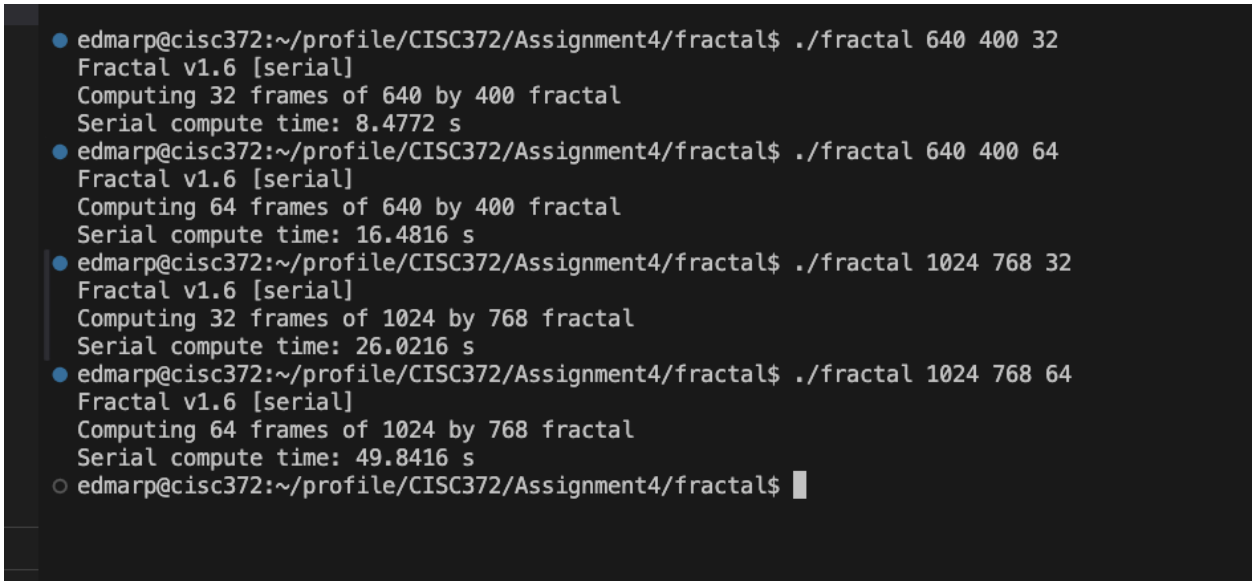
Parameters	Serial Time (Seconds)	CUDA Time (seconds)	Speedup
640 400 32	8.4772	0.0076	1114.32
640 400 64	16.4816	0.047	1373.47
1024 768 32	26.0216	0.0154	1682.56
1024 768 64	49.8146	0.0300	1660.49
1920 180 80	21.5790	0.0116	1860.26
1920 1080 50	38.2632	0.0201	1903.64
2560 1440 10	106.5755	0.0526	2026.15
2560 1440 30	114.2742	0.0736	2062.99
2560 1440 40	151.8362	0.0563 s	2029.73
4096 2160 60	545.0111	0.2480	2197.62

Serial:



