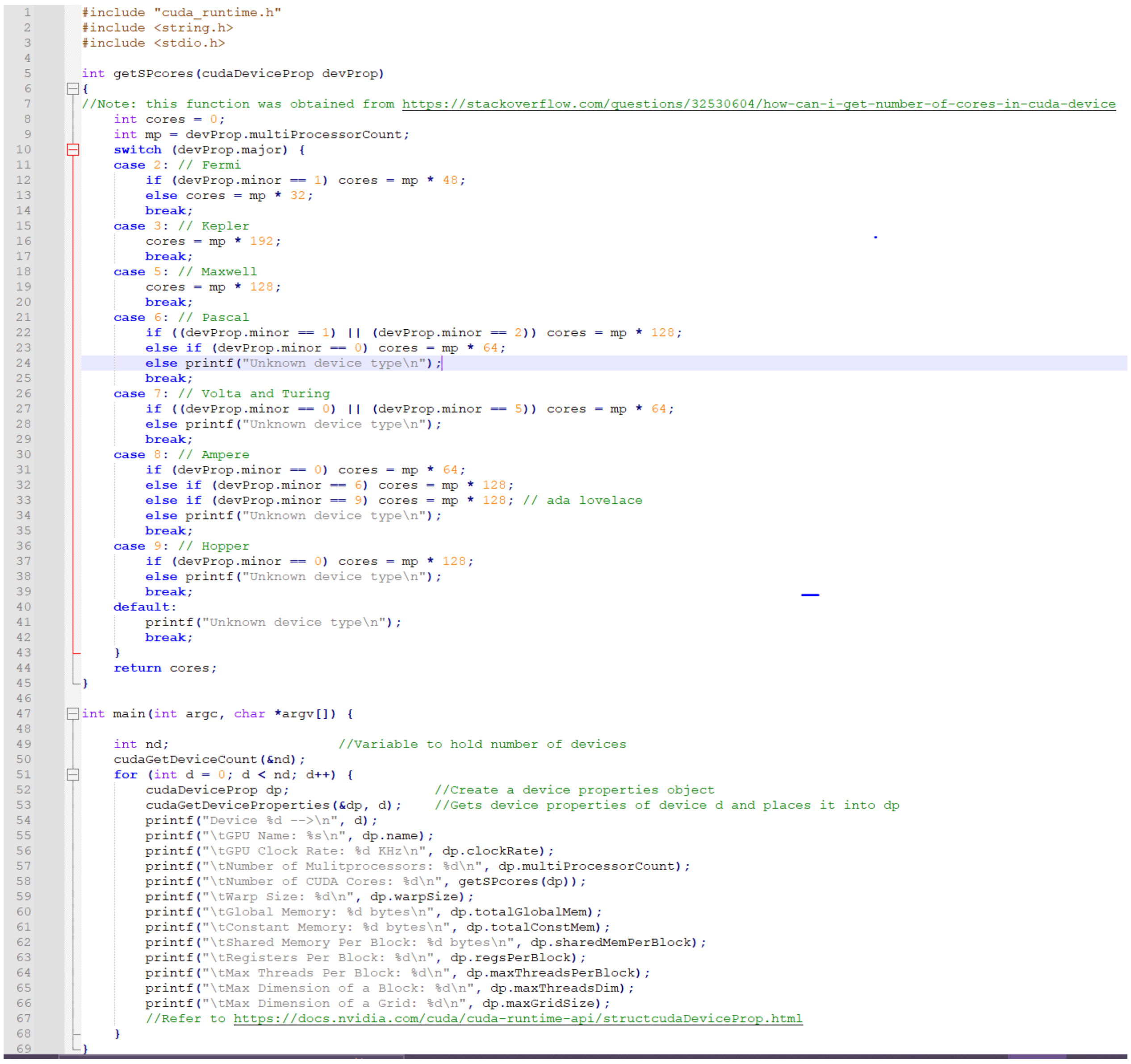
Machine Problem 1 Report

Presented to Prof. Ahmad Afsahi
ELEC 374: Digital Systems Engineering
Faculty of Applied Science
Queen's University

