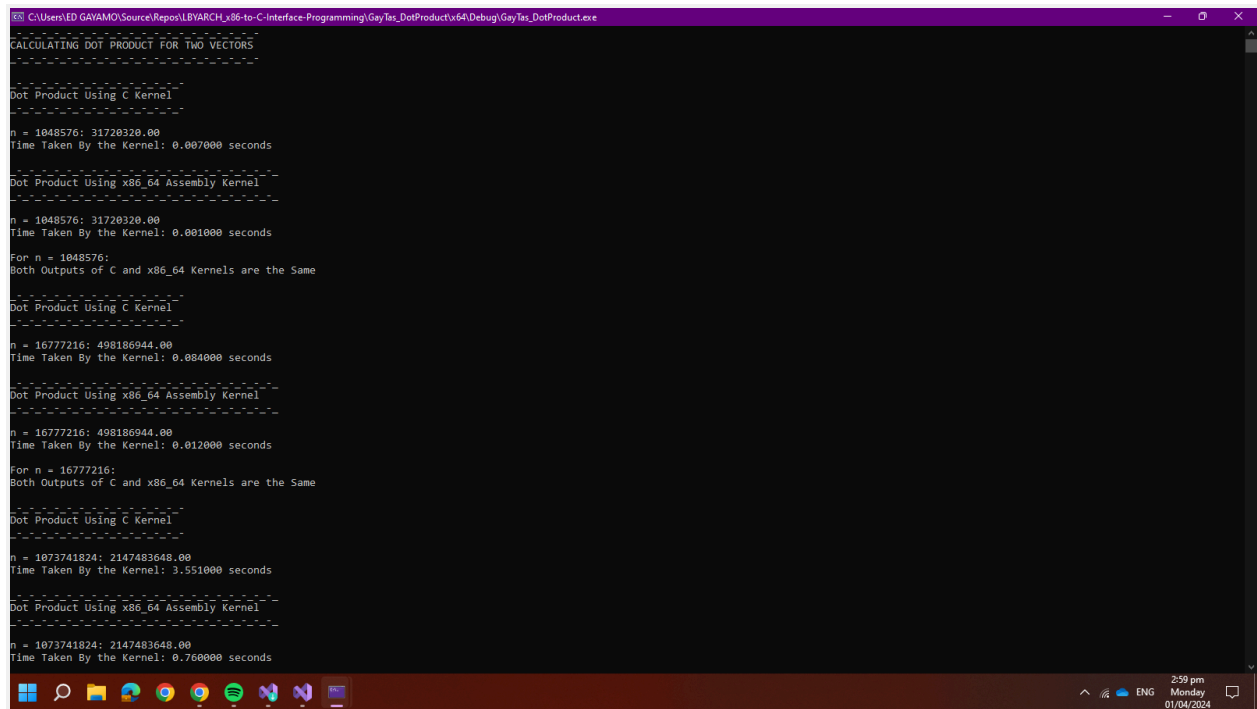


Debug Mode:



```
C:\Users\ED GAYAMO\Source\Repos\LBVARCH_x86-to-C-Interface-Programming\Gay\DotProduct\Debug\Gay\DotProduct.exe
CALCULATING DOT PRODUCT FOR TWO VECTORS
-----
Dot Product Using C Kernel
-----
n = 1048576: 31720320.00
Time Taken By the Kernel: 0.007000 seconds
-----
Dot Product Using x86_64 Assembly Kernel
-----
n = 1048576: 31720320.00
Time Taken By the Kernel: 0.001000 seconds
For n = 1048576:
Both Outputs of C and x86_64 Kernels are the Same
-----
Dot Product Using C Kernel
-----
n = 16777216: 498186944.00
Time Taken By the Kernel: 0.084000 seconds
-----
Dot Product Using x86_64 Assembly Kernel
-----
n = 16777216: 498186944.00
Time Taken By the Kernel: 0.012000 seconds
For n = 16777216:
Both Outputs of C and x86_64 Kernels are the Same
-----
Dot Product Using C Kernel
-----
n = 1073741824: 2147483648.00
Time Taken By the Kernel: 3.551000 seconds
-----
Dot Product Using x86_64 Assembly Kernel
-----
n = 1073741824: 2147483648.00
Time Taken By the Kernel: 0.760000 seconds
```

When operating in debug mode, the code execution tends to be unoptimized and is likely to incur longer execution times as compared to release mode, mainly due to the presence of additional debugging information and less aggressive compiler optimizations. We conducted 30 runs in debug mode and recorded the execution times of the C and assembly implementations. Despite the possibility of extended execution times due to debugging overhead, we observed that the debug mode runs demonstrated consistent and stable behavior throughout. The dot product calculation exhibited an average execution time of approximately 0.046 seconds during the debug mode execution. This average takes into consideration the cumulative performance of the program in debug mode while considering variations in individual execution times.

Release Mode:

```
C:\Users\ED GAVAMO\Source\Repos\LBVARCH\X86-to-C-Interface-Programming\Gay\DotProduct\64\Release\Gay\DotProduct.exe
CALCULATING DOT PRODUCT FOR TWO VECTORS
-----
Dot Product Using C Kernel
-----
n = 1048576: 31773524.00
Time Taken By the Kernel: 0.001000 seconds

Dot Product Using x86_64 Assembly Kernel
-----
n = 1048576: 31773524.00
Time Taken By the Kernel: 0.001000 seconds

For n = 1048576:
Both Outputs of C and x86_64 Kernels are the Same

Dot Product Using C Kernel
-----
n = 16777216: 498203712.00
Time Taken By the Kernel: 0.019000 seconds

Dot Product Using x86_64 Assembly Kernel
-----
n = 16777216: 498203712.00
Time Taken By the Kernel: 0.012000 seconds

For n = 16777216:
Both Outputs of C and x86_64 Kernels are the Same

Dot Product Using C Kernel
-----
n = 1073741824: 2147483648.00
Time Taken By the Kernel: 1.167000 seconds

Dot Product Using x86_64 Assembly Kernel
-----
n = 1073741824: 2147483648.00
Time Taken By the Kernel: 0.817000 seconds
```

On the other hand, release mode compilation optimizes code execution for the best possible performance unlike what occurs when it is in a debug state. We also conducted 30 runs of the program in release mode to capture the execution times. The optimizations applied during release mode compilation resulted in shorter execution times compared to debug mode. Release mode benefits from compiler optimizations such as function inlining, loop unrolling, and removal of debugging symbols, resulting in faster execution times compared to debug mode. These optimizations eliminate unnecessary overhead, leading to improved performance. In release mode, the code is optimized for performance by the compiler, resulting in a notable improvement in execution times. The average execution time for the dot product calculation in release mode was approximately 0.0128 seconds. The significantly shorter average execution time in release mode highlights the effectiveness of compiler optimizations in improving program performance.

Different compilation modes affect program performance. Debug mode offers enhanced debugging capabilities but slower execution, while release mode prioritizes performance optimization, resulting in faster execution times. Developers should choose the appropriate compilation mode based on their project requirements to achieve optimal performance and efficiency.