CRUZ, AIRON JOHN R.
CEPARCO-S11

I. EXECUTION TIME (Input A = 1 to n , Input B = 1 to 1<<7)

	Vector Size (n)		
Implemented Version	2^{20}	2^{24}	2^{29}
C	3133.33 us	51333.33 us	1665400.00 us
X86-64 (Non-SIMD)	933.33 us	13666.67 us	451533.33 us
SIMD YMM x86	733.33 us	10966.67 us	356633.33 us
CUDA	427.72 us	3989.5 us	49576 us

II. CUDA Overhead

	Vector Size (n)		
Parameters	2^{20}	2^{24}	2^{29}
GPU page fault	149	30	30
CPU page fault	35	30	30
Host to Device Transfer	15089.88 us	22681.92 us	712757 us
Device to Host Transfer	75.52 us	100.42 us	71.711 us

At a vector length ***(n) of*** 2^{20}, the performance of different implementations was evaluated. The C implementation showed the slowest execution time, taking 3,133.33 microseconds (us) to complete. In contrast, both x86-64 implementations, the non-SIMD and SIMD, demonstrated relatively fast execution times of 933.33 us and 733.33 us, respectively. Specifically, the SIMD implementation exhibited a significant speed advantage over the C counterpart, being 4.3 times faster. Furthermore, it demonstrated a 1.27 times speed advantage over the x86-64 implementation. This notable difference in execution times among the first three versions can be attributed to the inherent advantage of SIMD's implementation, which leverages data-level parallelism to perform operations on multiple data elements simultaneously.

However, with the addition of the CUDA implementation, a remarkable improvement in performance was achieved, making it the fastest with an execution time of 427.72 us.

Comparing it to the other versions, the CUDA implementation showcased a substantial speed advantage. It was found to be 7.32 times faster than the C version, 2.18 times faster than the non-SIMD version, and 1.71 times faster than the SIMD version. The reason behind this notable speed difference is that the CUDA implementation harnesses the power of parallel processing on a GPU, allowing for efficient execution of computations on large datasets. By distributing the workload across multiple cores, CUDA can achieve significant performance gains compared to the other implementations.

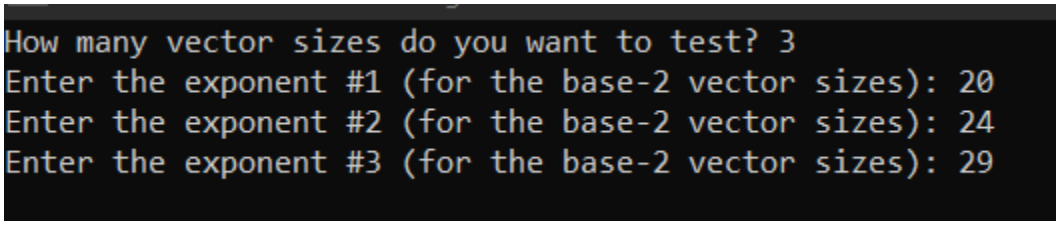
At a vector length ***(n) of 2^{24}***, the comparison between different implementations revealed interesting insights. The C implementation continued to exhibit the slowest execution time, taking a considerable 51,333.33 us to complete. In contrast, the x86-64 implementation showed improved performance with an execution time of 13,666.67 us, followed by the SIMD implementation with 10,966.67 us. However, the CUDA version once again emerged as the fastest, demonstrating its efficiency with an execution time of just 3,989.5 us. Looking at the performance variations from the perspective of SIMD, it was found that SIMD was now 4.7 times faster than its C counterpart and 1.25 times faster than x86-64. This showcased a significantly larger margin of efficiency for SIMD compared to the previous vector size. However, when comparing SIMD to the CUDA version, it became apparent that SIMD was 2.75 times slower than the fastest implementation, highlighting the superior performance of CUDA in this scenario. Finally, at a vector length ***(n) of 2^{29}***, the CUDA implementation maintained its lead in terms of performance. The gap widens as CUDA's execution is now found to be 33.6 times faster than the C version, 9.1 times faster than the x86 non-SIMD version, and 7.2 times faster than the SIMD version. This significant difference further highlighted the performance advantage offered by the CUDA implementation over the other counterparts. Moreover, in addition to leading in terms of execution speed, CUDA consistently demonstrated a remarkable level of efficiency with minimal faults in both the GPU and CPU. On average, there were approximately 30 faults for each, except in the case of n = 2^20 where it had 149 CPU faults and 35 GPU faults. However, these faults did not degrade the overall performance of CUDA as the numbers just prove its reputation as a highly efficient and dependable solution for parallel processing tasks— at the expense of money and complexity in system setup.

