("  Warp size:                                     %d\n", deviceProp.warpSize);

        printf("  Multiprocessor count:                          %d\n", deviceProp.multiProcessorCount);

        printf("  Maximum number of threads per block:           %d\n", deviceProp.maxThreadsPerBlock);

        printf("  Maximum sizes of each dimension of a block:    %d x %d x %d\n", deviceProp.maxThreadsDim[0], deviceProp.maxThreadsDim[1], deviceProp.maxThreadsDim[2]);

        printf("  Maximum sizes of each dimension of a grid:     %d x %d x %d\n", deviceProp.maxGridSize[0], deviceProp.maxGridSize[1], deviceProp.maxGridSize[2]);

        printf("  Maximum memory pitch:                          %ld bytes\n", deviceProp.memPitch);

        printf("  Texture alignment:                             %ld bytes\n", deviceProp.textureAlignment);

        printf("  Clock rate:                                    %d kilohertz\n", deviceProp.clockRate);

    }

    return 0;

}

**Ans:**

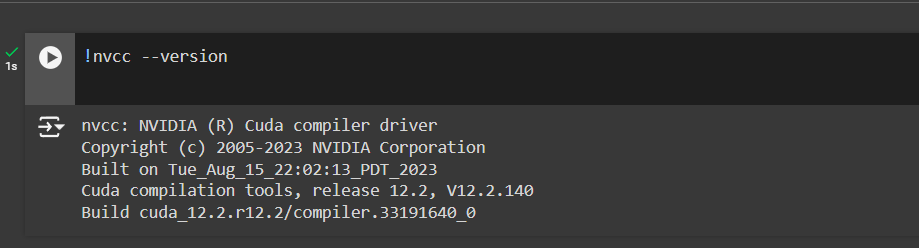
**GPGPU** stands for **"General-Purpose Graphics Processing Unit"**.

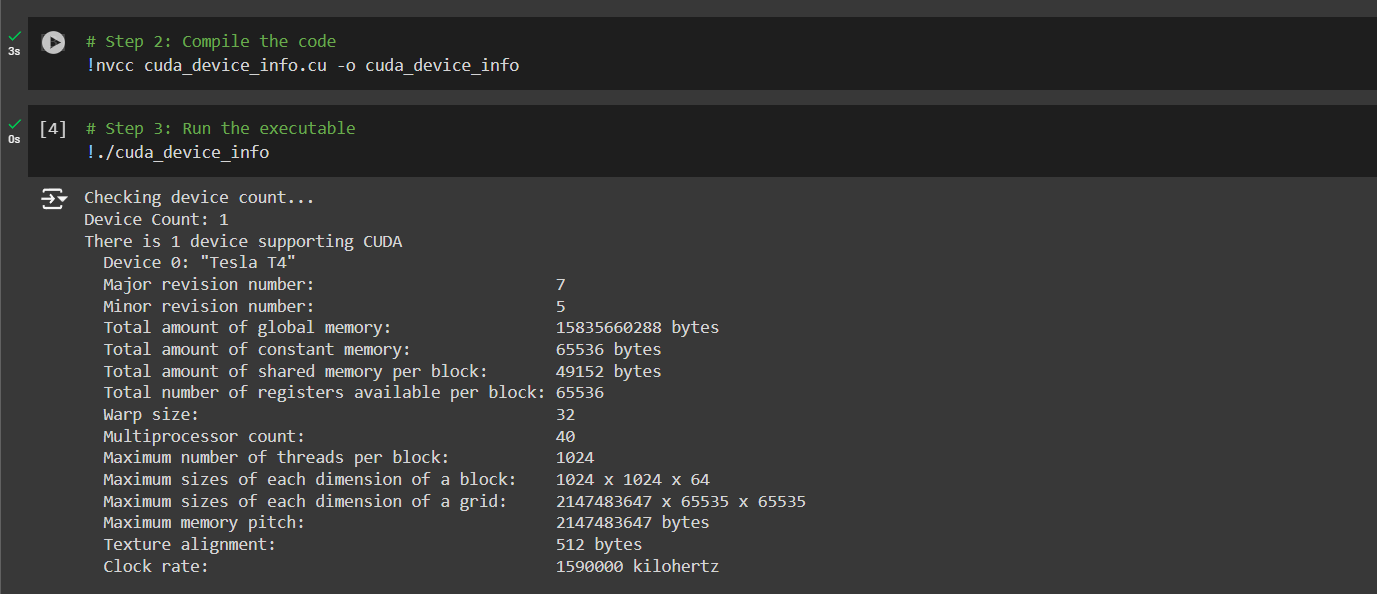
It's a technology that uses a graphics processing unit (GPU) to perform tasks beyond traditional graphics rendering.

