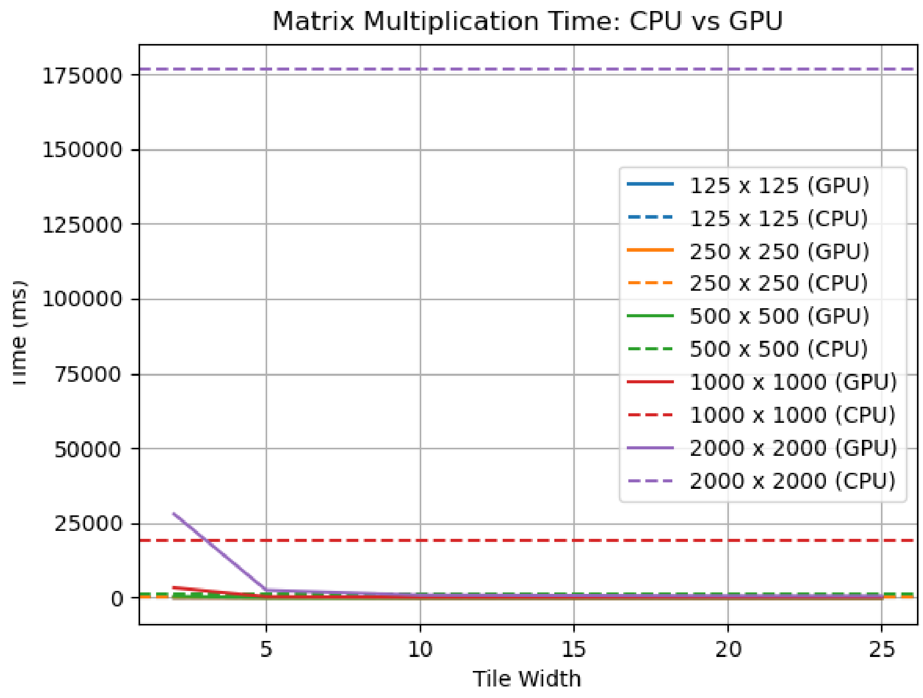
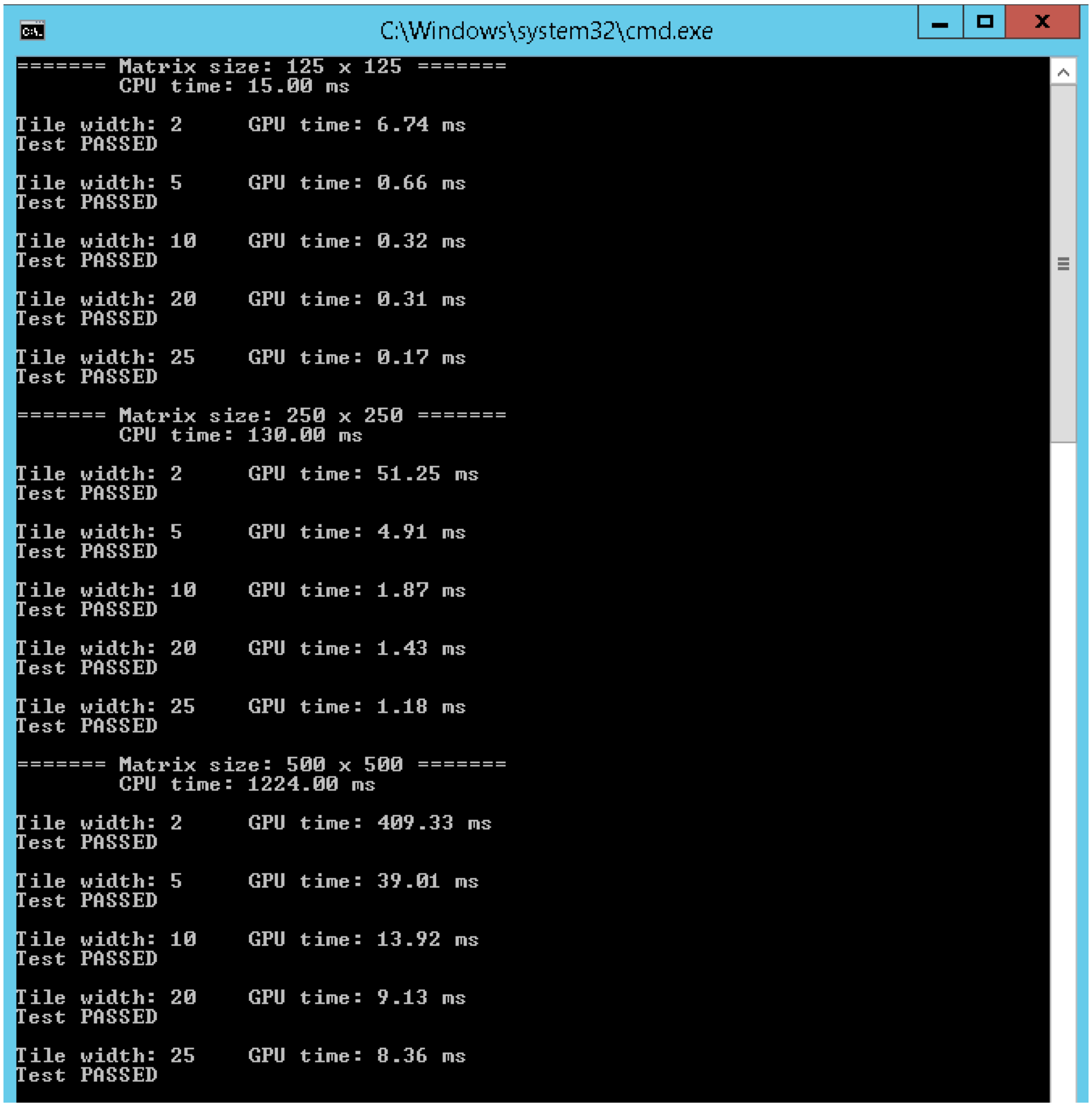
Machine Problem 4