Overall, as the length of the vector increased, the CUDA implementation consistently outperformed the other implementations, showcasing its efficiency and ability to leverage the power of parallel processing on GPUs. While SIMD showed improvements compared to the C and x86-64 versions, it still fell short in terms of performance when compared to CUDA. Putting aside the specific version of CUDA employed, the consistent increasing trend in execution times observed across all vector sizes for the C, x86-64, and SIMD implementations indicates that SIMD remained the fastest among the three executions. Based on these observations, it can be concluded that SIMD offers a relatively efficient approach for executing dot product multiplications. These findings align with expectations, as SIMD leverages data-level parallelism to perform simultaneous operations on multiple data elements, resulting in improved performance. SIMD instructions are specifically designed to operate on multiple data elements simultaneously, enabling efficient processing of large datasets. On the other hand, the slower performance of the C implementation can be attributed to its classification as a "high-level"

language, which provides a higher level of abstraction and programmer-friendly features. While these high-level features offer convenience and ease of use, they often come at the cost of performance optimization. Although both C and x86-64 performed the process sequentially, assembly programming, such as the x86-64 implementation, allows programmers to have fine-grained control over the hardware and executed instructions, resulting in lower execution times.

III. PROGRAM EXECUTION SCREENSHOTS

<The input>



<OUTPUT in C>

```
---------------- For ARRAY-SIZE: 1048576-----------------
C function took 3133.33 microseconds ave. time for array size of 1048576 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 34910892457984
Actual Result = 34910892457984
```

```
---------------- For ARRAY-SIZE: 16777216-----------------
C function took 51333.33 microseconds ave. time for array size of 16777216 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 8936852882980864
Actual Result = 8936852882980864
```

```
---------------- For ARRAY-SIZE: 536870912-----------------
C function took 1665400.00 microseconds ave. time for array size of 536870912 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 9151315158734209024
Actual Result = 9151315158734209024
```

<OUTPUT in x86-64>

```
x86-64 kernel took 933.33 microseconds ave. time for array size of 1048576 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 34910892457984
Actual Result = 34910892457984
```

```
x86-64 kernel took 13666.67 microseconds ave. time for array size of 16777216 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 8936852882980864
Actual Result = 8936852882980864
```

```
x86-64 kernel took 451533.33 microseconds ave. time for array size of 536870912 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 9151315158734209024
Actual Result = 9151315158734209024
```

<OUTPUT in SIMD>

```
x86-64 SIMD kernel took 733.33 microseconds ave. time for array size of 1048576 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 34910892457984
Actual Result = 34910892457984
```

```
x86-64 SIMD kernel took 10966.67 microseconds ave. time for array size of 16777216 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 8936852882980864
Actual Result = 8936852882980864
```

```
x86-64 SIMD kernel took 356633.33 microseconds ave. time for array size of 536870912 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 9151315158734209024
Actual Result = 9151315158734209024
```

<Overall Output of test run>

```
Microsoft Visual Studio Debug Console
How many vector sizes do you want to test? 3
Enter the exponent #1 (for the base-2 vector sizes): 20
Enter the exponent #2 (for the base-2 vector sizes): 24
Enter the exponent #3 (for the base-2 vector sizes): 29

---------------- For ARRAY-SIZE: 1048576----------------
C function took 3133.33 microseconds ave. time for array size of 1048576 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 34910892457984
Actual Result = 34910892457984

x86-64 kernel took 933.33 microseconds ave. time for array size of 1048576 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 34910892457984
Actual Result = 34910892457984

x86-64 SIMD kernel took 733.33 microseconds ave. time for array size of 1048576 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 34910892457984
Actual Result = 34910892457984

---------------- For ARRAY-SIZE: 16777216----------------
C function took 51333.33 microseconds ave. time for array size of 16777216 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 8936852882980864
Actual Result = 8936852882980864

x86-64 kernel took 13666.67 microseconds ave. time for array size of 16777216 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 8936852882980864
Actual Result = 8936852882980864

x86-64 SIMD kernel took 10966.67 microseconds ave. time for array size of 16777216 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 8936852882980864
Actual Result = 8936852882980864

---------------- For ARRAY-SIZE: 536870912----------------
C function took 1665400.00 microseconds ave. time for array size of 536870912 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 9151315158734209024
Actual Result = 9151315158734209024

x86-64 kernel took 451533.33 microseconds ave. time for array size of 536870912 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 9151315158734209024
Actual Result = 9151315158734209024

x86-64 SIMD kernel took 356633.33 microseconds ave. time for array size of 536870912 throughout 30 runs
 CORRECT PROGRAM
Expected Result = 9151315158734209024
Actual Result = 9151315158734209024

D:\NASM\Cruz,Airon_DiveDeepProject\x64\Debug\Cruz,Airon_DiveDeepProject.exe (process 15068) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

<OUTPUT In CUDA>