CUDA:

```
[edmarp@r2v01 asignment4]$ srun ./fractal_cuda 640 480 32
Fractal v1.6 [CUDA]
Computing 32 frames of 640 by 480 fractal
CUDA compute time: 0.0078 s
[edmarp@r2v01 asignment4]$ srun ./fractal_cuda 640 400 64
Fractal v1.6 [CUDA]
Computing 64 frames of 640 by 400 fractal
CUDA compute time: 0.0127 s
[edmarp@r2v01 asignment4]$ srun ./fractal_cuda 1024 768 32
Fractal v1.6 [CUDA]
Computing 32 frames of 1024 by 768 fractal
CUDA compute time: 0.0179 s
[edmarp@r2v01 asignment4]$ srun ./fractal_cuda 1024 768 64
Fractal v1.6 [CUDA]
Computing 64 frames of 1024 by 768 fractal
CUDA compute time: 0.0349 s
[edmarp@r2v01 asignment4]$ exit
```

```
edmarp@cisc372:~/profile/CISC372/Assignment4/fractal$ srun ./fractal_cuda 640 400 64
Fractal v1.6 [CUDA]
Computing 64 frames of 640 by 400 fractal
CUDA compute time: 0.0473 s
edmarp@cisc372:~/profile/CISC372/Assignment4/fractal$ srun ./fractal_cuda 1920 1080 10
Fractal v1.6 [CUDA]
Computing 10 frames of 1920 by 1080 fractal
CUDA compute time: 0.0502 s
edmarp@cisc372:~/profile/CISC372/Assignment4/fractal$ srun ./fractal_cuda 1920 1080 20
Fractal v1.6 [CUDA]
Computing 20 frames of 1920 by 1080 fractal
CUDA compute time: 0.0997 s
edmarp@cisc372:~/profile/CISC372/Assignment4/fractal$ srun ./fractal_cuda 1920 1080 30
Fractal v1.6 [CUDA]
Computing 30 frames of 1920 by 1080 fractal
CUDA compute time: 0.1545 s
edmarp@cisc372:~/profile/CISC372/Assignment4/fractal$ srun ./fractal_cuda 1920 1080 40
Fractal v1.6 [CUDA]
Computing 40 frames of 1920 by 1080 fractal
CUDA compute time: 0.1970 s
edmarp@cisc372:~/profile/CISC372/Assignment4/fractal$ srun ./fractal_cuda 4096 2160 60
Fractal v1.6 [CUDA]
Computing 60 frames of 4096 by 2160 fractal
CUDA compute time: 1.1797 s
edmarp@cisc372:~/profile/CISC372/Assignment4/fractal$ srun ./fractal_cuda 4096 2160 60
Fractal v1.6 [CUDA]
Computing 60 frames of 4096 by 2160 fractal
CUDA compute time: 1.1796 s
```

Bonus:

Parameters	Serial Time (Seconds)	CUDA Time (seconds)	CUDA Float
640 400 32	8.4772	0.0076	0.004984
640 400 64	16.4816	0.047	0.009691
1024 768 32	26.0216	0.0154	0.009737
1024 768 64	49.8146	0.0300	0.025819
1920 180 80	21.5790	0.0116	0.015374
1920 1080	38.2632	0.0201	0.046468

50			
2560 1440 10	106.5755	0.0526	0.018169 s
2560 1440 30	114.2742	0.0736	0.049555
2560 1440 40	151.8362	0.0563 s	0.064712
4096 2160 60	545.0111	0.2480	0.210796

```
[edmarp@r2v01 asignment4]$ srun ./fractal_float 1024 768 32
Fractal v1.6 [CUDA] - Using floats for computation and doubles for time tracking
Computing 32 frames of 1024 by 768 fractal
CUDA compute time: 0.012767 s
[edmarp@r2v01 asignment4]$ srun ./fractal_float 640 400 64

Fractal v1.6 [CUDA] - Using floats for computation and doubles for time tracking
Computing 64 frames of 640 by 40
CUDA compute time: 0.009691 s
[edmarp@r2v01 asignment4]$ srun ./fractal_float 1024 768 64
Fractal v1.6 [CUDA] - Using floats for computation and doubles for time tracking
Computing 64 frames of 1024 by 768 fractal
CUDA compute time: 0.025819 s
[edmarp@r2v01 asignment4]$ srun ./fractal_float 1920 180 80
Fractal v1.6 [CUDA] - Using floats for computation and doubles for time tracking
Computing 80 frames of 1920 by 180 fractal
```

Open file in editor (cmd + click)

Bonus: Write a second version of your code where all variables of type double are changed to type float. Measure the performance of this second version of your code and give plausible explanations for your observations.

After I replaced all double variables with floats to use less memory and speed up the calculations, since GPUs typically handle float operations faster. Above is where I measured the performance and found that the float-based version ran faster, likely due to lower memory and computational demands. However, this comes with a trade-off in precision, which might slightly impact the visual quality of the fractal output. So we trade off precision for speed again because floats are less accurate since they take up less space