Abdellah Ghassel (20230384)

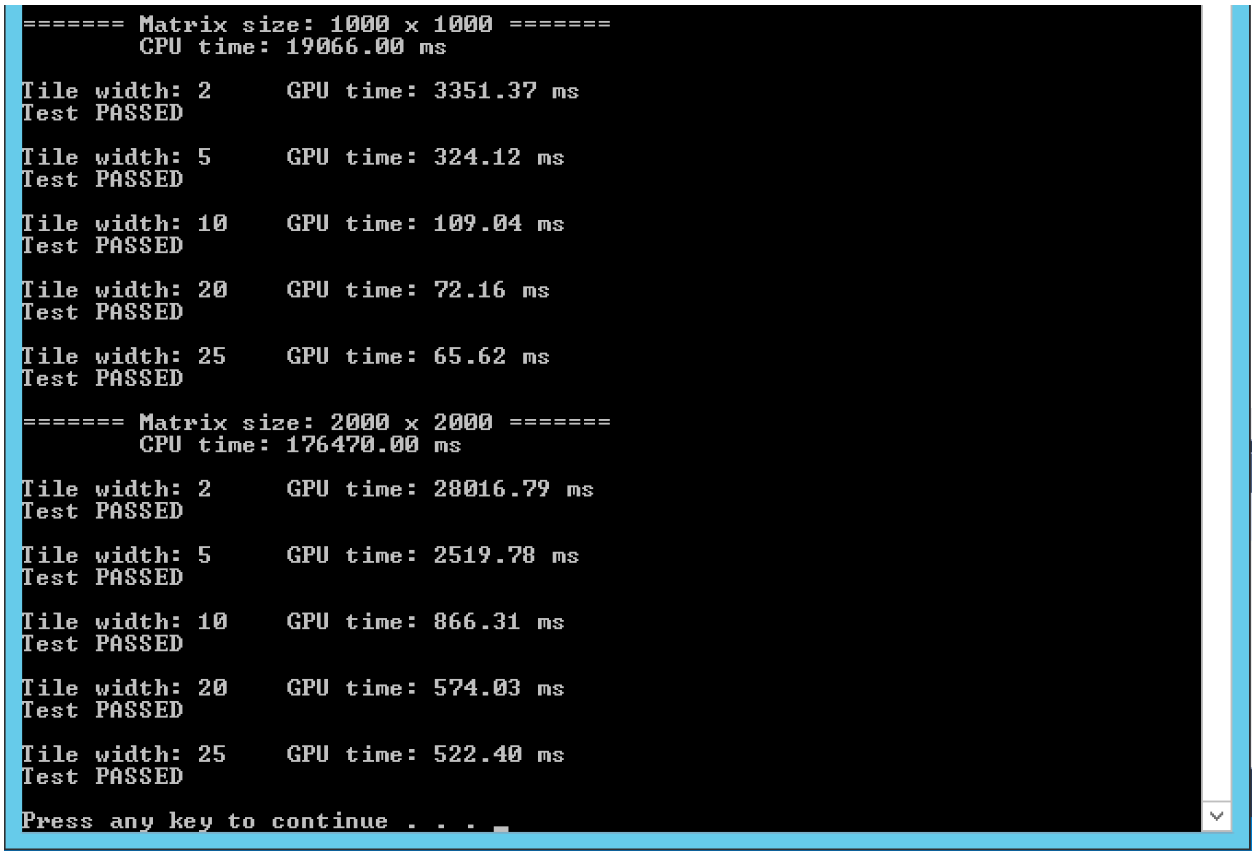
"I do hereby verify that this machine problem submission is my own work and contains my own original ideas, concepts, and designs. No portion of this report or code has been copied in whole or in part from another source, with the possible exception of properly referenced material".



Terminal Output


C:\Windows\system32\cmd.exe

```
======= Matrix size: 125 x 125 =======
        CPU time: 15.00 ms

Tile width: 2      GPU time: 6.74 ms
Test PASSED

Tile width: 5      GPU time: 0.66 ms
Test PASSED

Tile width: 10     GPU time: 0.32 ms
Test PASSED

Tile width: 20     GPU time: 0.31 ms
Test PASSED

Tile width: 25     GPU time: 0.17 ms
Test PASSED

======= Matrix size: 250 x 250 =======
        CPU time: 130.00 ms

Tile width: 2      GPU time: 51.25 ms
Test PASSED

Tile width: 5      GPU time: 4.91 ms
Test PASSED

Tile width: 10     GPU time: 1.87 ms
Test PASSED

Tile width: 20     GPU time: 1.43 ms
Test PASSED

Tile width: 25     GPU time: 1.18 ms
Test PASSED

======= Matrix size: 500 x 500 =======