```
==1418== NVPROF is profiling process 1418, command: ./dotproduct_cuda
Device # = 0
numThreads = 1024, numBlocks = 1024

 PROGRAM CHECKER:

 CORRECT PROGRAM
EXPECTED OUTPUT: 34910892457984
ACTUAL OUTPUT: 34910892457984
==1418== Profiling application: ./dotproduct_cuda
==1418== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:  100.00%  11.663ms        30  388.75us  296.73us  824.78us  dotproduct(__int64*, __int64*, int, __int64*)
      API calls:   92.34%  247.40ms         3  82.467ms  15.440us  247.34ms  cudaMallocManaged
                    4.75%  12.727ms        30  424.23us  301.18us  1.3121ms  cudaDeviceSynchronize
                    1.90%  5.0989ms         6  849.81us  32.575us  2.1831ms  cudaMemPrefetchAsync
                    0.45%  1.2057ms         3  401.89us  66.508us  597.06us  cudaFree
                    0.30%  811.48us         4  202.87us  2.5500us  406.61us  cudaMemAdvise
                    0.20%  528.64us        30  17.621us  12.788us  50.866us  cudaLaunchKernel
                    0.05%  122.11us       101  1.2080us     129ns  56.300us  cuDeviceGetAttribute
                    0.01%  23.714us         1  23.714us  23.714us  23.714us  cuDeviceGetName
                    0.00%  5.4550us         1  5.4550us  5.4550us  5.4550us  cuDeviceGetPCIBusId
                    0.00%  2.2050us         1  2.2050us  2.2050us  2.2050us  cudaGetDevice
                    0.00%  1.9170us         3     639ns     298ns  1.3150us  cuDeviceGetCount
                    0.00%  1.0080us         2     504ns     277ns     731ns  cuDeviceGet
                    0.00%     531ns         1     531ns     531ns     531ns  cuModuleGetLoadingMode
                    0.00%     362ns         1     362ns     362ns     362ns  cuDeviceTotalMem
                    0.00%     205ns         1     205ns     205ns     205ns  cuDeviceGetUuid

==1418== Unified Memory profiling result:
Device "Tesla T4 (0)"
   Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
      41  406.93KB  4.0000KB  2.0000MB  16.29297MB  1.508988ms  Host To Device
      33  7.3936KB  4.0000KB  60.000KB  244.0000KB  75.51800us  Device To Host
     149         -         -         -           -  4.086829ms  Gpu page fault groups
       5         -         -         -           -  2.153646ms  Page throttles
      25  4.0000KB  4.0000KB  4.0000KB  100.0000KB           -  Memory thrashes
Total CPU Page faults: 35
Total CPU thrashes: 25
Total CPU throttles: 5
```

```
==726== NVPROF is profiling process 726, command: ./dotproduct_cuda1
Device # = 0
numThreads = 1024, numBlocks = 16384

 PROGRAM CHECKER:

 CORRECT PROGRAM
EXPECTED OUTPUT: 8936852882980864
ACTUAL OUTPUT: 8936852882980864
==726== Profiling application: ./dotproduct_cuda1
==726== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:  100.00%  119.68ms        30  3.9895ms  3.9644ms  4.0550ms  dotproduct(__int64*, __int64*, int, __int64*)
      API calls:   49.27%  244.83ms         3  81.609ms  22.114us  244.73ms  cudaMallocManaged
                   26.85%  133.45ms        30  4.4484ms  3.9695ms  17.475ms  cudaDeviceSynchronize
                   16.66%  82.813ms         6  13.802ms  42.840us  36.390ms  cudaMemPrefetchAsync
                    4.07%  20.250ms         3  6.7499ms  111.90us  10.406ms  cudaFree
                    2.98%  14.828ms         4  3.7070ms  2.9340us  7.8014ms  cudaMemAdvise
                    0.13%  624.43us        30  20.814us  13.142us  50.350us  cudaLaunchKernel
                    0.02%  116.01us       101  1.1480us     140ns  48.100us  cuDeviceGetAttribute
                    0.00%  23.672us         1  23.672us  23.672us  23.672us  cuDeviceGetName
                    0.00%  6.7520us         1  6.7520us  6.7520us  6.7520us  cuDeviceGetPCIBusId
                    0.00%  2.3660us         1  2.3660us  2.3660us  2.3660us  cudaGetDevice
                    0.00%  1.9120us         3     637ns     207ns  1.4290us  cuDeviceGetCount
                    0.00%  1.1910us         2     595ns     268ns     923ns  cuDeviceGet
                    0.00%     488ns         1     488ns     488ns     488ns  cuModuleGetLoadingMode
                    0.00%     418ns         1     418ns     418ns     418ns  cuDeviceTotalMem
                    0.00%     248ns         1     248ns     248ns     248ns  cuDeviceGetUuid

==726== Unified Memory profiling result:
Device "Tesla T4 (0)"
   Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
     163  1.5731MB  4.0000KB  2.0000MB  256.4102MB  22.68192ms  Host To Device
      35  10.399KB  4.0000KB  60.000KB  364.0000KB  100.4150us  Device To Host
      30         -         -         -           -  2.286059ms  Gpu page fault groups
      22  4.0000KB  4.0000KB  4.0000KB  88.00000KB           -  Memory thrashes
Total CPU Page faults: 30
Total CPU thrashes: 22
```

```
==815== NVPROF is profiling process 815, command: ./dotproduct_cuda2
Device # = 0
numThreads = 1024, numBlocks = 524288

 PROGRAM CHECKER:

 CORRECT PROGRAM
EXPECTED OUTPUT: 9151315158734209024
ACTUAL OUTPUT: 9151315158734209024
==815== Profiling application: ./dotproduct_cuda2
==815== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:  100.00%  1.48727s        30  49.576ms  48.370ms  54.072ms  dotproduct(__int64*, __int64*, int, __int64*)
      API calls:   53.15%  3.17183s         6  528.64ms  44.245us  1.18577s  cudaMemPrefetchAsync
                   24.93%  1.48746s        30  49.582ms  48.378ms  54.079ms  cudaDeviceSynchronize
                   10.95%  653.53ms         3  217.84ms  174.64us  332.54ms  cudaFree
                    6.87%  409.76ms         4  102.44ms  4.9460us  205.18ms  cudaMemAdvise
                    4.08%  243.69ms         3  81.230ms  21.591us  243.61ms  cudaMallocManaged
                    0.01%  811.72us        30  27.057us  20.165us  59.782us  cudaLaunchKernel
                    0.00%  114.01us       101  1.1280us     135ns  47.822us  cuDeviceGetAttribute
                    0.00%  26.663us         1  26.663us  26.663us  26.663us  cuDeviceGetName
                    0.00%  5.5250us         1  5.5250us  5.5250us  5.5250us  cuDeviceGetPCIBusId
                    0.00%  1.9740us         3     658ns     237ns  1.4330us  cuDeviceGetCount
                    0.00%  1.9330us         1  1.9330us  1.9330us  1.9330us  cudaGetDevice
                    0.00%     958ns         2     479ns     284ns     674ns  cuDeviceGet
                    0.00%     439ns         1     439ns     439ns     439ns  cuModuleGetLoadingMode
                    0.00%     394ns         1     394ns     394ns     394ns  cuDeviceTotalMem
                    0.00%     220ns         1     220ns     220ns     220ns  cuDeviceGetUuid

==815== Unified Memory profiling result:
Device "Tesla T4 (0)"
   Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
    4131  1.9832MB  4.0000KB  2.0000MB  8.000401GB  712.7570ms  Host To Device
      35  10.399KB  4.0000KB  60.000KB  364.0000KB  71.71100us  Device To Host
      30         -         -         -           -  7.543483ms  Gpu page fault groups
      22  4.0000KB  4.0000KB  4.0000KB  88.00000KB           -  Memory thrashes
Total CPU Page faults: 30
Total CPU thrashes: 22
```