GPGPUs are increasingly common and can be used for a variety of purposes, including:

1. Scientific computing
2. Accelerating parts of an application
3. Creating faster, high-performance applications

GPGPUs are efficient parallel processors because of their large number of cores. These cores operate at lower frequencies than CPUs, but are better suited for data that's in graphical form. GPUs can run multiple operations faster than a CPU, making them an ideal choice for many applications.

**Screenshots:**This program queries your system for available CUDA-capable devices using cudaGetDeviceCount() and cudaGetDeviceProperties(). It prints out various properties of each device, such as the number of CUDA cores, memory size, warp size, clock rate, etc.

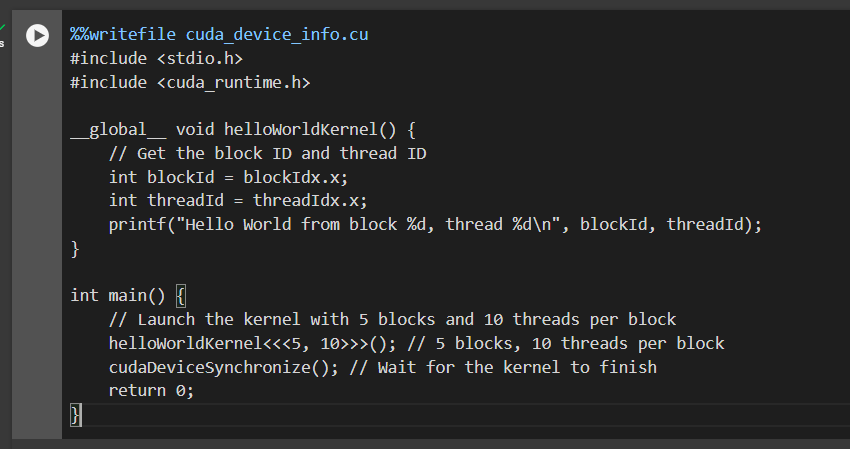
****

**Problem Statement 2:**

**Write a program to where each thread prints its thread ID along with hello world. Lauch the kernel with one block and multiple threads.**