        CPU time: 1224.00 ms

Tile width: 2      GPU time: 409.33 ms
Test PASSED

Tile width: 5      GPU time: 39.01 ms
Test PASSED

Tile width: 10     GPU time: 13.92 ms
Test PASSED

Tile width: 20     GPU time: 9.13 ms
Test PASSED

Tile width: 25     GPU time: 8.36 ms
Test PASSED
```



```
======= Matrix size: 1000 x 1000 =======
        CPU time: 19066.00 ms

Tile width: 2      GPU time: 3351.37 ms
Test PASSED

Tile width: 5      GPU time: 324.12 ms
Test PASSED

Tile width: 10     GPU time: 109.04 ms
Test PASSED

Tile width: 20     GPU time: 72.16 ms
Test PASSED

Tile width: 25     GPU time: 65.62 ms
Test PASSED

======= Matrix size: 2000 x 2000 =======
        CPU time: 176470.00 ms

Tile width: 2      GPU time: 28016.79 ms
Test PASSED

Tile width: 5      GPU time: 2519.78 ms
Test PASSED

Tile width: 10     GPU time: 866.31 ms
Test PASSED

Tile width: 20     GPU time: 574.03 ms
Test PASSED

Tile width: 25     GPU time: 522.40 ms
Test PASSED

Press any key to continue . . .
```

With a tile width of 25, this code outperforms all the GPU block sizes from MP3.

