Matrix Multiplication

This document will serve to describe implementation, usage and more.

Documentation Contents	General Information
Summary 1. **Project Introduction** 2. **Summary** 3. **Usage Instructions** 4. **Results** 5. **Conclusion**	Student: **Franklin Henry Boswell** Github: **fhboswell** Class: **CSC 510 SFSU** Professor: **DR. Singh**

Project Introduction

For a final project in CSC 510 I implemented several matrix multiplication algorithms. The algorithms were first implemented to operate on a single thread. Next they were implemented with certain parts distributed to multiple threads. All the implementations were benchmarked and the results were largely as expected.

Summary

All algorithms were tested by compiling using Xcode(clang) and gcc. The results were very different. The flags -O2 and -pthread were used. The flag -O2 tells both compilers to use a certain set of predetermined optimization suites. In certain cases these -O2 optimizations can involve the compiler using multithreading on loops or other groups of instructions that it things can be done in parallel. This explains some of the results that I didn't expect.
Usage Instructions

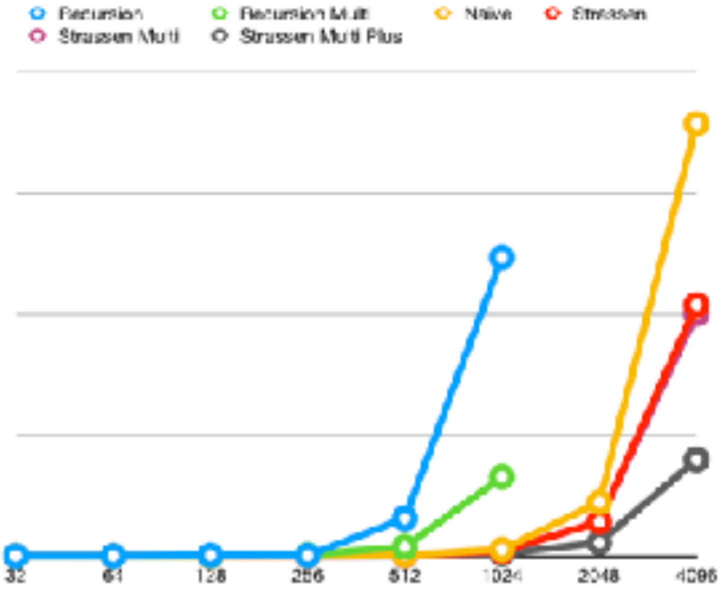
• To run the project in Xcode :

- open the project
- hit the play button in the top left
- Observe the output at the bottom of the editor
- It will look like the right column

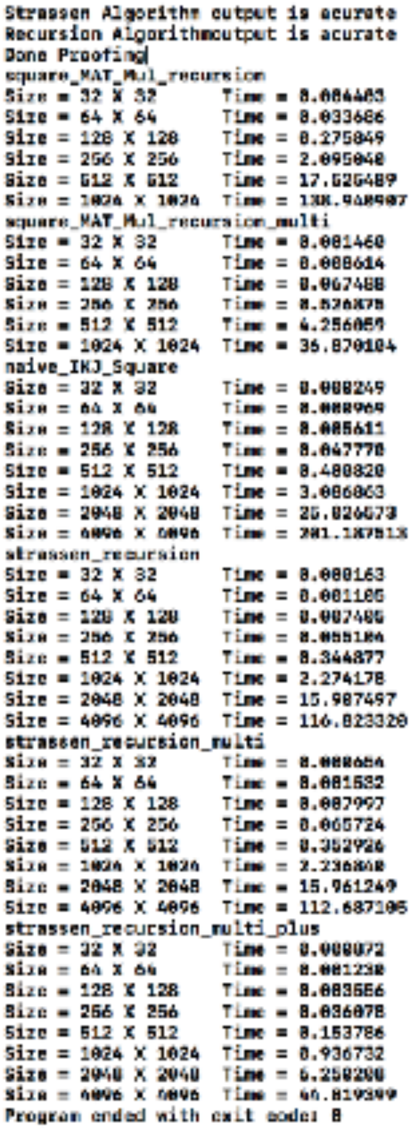
• To run the project from the command line:

- Change directory to the directory with the source code
- Use the command : gcc -O2 main.c -pthread
- To run use the command ./a.out

