ELEC 374
Machine Problem 1

Liam Salass
20229595
April 1, 2023

“I do hereby verify that this machine problem submission is my own work and contains my own original ideas, concepts, and designs. No portion of this report or code has been copied in whole or in part from another source, with the possible exception of properly referenced material".

Code:

The code below was used to get the desired output. A function cores, was created to return the number of cores a gpu has depending on it's compute capability major and minor values. This value then would be multiplied by the number of processors it has to print out the number of total CUDA cores the device has.

```
// Liam Salass
// 20229595

#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <stdio.h>

int cores(int major, int minor) {
    //Define GPU arch types using SM version to determine # cores


    switch (major) {
    case 2:
        if (minor == 1) return 48;
        else return 32;
        break;
    case 3:
        return 192;
        break;
    case 5:
        return 128;
        break;
    case 6:
        if ((minor == 1) || (minor == 2)) return 128;
        else if (minor == 0) return 64;
        break;
    case 7:Q
        if ((minor == 0) || (minor == 5)) return 64;
        break;
    }
    printf("Failed to find # cores for Major = %d and Minor = %d \n", major, minor);
    printf("Returned -1\n");
    return -1;
}


int main()
{
    int nDev;
```

```
    //Get count of devices
    cudaGetDeviceCount(&nDev);

    if (nDev == 0){
        printf("No devices");
    }
    else {
        for (int i = 0; i < nDev; i++) {
            //Get information about each cuda dev
            cudaDeviceProp dp;
            cudaGetDeviceProperties(&dp, i);

            //Print information
            printf("Device Number : %d\n", i);
            printf("\tDevice Name: %s\n", dp.name);
            printf("\tClock Rate: %d\n", dp.clockRate);
            printf("\t# Multiprocessors: %d\n", dp.multiProcessorCount);
            printf("\t# CUDA cores: %d\n", dp.multiProcessorCount * cores(dp.major,
dp.minor)); //Multiply number of processors by number of cores
            printf("\tWarp Size: %d\n", dp.warpSize);
            printf("\tGlobal Memory: %ld\n", dp.totalGlobalMem);
            printf("\tConstant Memory: %ld\n", dp.totalConstMem);
            printf("\tShared Memory per block: %ld\n", dp.sharedMemPerBlock);
            printf("\tRegister per block: %d\n", dp.regsPerBlock);
            printf("\tMax threads per block: %d\n", dp.maxThreadsPerBlock);
            printf("\tMax block dimensions: (%d, %d, %d)\n",
                dp.maxThreadsDim[0],
                dp.maxThreadsDim[1],
                dp.maxThreadsDim[2]);
            printf("\tMax Grid dimensions: (%d, %d, %d)\n",
                dp.maxGridSize[0],
                dp.maxGridSize[1],
                dp.maxGridSize[2]);

        }
    }

}
```

Output

The code had the following output in the terminal:

```
C:\Windows\system32\cmd.exe
Device Number : 0
        Device Name: Tesla C2075
        Clock Rate: 1147000
        # Multiprocessors: 14
        # CUDA cores: 448
        Warp Size: 32
        Global Memory: -1
        Constant Memory: 65536
        Shared Memory per block: 49152
        Register per block: 32768
        Max threads per block: 1024
        Max block dimensions: (1024, 1024, 64)
        Max Grid dimensions: (65535, 65535, 65535)
Device Number : 1
        Device Name: Quadro 600
        Clock Rate: 1280000
        # Multiprocessors: 2
        # CUDA cores: 96
        Warp Size: 32
        Global Memory: 1073741824
        Constant Memory: 65536
        Shared Memory per block: 49152
        Register per block: 32768
        Max threads per block: 1024
        Max block dimensions: (1024, 1024, 64)
        Max Grid dimensions: (65535, 65535, 65535)
Press any key to continue . . .
```

An interesting thing to note here is that the first GPU device, the Tesla C2075, failed to return information on its global memory.