Questions

Note: the questions will be answered based on the properties from MP1:

```
Name: Tesla C2075
Clock rate: 1147000
Number of SMs: 14
Number of cores: 448
Warp size: 32
Global memory: 4294967295 bytes
Constant memory: 65536 bytes
Shared memory per block: 49152 bytes
Registers per block: 32768
Max threads per block: 1024
Max block size dimensions: (1024, 1024, 64)
Max grid size dimensions: (65535, 65535, 65535)
```

Question 1

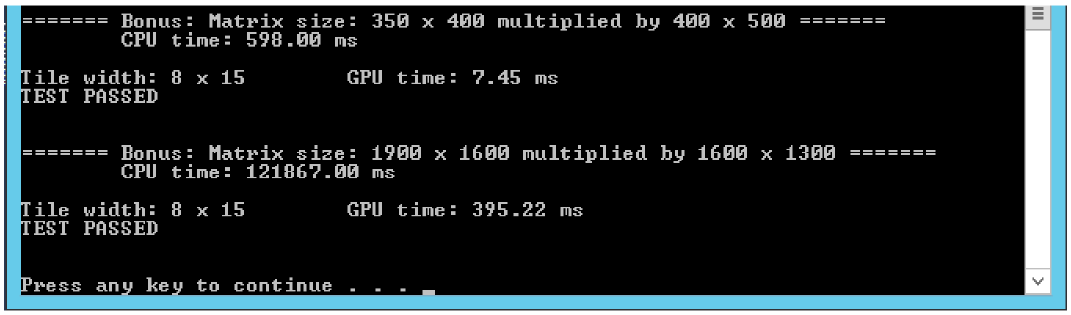
Each streaming multiprocessor (SM) on the Tesla C2075 device can schedule up to 2048 threads (max threads per block * max blocks per SM), so maximum number of threads that can be scheduled is 14 SMs * 2048 threads/SM = **28672 threads**.