Nathan Goodman: 20228249
Date: April 4th, 2023

CUDA Code

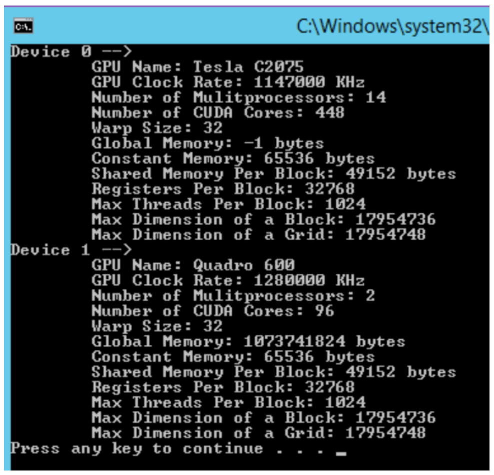


```c
#include "cuda_runtime.h"
#include <string.h>
#include <stdio.h>

int getSPcores(cudaDeviceProp devProp)
{
//Note: this function was obtained from https://stackoverflow.com/questions/32530604/how-can-i-get-number-of-cores-in-cuda-device
    int cores = 0;
    int mp = devProp.multiProcessorCount;
    switch (devProp.major) {
    case 2: // Fermi
        if (devProp.minor == 1) cores = mp * 48;
        else cores = mp * 32;
        break;
    case 3: // Kepler
        cores = mp * 192;
        break;
    case 5: // Maxwell
        cores = mp * 128;
        break;
    case 6: // Pascal
        if ((devProp.minor == 1) || (devProp.minor == 2)) cores = mp * 128;
        else if (devProp.minor == 0) cores = mp * 64;
        else printf("Unknown device type\n");
        break;
    case 7: // Volta and Turing
        if ((devProp.minor == 0) || (devProp.minor == 5)) cores = mp * 64;
        else printf("Unknown device type\n");
        break;
    case 8: // Ampere
        if (devProp.minor == 0) cores = mp * 64;
        else if (devProp.minor == 6) cores = mp * 128;
        else if (devProp.minor == 9) cores = mp * 128; // ada lovelace
        else printf("Unknown device type\n");
        break;
    case 9: // Hopper
        if (devProp.minor == 0) cores = mp * 128;
        else printf("Unknown device type\n");
        break;
    default:
        printf("Unknown device type\n");
        break;
    }
    return cores;
}

int main(int argc, char *argv[]) {

    int nd;                         //Variable to hold number of devices
    cudaGetDeviceCount(&nd);
    for (int d = 0; d < nd; d++) {
        cudaDeviceProp dp;                      //Create a device properties object
        cudaGetDeviceProperties(&dp, d);        //Gets device properties of device d and places it into dp
        printf("Device %d -->\n", d);
        printf("\tGPU Name: %s\n", dp.name);
        printf("\tGPU Clock Rate: %d KHz\n", dp.clockRate);
        printf("\tNumber of Mulitprocessors: %d\n", dp.multiProcessorCount);
        printf("\tNumber of CUDA Cores: %d\n", getSPcores(dp));
        printf("\tWarp Size: %d\n", dp.warpSize);
        printf("\tGlobal Memory: %d bytes\n", dp.totalGlobalMem);
        printf("\tConstant Memory: %d bytes\n", dp.totalConstMem);
        printf("\tShared Memory Per Block: %d bytes\n", dp.sharedMemPerBlock);
        printf("\tRegisters Per Block: %d\n", dp.regsPerBlock);
        printf("\tMax Threads Per Block: %d\n", dp.maxThreadsPerBlock);
        printf("\tMax Dimension of a Block: %d\n", dp.maxThreadsDim);
        printf("\tMax Dimension of a Grid: %d\n", dp.maxGridSize);
        //Refer to https://docs.nvidia.com/cuda/cuda-runtime-api/structcudaDeviceProp.html
    }
}
```



```
Device 0 -->
        GPU Name: Tesla C2075
        GPU Clock Rate: 1147000 KHz
        Number of Mulitprocessors: 14
        Number of CUDA Cores: 448
        Warp Size: 32
        Global Memory: -1 bytes
        Constant Memory: 65536 bytes
        Shared Memory Per Block: 49152 bytes
        Registers Per Block: 32768
        Max Threads Per Block: 1024
        Max Dimension of a Block: 17954736
        Max Dimension of a Grid: 17954748
Device 1 -->
        GPU Name: Quadro 600
        GPU Clock Rate: 1280000 KHz
        Number of Mulitprocessors: 2
        Number of CUDA Cores: 96
        Warp Size: 32
        Global Memory: 1073741824 bytes
        Constant Memory: 65536 bytes
        Shared Memory Per Block: 49152 bytes
        Registers Per Block: 32768
        Max Threads Per Block: 1024
        Max Dimension of a Block: 17954736
        Max Dimension of a Grid: 17954748
Press any key to continue . . . _
```

Description

To retrieve CUDA device properties a series of cudaDeviceProp fiels were referenced including multiProcessorCount, clockRate, and warpSize (among others).

The only manually calculated component was the CUDA cores which was derived from the CUDA device major and minor architecture and multiprocessor count. There are more modern ways of calculating this value such as using the _ConvertSMVer2Cores() function from the library helper_cuda.h. However, this library is unsupported in older versions of CUDA (such as 8.0 this script was coded with). As such, the former method was used.