**Ans:**

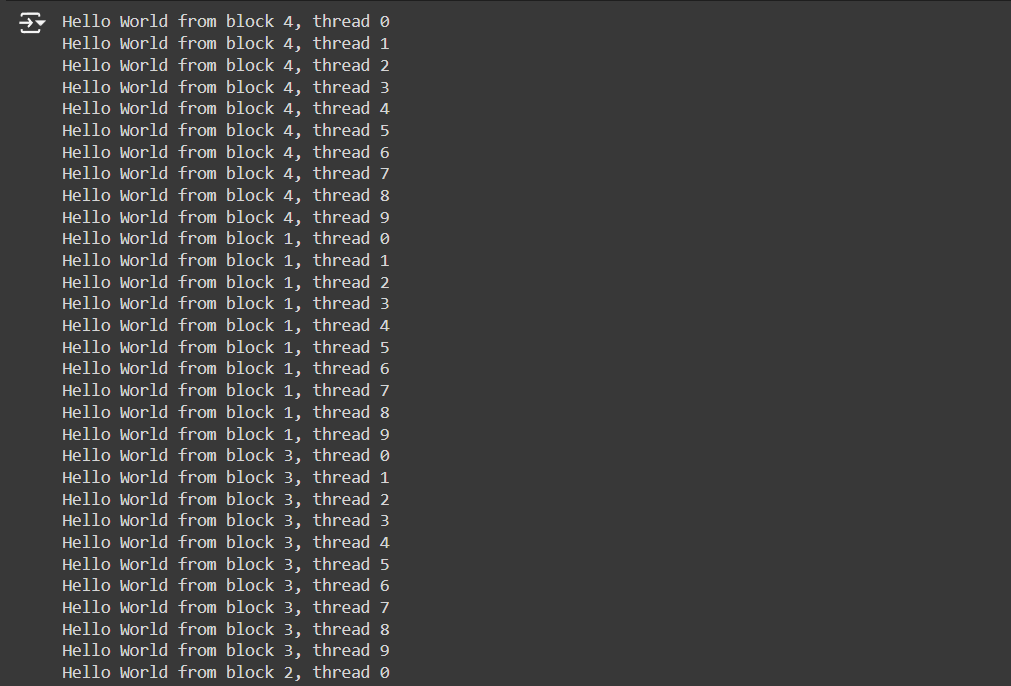
**Code:**

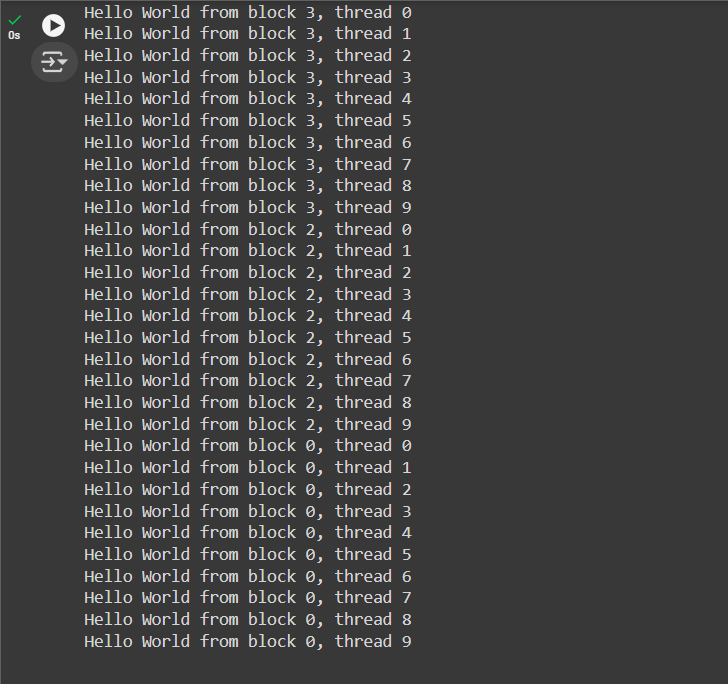


#### Explanation:

* helloWorld is a CUDA kernel where each thread prints its ID along with "Hello World!".
* The kernel is launched with one block (<<<5, 10>>>) and 10 threads.
* **cudaDeviceSynchronize()** ensures that the CPU waits for the GPU to complete execution before exiting.