Question 2

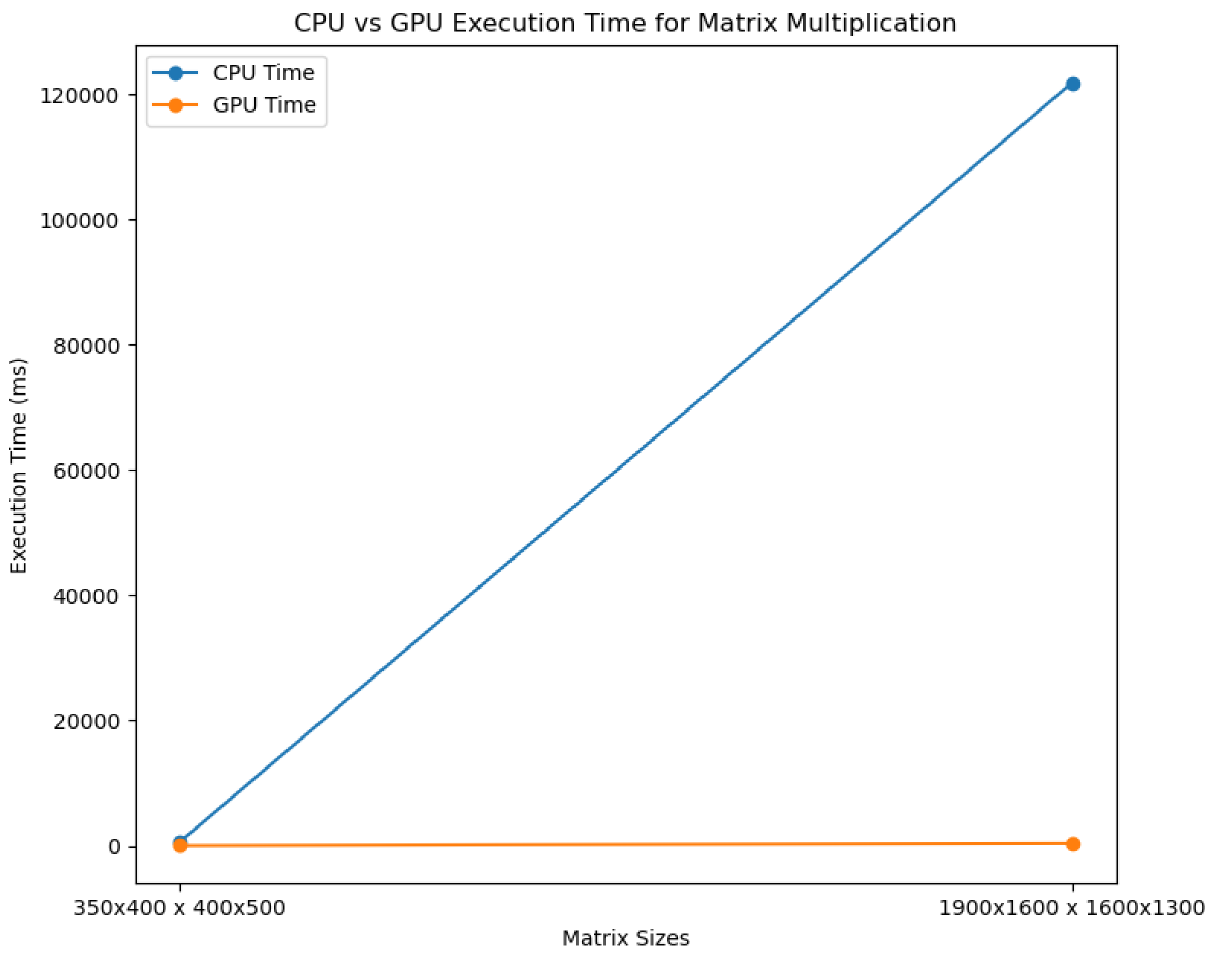
- **Number of registers per thread:** 32768 registers per block
- **Shared memory size:** 49152 bytes
- **Number of blocks per streaming multiprocessor:** 16
    - shared memory per SM divided by shared memory per block
- **Maximum total threads simultaneously scheduled/executing:** 229376 threads
    - 14 SMs * 16 blocks/SM * 1024 threads/block

Bonus: Rectangular Matrices

In order to perform rectangular matrix multiplication, the CPU and GPU matrix multiplication function were modified to take in the matrix height and width.



```
======= Bonus: Matrix size: 350 x 400 multiplied by 400 x 500 =======
        CPU time: 598.00 ms

Tile width: 8 x 15        GPU time: 7.45 ms
TEST PASSED


======= Bonus: Matrix size: 1900 x 1600 multiplied by 1600 x 1300 =======
        CPU time: 121867.00 ms

Tile width: 8 x 15        GPU time: 395.22 ms
TEST PASSED


Press any key to continue . . .
```



These plots demonstrate the drastic difference in computational performance between GPU and the CPU. This plots lacks some significance since different rectangular tile widths were not experimented with more additional matrix sizes.