Findings Clang



X = size of matrix
Y = time in seconds



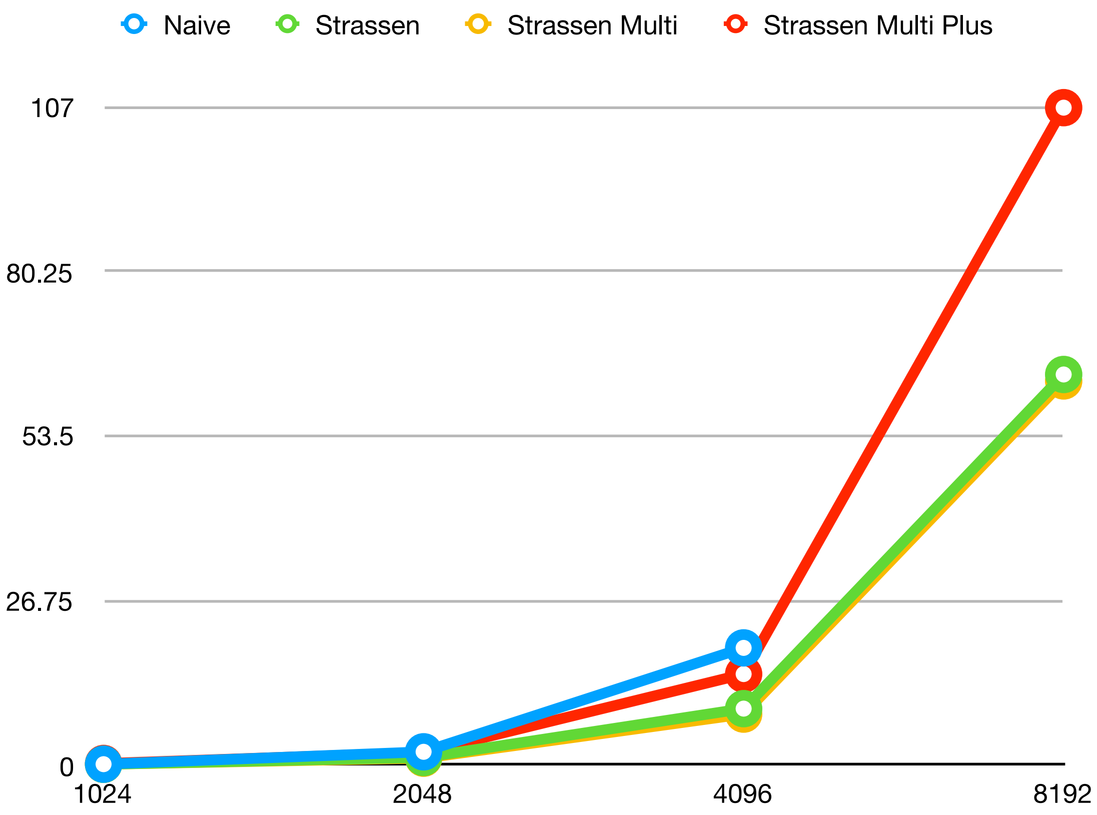
```
Strassen Algorithm output is acurate
Recursion Algorithmoutput is acurate
Done Proofing
square_MAT_Mul_recursion
Size = 32 X 32         Time = 0.004403
Size = 64 X 64         Time = 0.033686
Size = 128 X 128       Time = 0.275849
Size = 256 X 256       Time = 2.095048
Size = 512 X 512       Time = 17.626489
Size = 1024 X 1024     Time = 138.948907
square_MAT_Mul_recursion_multi
Size = 32 X 32         Time = 0.001460
Size = 64 X 64         Time = 0.008614
Size = 128 X 128       Time = 0.067488
Size = 256 X 256       Time = 0.526878
Size = 512 X 512       Time = 4.256059
Size = 1024 X 1024     Time = 36.870104
naive_IKJ_Square
Size = 32 X 32         Time = 0.000249
Size = 64 X 64         Time = 0.000969
Size = 128 X 128       Time = 0.005611
Size = 256 X 256       Time = 0.047778
Size = 512 X 512       Time = 0.480820
Size = 1024 X 1024     Time = 3.086063
Size = 2048 X 2048     Time = 25.026673
Size = 4096 X 4096     Time = 201.187513
strassen_recursion
Size = 32 X 32         Time = 0.000163
Size = 64 X 64         Time = 0.001185
Size = 128 X 128       Time = 0.007485
Size = 256 X 256       Time = 0.055184
Size = 512 X 512       Time = 0.344877
Size = 1024 X 1024     Time = 2.274178
Size = 2048 X 2048     Time = 15.987497
Size = 4096 X 4096     Time = 116.823320
strassen_recursion_multi
Size = 32 X 32         Time = 0.000654
Size = 64 X 64         Time = 0.001532
Size = 128 X 128       Time = 0.007997
Size = 256 X 256       Time = 0.065724
Size = 512 X 512       Time = 0.352926
Size = 1024 X 1024     Time = 2.236868
Size = 2048 X 2048     Time = 15.961249
Size = 4096 X 4096     Time = 112.687106
strassen_recursion_multi_plus
Size = 32 X 32         Time = 0.000072
Size = 64 X 64         Time = 0.001238
Size = 128 X 128       Time = 0.003556
Size = 256 X 256       Time = 0.036078
Size = 512 X 512       Time = 0.153786
Size = 1024 X 1024     Time = 0.936732
Size = 2048 X 2048     Time = 6.258288
Size = 4096 X 4096     Time = 46.819399
Program ended with exit code: 0
```

Conclusion

These findings are consistent with what we would inspect, however. As the graph shoes the stressed multi threaded algorithm(purple) is only a very small amount faster than Strassen's algorithm implemented without multithreading. I believe this is because the -O2 flag generates multithreaded assembler for the non-multithreaded algorithm.

