

Lab 6: Graphics Processing Unit CME433-01

Addi Amaya Caa746 11255790

Dec 6th, 2021

Contents

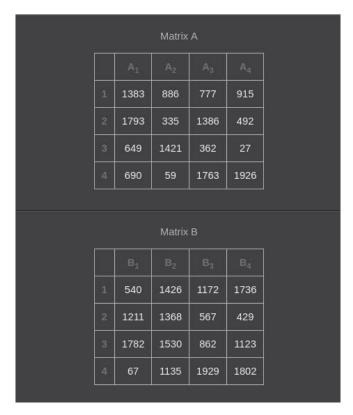
| Single-Line Cache | Error! Bookmark not defined. |
|-----------------------|------------------------------|
| Microprocessor Layout | Error! Bookmark not defined. |
| My Program Sequencer | Error! Bookmark not defined. |
| Questions | Error! Bookmark not defined. |
| Question 1 | Error! Bookmark not defined. |
| Question 2 | Error! Bookmark not defined. |
| Question 3 | Error! Bookmark not defined. |
| Question 4 | Error! Bookmark not defined. |
| Cache Memory Benefits | Error! Bookmark not defined. |
| My Program Sequencer | Error! Bookmark not defined. |
| Questions | Error! Bookmark not defined. |
| Question 1 | Error! Bookmark not defined. |
| Question 2 | Error! Bookmark not defined. |
| Question 3 | Error! Bookmark not defined. |

Step 1: matrixMul.c

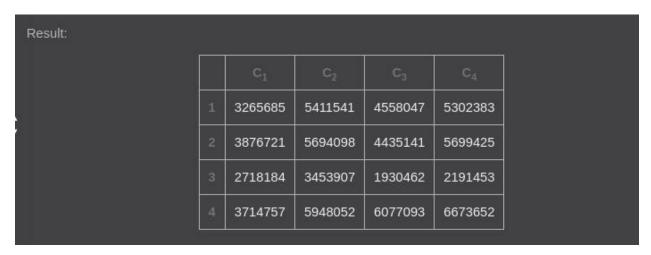
This will outline the proof that confirms step one is working as intended

Expected Output

The below photo shows in the input to the matrix calculator



The below image shows the output to the matrix multiplication between matrix A and matrix B.



Actual Output

After finishing matrixMul.c and compiling the code. The below picture shows the outcome with the same inputs.

```
[caa746@engr-elec70-09L files]$ ./matrixMul
matrixA =
1383
       886
               777
                       915
1793
       335
               1386
                       492
649
      1421
               362
                       27
690
      59
               1763
                       1926
matrixB =
540
      1426
               1172
                       1736
1211
      1368
               567
                       429
1782
       1530
               862
                       1123
67
       1135
               1929
                       1802
matrixC=
3265685 5411541 4558047 5302383
3876721 5694098 4435141 5699425
2718184 3453907 1930462 2191453
3714757 5948052 6077093 6673652
Dimension of matrixA: 4 x 4
Dimension of matrixB: 4 x 4
Multiplication of matrixA and matrixB need 1.000000 ms
```

Table of Matrixes

| Matrix Size | Time |
|-------------|----------|
| 3x3 | 1ms |
| 10x10 | 5ms |
| 123x123 | 8077ms |
| 210x210 | 40031ms |
| 421x421 | 323071ms |
| 512x512 | 579135ms |

Conclusion

The matrixMul.c does work and its quite slow when it hits 512x512

Step 2: matrixMul host.c and matrixMul kernel.cl

The below photo shows the outcome of a 4x4 matrix to confirm that the outcome is the same as in step 1

```
Device: Intel(R) UHD Graphics 630 [0x9bc5]
Running matrix multiplication for matrices A (4x4) and B (4x4) ...
Matrix multiplication completed...
Execution time in milliseconds = 0.014 ms
matrixA =
1383
        886
                777
                        915
1793
        335
                1386
                        492
649
        1421
                        27
                362
690
        59
                1763
                        1926
matrixB =
540
        1426
                1172
                        1736
1211
        1368
                567
                        429
1782
        1530
                862
                        1123
67
        1135
                1929
                        1802
matrixC =
3265685 5411541 4558047 5302383
3876721 5694098 4435141 5699425
2718184 3453907 1930462 2191453
3714757 5948052 6077093 6673652
[caa746@engr-elec70-04L files]$
```

Table of Matrixes

| Matrix Size | Time |
|-------------|----------|
| 3x3 | 0.3ms |
| 10x10 | 0.018ms |
| 123x123 | 0.755ms |
| 210x210 | 2.895ms |
| 421x421 | 84.242ms |
| 512x512 | 42.656ms |

Conclusion

The parallel processing is a magnitude faster than the single thread processing. Using the GPU implementation shows the value of using processing on the GPU and the speed that it can save you. OpenCL implementation on the GPU offers faster processing than the just a single core CPU. There were some strange instances in which bigger matrices results in a faster time. Additionally, Theses calculations were only implemented with square by square matrix multiplication