Appendix

Python Code:

```
import re
import matplotlib.pyplot as plt

data = '''
======= Matrix size: 125 x 125 =======
        CPU time: 15.00 ms

Tile width: 2     GPU time: 6.74 ms
Test PASSED
```

```
Tile width: 5     GPU time: 0.66 ms
Test PASSED

Tile width: 10    GPU time: 0.32 ms
Test PASSED

Tile width: 20    GPU time: 0.31 ms
Test PASSED

Tile width: 25    GPU time: 0.17 ms
Test PASSED

======= Matrix size: 250 x 250 =======
        CPU time: 130.00 ms

Tile width: 2     GPU time: 51.25 ms
Test PASSED

Tile width: 5     GPU time: 4.91 ms
Test PASSED

Tile width: 10    GPU time: 1.87 ms
Test PASSED

Tile width: 20    GPU time: 1.43 ms
Test PASSED

Tile width: 25    GPU time: 1.18 ms
Test PASSED

======= Matrix size: 500 x 500 =======
        CPU time: 1224.00 ms

Tile width: 2     GPU time: 409.33 ms
Test PASSED

Tile width: 5     GPU time: 39.01 ms
Test PASSED

Tile width: 10    GPU time: 13.92 ms
Test PASSED

Tile width: 20    GPU time: 9.13 ms
Test PASSED

Tile width: 25    GPU time: 8.36 ms
Test PASSED

======= Matrix size: 1000 x 1000 =======
        CPU time: 19066.00 ms

Tile width: 2     GPU time: 3351.37 ms
Test PASSED

Tile width: 5     GPU time: 324.12 ms
Test PASSED

Tile width: 10    GPU time: 109.04 ms
Test PASSED

Tile width: 20    GPU time: 72.16 ms
Test PASSED

Tile width: 25    GPU time: 65.62 ms
Test PASSED

======= Matrix size: 2000 x 2000 =======
        CPU time: 176470.00 ms
```

```
Tile width: 2     GPU time: 28016.79 ms
Test PASSED

Tile width: 5     GPU time: 2519.78 ms
Test PASSED

Tile width: 10    GPU time: 866.31 ms
Test PASSED

Tile width: 20    GPU time: 574.03 ms
Test PASSED

Tile width: 25    GPU time: 522.40 ms
Test PASSED
'''

matrix_sizes = []
cpu_times = []
gpu_times = []

for m in re.finditer(r'Matrix size: (\d+) x (\d+)', data):
    matrix_sizes.append(int(m.group(1)))

for cpu_time in re.findall(r'CPU time: ([\d.]+) ms', data):
    cpu_times.append(float(cpu_time))

for gpu_time in re.findall(r'GPU time: ([\d.]+) ms', data):
    gpu_times.append(float(gpu_time))

tile_widths = [2, 5, 10, 20, 25]

plt.figure()

for i, size in enumerate(matrix_sizes):
    plt.plot(tile_widths, gpu_times[i * len(tile_widths): (i + 1) * len(tile_widths)], label=f'{size} x {size} (GPU)')
    plt.axhline(cpu_times[i], color='C{}'.format(i), linestyle='--', label=f'{size} x {size} (CPU)')

plt.xlabel('Tile Width')
plt.ylabel('Time (ms)')
plt.title('Matrix Multiplication Time: CPU vs GPU')
plt.legend()
plt.grid()
plt.show()
```

Bonus Plot Python Code

```
import matplotlib.pyplot as plt

# Matrix sizes
matrix_sizes = ['350x400 x 400x500', '1900x1600 x 1600x1300']

# CPU times in milliseconds
cpu_times = [598.00, 121867.00]

# GPU times in milliseconds
gpu_times = [7.45, 395.22]

# Plot CPU and GPU times
plt.plot(matrix_sizes, cpu_times, label='CPU Time', marker='o', linestyle='-')
plt.plot(matrix_sizes, gpu_times, label='GPU Time', marker='o', linestyle='-')

# Add labels and legend
```

```
plt.xlabel('Matrix Sizes')
plt.ylabel('Execution Time (ms)')
plt.title('CPU vs GPU Execution Time for Matrix Multiplication')
plt.legend()

# Show the plot
plt.show()
```