**Screenshots:**



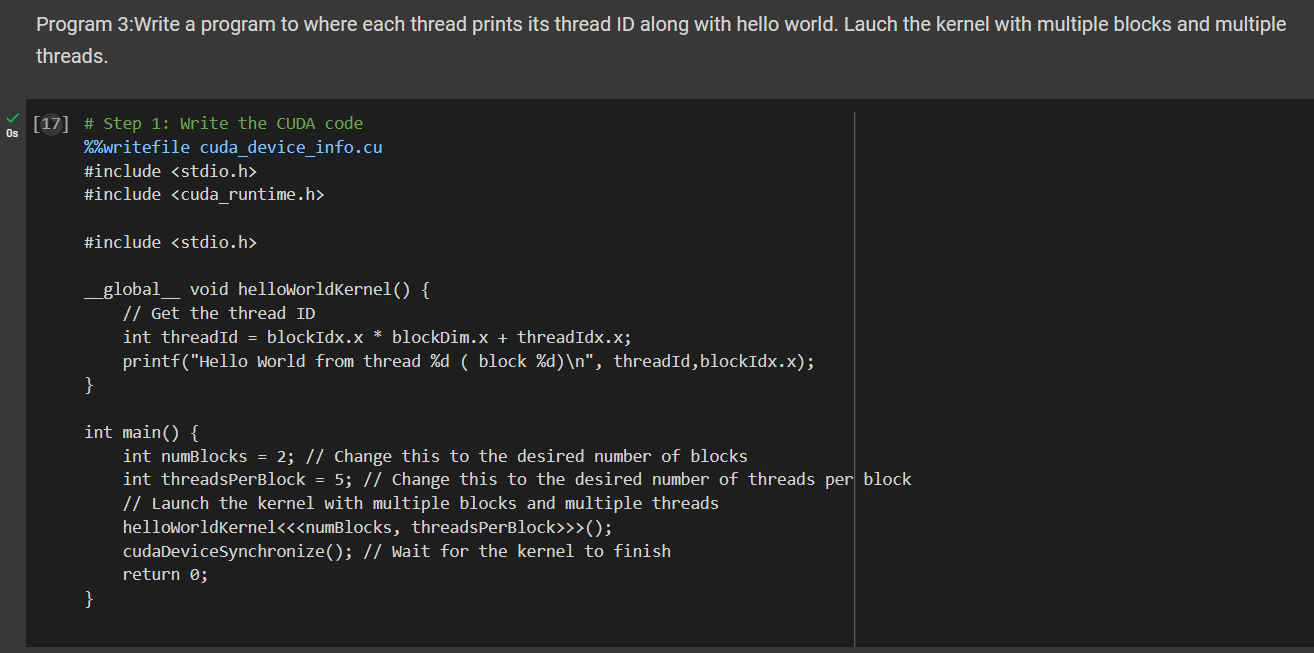


**Problem Statement 3:**

**Write a program to where each thread prints its thread ID along with hello world. Lauch the kernel with multiple blocks and multiple threads.**

**Ans:**

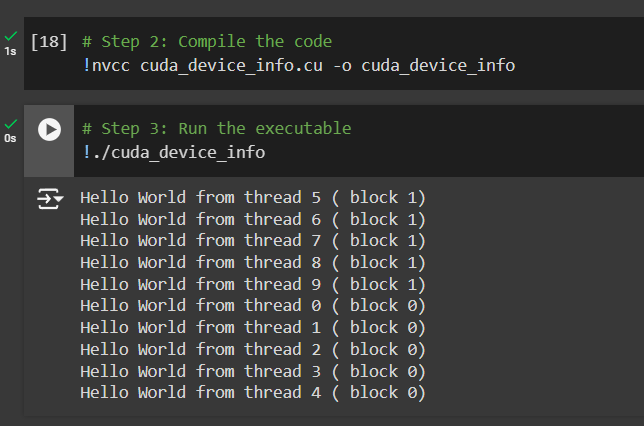
**Code:**

****

#### Explanation:

* The kernel is launched with 2 blocks, each having 5 threads (<<<2, 5>>>).
* Each thread prints its unique ID (calculated as threadIdx.x + blockIdx.x \* blockDim.x) and the block it belongs to.

