Manpreet Dhindsa 15859742
Takumi Jeff Okamoto 86483191

EECS 117 HW#3

1. Compile and run the naive code. Report the input vector size, the time to execute the kernel, and the effective bandwidth. Explain how the effective bandwidth is being calculated.
    a. Input vector size = N = 8388608
    b. The time to execute the kernel = 0.003347 seconds
    c. Effective bandwidth = 10.03 GB/s
        i. The effective bandwidth is being calculated by
        Effective bandwidth = (input vector size x 4 bytes) / time to execute Kernel

```
=== Running 5 trials of naive ... ===
*** Trial 1 ***
N: 8388608
Timer: gettimeofday
Timer resolution: ~ 1 us (?)
Time to execute naive GPU reduction kernel: 0.003347 secs
Effective bandwidth: 10.03 GB/s
Time to execute naive CPU reduction: 0.128085 secs
SUCCESS: GPU: 41.940048         CPU: 41.940048
```

Figure 1: Trial 1 of Naive.

2. Implement the scheme in kernel1 of stride.cu. Measure and record the resulting performance. How much faster than the initial code is this version?
    a. The effective bandwidth for stride method is 15.03 GB/s.
    b. It is about 1.5x faster than the naive version.

```
=== Running 5 trials of stride ... ===
*** Stride Trial 1 ***
N: 8388608
Timer: gettimeofday
Timer resolution: ~ 1 us (?)
Time to execute strided index GPU reduction kernel: 0.002232 secs
Effective bandwidth: 15.03 GB/s
Time to execute naive CPU reduction: 0.107849 secs
SUCCESS: GPU: 41.934517  CPU: 41.934517
```

Figure 2: Trial 1 of Stride.

3. Implement the scheme in kernel2 of sequential.cu. Record the new effective bandwidth.
    a. The effective bandwidth for sequential method is 19.13 GB/s.
    b. It is about 1.9x better than the naive version.

```
*** Sequential Trial 5 ***
N: 8388608
Timer: gettimeofday
Timer resolution: ~ 1 us (?)
Time to execute sequential index GPU reduction kernel: 0.001754 secs
Effective bandwidth: 19.13 GB/s
Time to execute naive CPU reduction: 0.116089 secs
SUCCESS: GPU: 41.930511  CPU: 41.930511
```

Figure 3: Trial 5 of Sequential.

4. Implement the scheme in kernel3 of first_add.cu and report the effective bandwidth.
    a. The effective bandwidth for the first_add method is 34.17 GB/s.
    b. It is about 3.4x better than the naive version.

```
*** First Add Trial 5 ***
N: 8388608
Timer: gettimeofday
Timer resolution: ~ 1 us (?)
Time to execute first add GPU reduction kernel: 0.000982 secs
Effective bandwidth: 34.17 GB/s
Time to execute naive CPU reduction: 0.115593 secs
SUCCESS: GPU: 41.943066  CPU: 41.943066
```

Figure 4: Trial 5 of First_Add.

5. Implement the scheme in kernel4 of unroll.cu and report the effective bandwidth.
    a. The effective bandwidth for the unroll method is 55.19 GB/s.
    b. It is about 5.5x better than the naive version.

```
=== Running 5 trials of unroll ... ===
*** Unroll Trial 1 ***
N: 8388608
Timer: gettimeofday
Timer resolution: ~ 1 us (?)
Time to execute unrolled GPU reduction kernel: 0.000608 secs
Effective bandwidth: 55.19 GB/s
Time to execute naive CPU reduction: 0.110307 secs
SUCCESS: GPU: 41.943798  CPU: 41.943798
```

Figure 5: Trial 1 of Unroll.

6. Implement the algorithm cascading scheme in multiple.cu and report the effective bandwidth.
    a. The effective bandwidth from the multiple method is 101.99 GB/s.
    b. It is about 10.1x better than the naive version.

```
*** Multiple Trial 3 ***
N: 8388608
Timer: gettimeofday
Timer resolution: ~ 1 us (?)
Time to execute multiple add GPU reduction kernel: 0.000329 secs
Effective bandwidth: 101.99 GB/s
Time to execute naive CPU reduction: 0.115634 secs
SUCCESS: GPU: 41.937424  CPU: 41.937424
```

Figure 6: Trial 3 of Multiple.