Furthermore the reason for the significant slowness of the divide and conquer recursive algorithm both threaded and not is thrashing (too many processes spending too much time competing for processor use rather than actually computing) and excess memory allocations in the implementation.

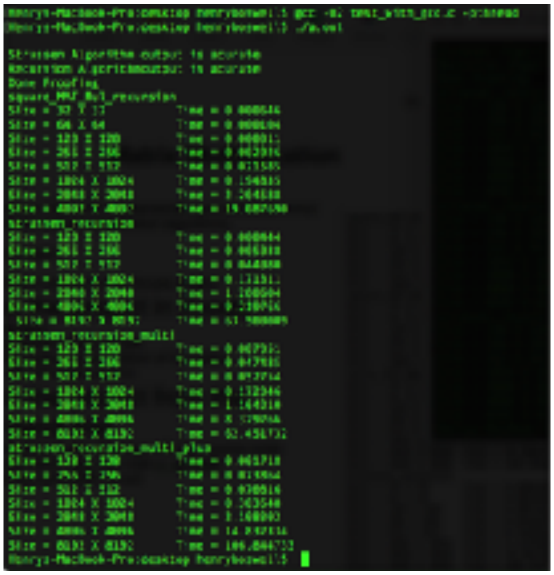
Findings GCC (GNU)



Conclusion

The -O2 command uses different optimizations when used while compiling with GCC or Clang. In this case the GCC compilation that was 5 times faster in almost every case. It turns out that the modification to Strassen's algorithm did not perform as well when compiled using gcc.

The fastest Java library according to a series of tests that I read about on stack overflow is Jeigen. I tested Jeigen on my machine, it multiplies a 4096X4096 matrix in 11 seconds on my machine. It took the c code compiled using GCC using Strassen's multithreaded algorithm just 8.3 seconds. Even though this is expected because Java is interpreted it puts this code's performance in perspective.



```
Henrys-MacBook-Pro:Desktop henryboswell$ gcc -O2 [illegible] -o [illegible]
Henrys-MacBook-Pro:Desktop henryboswell$ ./a.out

Strassen Algorithm output is accurate
Recursion Algorithm output is accurate
Done Proofing
square_MM_Mul_recursion
Size = 32 X 32            Time = 0.0000[illegible]
Size = 64 X 64            Time = 0.000[illegible]
Size = 128 X 128          Time = 0.000[illegible]
Size = 256 X 256          Time = 0.00[illegible]
Size = 512 X 512          Time = 0.0[illegible]
Size = 1024 X 1024        Time = 0.15[illegible]
Size = 2048 X 2048        Time = 1.[illegible]
Size = 4096 X 4096        Time = 19.[illegible]
strassen_recursion
Size = 128 X 128          Time = 0.000[illegible]
Size = 256 X 256          Time = 0.00[illegible]
Size = 512 X 512          Time = 0.0[illegible]
Size = 1024 X 1024        Time = 0.13[illegible]
Size = 2048 X 2048        Time = 1.[illegible]
Size = 4096 X 4096        Time = [illegible]
 Size = 8192 X 8192       Time = 61.[illegible]
strassen_recursion_multi
Size = 128 X 128          Time = 0.00[illegible]
Size = 256 X 256          Time = 0.0[illegible]
Size = 512 X 512          Time = 0.0[illegible]
Size = 1024 X 1024        Time = 0.13[illegible]
Size = 2048 X 2048        Time = 1.1[illegible]
Size = 4096 X 4096        Time = 8.[illegible]
Size = 8192 X 8192        Time = 62.458732
strassen_recursion_multi_plus
Size = 128 X 128          Time = 0.00[illegible]
Size = 256 X 256          Time = 0.0[illegible]
Size = 512 X 512          Time = 0.0[illegible]
Size = 1024 X 1024        Time = 0.[illegible]
Size = 2048 X 2048        Time = [illegible]
Size = 4096 X 4096        Time = 14.[illegible]
Size = 8192 X 8192        Time = 106.844733
Henrys-MacBook-Pro:Desktop henryboswell$ 
```