**Screenshots:**

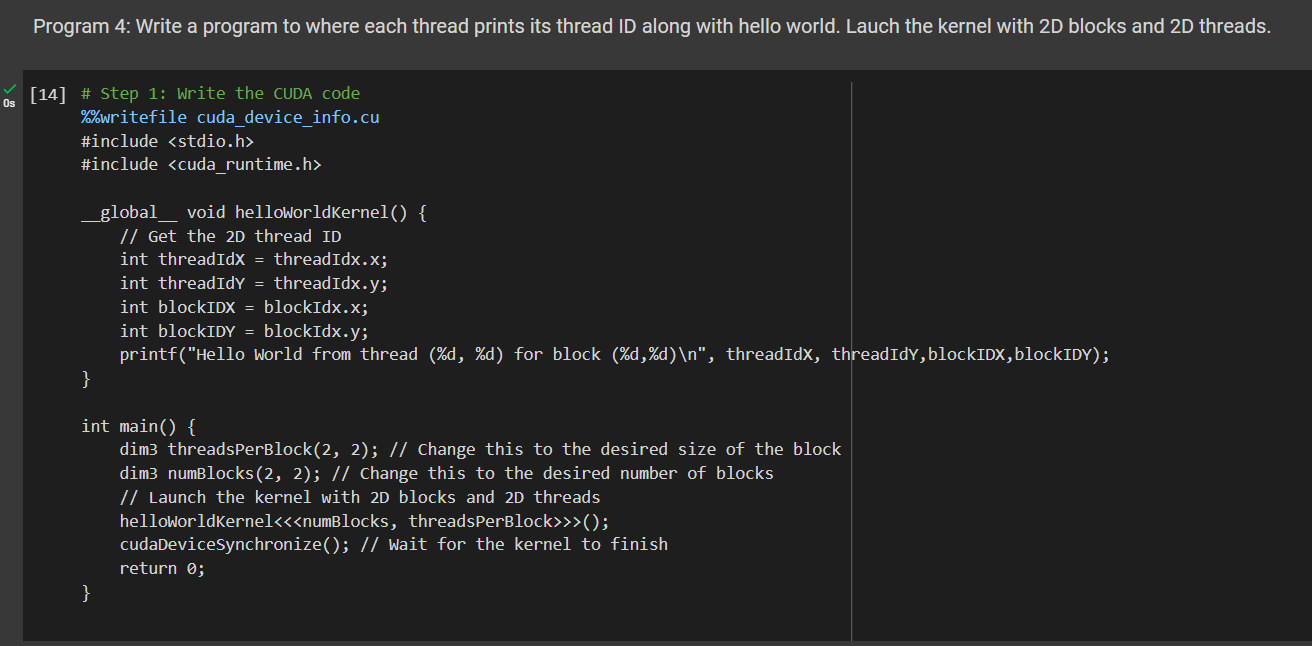
****

**Problem Statement 4:**

**Write a program to where each thread prints its thread ID along with hello world. Lauch the kernel with 2D blocks and 2D threads.**

**Ans:**

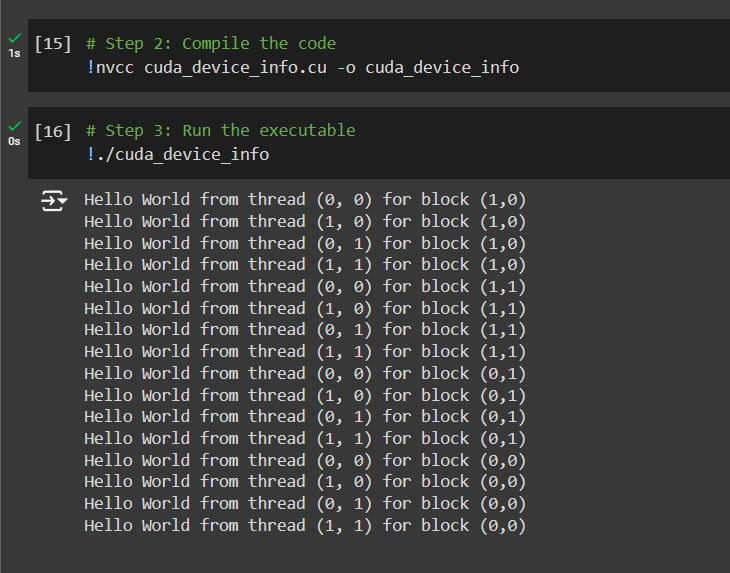
**Code:**

****

**Explaination:**

* This kernel uses 2D blocks and 2D threads to assign unique 2D coordinates (threadIdX, threadIdY) for each thread.
* dim3 grid(2, 2) creates a 2x2 grid of blocks, and dim3 block(4, 4) creates 4x4 threads per block.

**Screenshots:**



**Google Colab Link:**<https://colab.research.google.com/drive/1HwTbpWDO5uGTjT9Pz56dWczE_w8Mp1kh>

**Github Link:**

<https://github.com/AniketGhotkar/HPC_LAB_NEW/tree/main/practical%2011>