Name:

PRN No:

**High Performance Computing Lab**

**Practical No. 11**

**Title of practical: Understanding concepts of CUDA Programming**

**Problem Statement 1:**

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

**#include <stdio.h>**

**#include <stdlib.h>**

**int main()**

**{**

**int deviceCount;**

**cudaGetDeviceCount(&deviceCount);**

**if (deviceCount == 0)**

**{**

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

**}**

**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("\nDevice %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: %d bytes\n", deviceProp.totalGlobalMem);**

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

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

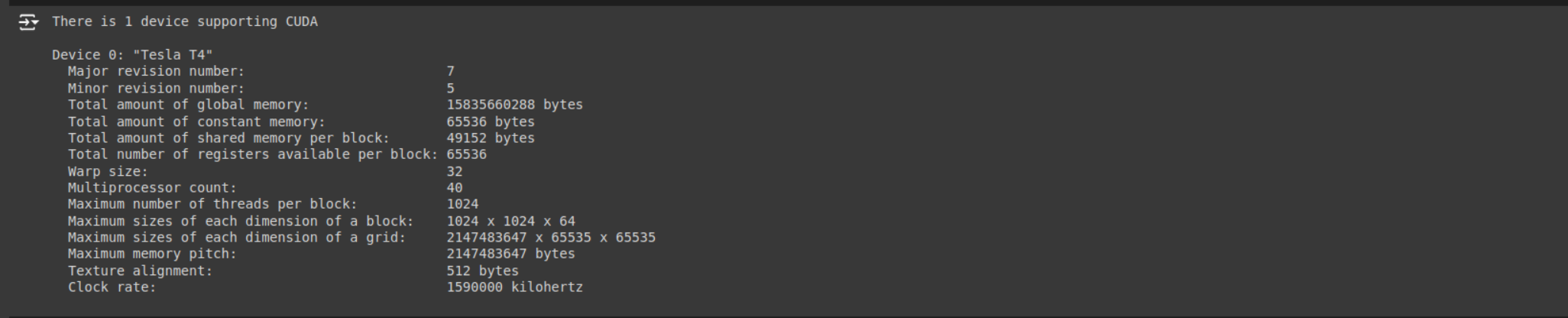
**printf(" Texture alignment: %d bytes\n", deviceProp.textureAlignment);**

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

**}**

**}**

**ScreenShot :**

****

**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.**

**CODE :**

%%writefile hello\_world\_threads.cu

#include <stdio.h>

\_\_global\_\_ void helloWorldFromGPU()

{

int threadID = threadIdx.x;

printf("Hello World from thread ID: %d\n", threadID);

}

int main()

{

int threadsPerBlock = 10; // Number of threads per block

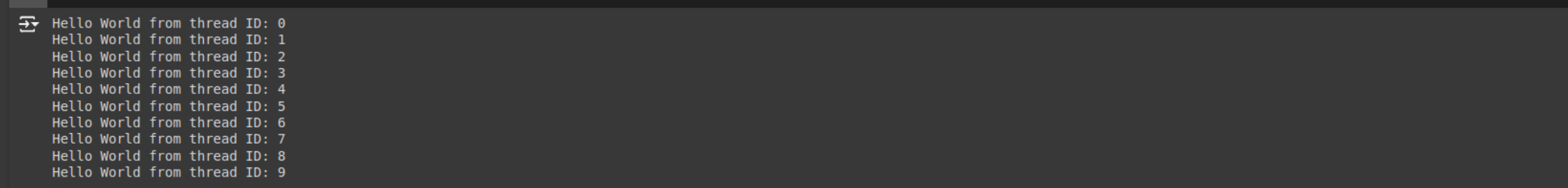
helloWorldFromGPU<<<1, threadsPerBlock>>>();

cudaDeviceSynchronize();

return 0;

}

**ScreenShot :**

****

**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.**

**Code :**

%%writefile hello\_world\_multiple\_blocks.cu

#include <stdio.h>

\_\_global\_\_ void helloWorldFromGPU()

{

int threadID = threadIdx.x;

int blockID = blockIdx.x;

int globalThreadID = blockID \* blockDim.x + threadID;

printf("Hello World from block %d, thread %d (global thread ID: %d)\n", blockID, threadID, globalThreadID);

}

int main()

{

int threadsPerBlock = 5;

int numberOfBlocks = 3;

helloWorldFromGPU<<<numberOfBlocks, threadsPerBlock>>>();

cudaDeviceSynchronize();

return 0;

}

**ScreenShot ;**

****

**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.**

**Code :**

**%%writefile hello\_world\_2D\_blocks\_threads.cu**

#include <stdio.h>

\_\_global\_\_ void helloWorldFromGPU()

{

int threadIDx = threadIdx.x;

int threadIDy = threadIdx.y;

int blockIDx = blockIdx.x;

int blockIDy = blockIdx.y;

int globalThreadIDx = blockIDx \* blockDim.x + threadIDx;

int globalThreadIDy = blockIDy \* blockDim.y + threadIDy;

printf("Hello World from block (%d, %d), thread (%d, %d) (global thread ID: (%d, %d))\n",

blockIDx, blockIDy, threadIDx, threadIDy, globalThreadIDx, globalThreadIDy);

}

int main()

{

dim3 threadsPerBlock(3, 3); // 2D threads (3x3)

dim3 numberOfBlocks(2, 2); // 2D blocks (2x2)

helloWorldFromGPU<<<numberOfBlocks, threadsPerBlock>>>();

cudaDeviceSynchronize();

return 0;

}

**ScreenShot :**

