

Final Take Home Test

Optimization of Dot Product Computation of Two Vectors Using Vector Instructions

Name: Ismail Akram

Date: 12/14/2021

Course: CSC 34200-G & CSC34300-DE

Contents

Objective:	3
Introduction	4
Functionality and Specification:	5
Intel x86 (Visual Studio Code)	5
Dot Product (Pointer) [Non-Optimized].....	6
Dot Product (Pointer) [Compiler Generated Optimization].....	7
Dot Product (Pointer) [Manually Optimized].....	8
VDPPS.....	9
Intel x64 (Linux GCC)	10
Dot Product (Pointer) [Non-Optimized].....	10
Dot Product (Pointer) [Compiler Optimization flags]	11
Dot Product (Pointer) [Compiler Assembly Output]	12
Simulation	13
Intel x86 (Visual Studio Code)	13
Dot Product (Pointer) [Non-Optimized].....	13
Dot Product (Pointer) [Compiler Generated Optimization].....	20
Dot Product (Pointer) [Manually Optimized].....	27
VDPPS.....	34
Intel x64 (Linux GCC)	41
Dot Product (Pointer) [Non-Optimized].....	41
Dot Product (Pointer) [Compiler Optimization flags]	42
Results	45
Conclusion.....	47

Objective:

To optimize compiler generated code to compute dot product using vector instructions.

To understand the compiler's ability to reduce run-time of a program, we must turn off optimizations. We then run our dot products (I've chosen pointers since it's more ideal) for the arrays. The time needed to run these programs is measure for the functions with various array sizes of magnitude 2.

After the runs, we then compare the compiler optimizes code with the non-optimized code. We then manually optimized the compiler-generated assembly code so that the dot product for vastly greater arrays will be calculated more swiftly. We will then use VDPPS for our final runs. The compiler will use auto-parallelization and auto-vectorization for optimization of large arrays.

Introduction

To identify my hardware's processor, I used CPU-Z application. I took note of the instruction sets that are available.

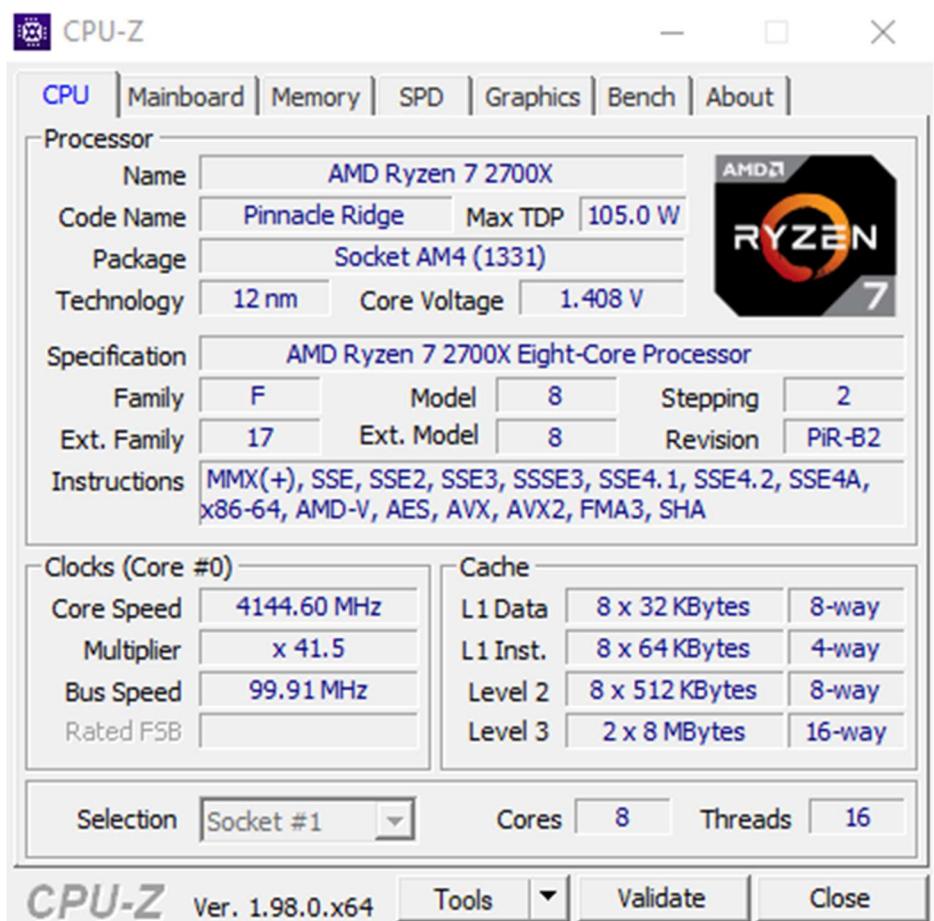


Fig 1. My CPU Processor info from CPU-Z (AVX2)

```
izzy@izzy-VirtualBox:~$ lscpu | grep "Model name"
Model name: AMD Ryzen 7 2700X Eight-Core Processor
```

Fig 1a: CPU Processor info in VM

Functionality and Specification:

Intel x86 (Visual Studio Code)

```

1 //include <iostream>
2 //include <vector>
3 //include <algorithm>
4 //include <valgrind>
5
6 using namespace std;
7
8 const int arr_size = 16; // array size is constant; we'll be performing for N = 16, 32, 64, 128, 256, 512, 8192
9
10 static float arr_size; // two arrays
11 static float arr_1[16];
12 static float arr_2[16];
13 float Dot_Prod_Ptr(float arr_1, float arr_2, const int size);
14
15 #ifndef main()
16     int arr_1[16], arr_2[16], freq = 0;
17     float arr_1[16], arr_2[16];
18     double total_time = 0;
19
20     for (int n = 0; n < arr_size; n++) {
21         arr_1[n] = n * 2.0;
22         arr_2[n] = n * 3.0;
23     }
24
25     cout << "Array size: " << arr_size << endl;
26
27     // to account for deslatency in error (this is normal), we are running this program 10 times, and then taking the average result. That way we get the accurate representative running time.
28     for (int i = 0; i < 10; i++) {
29         if (QueryPerformanceCounter((LARGE_INTEGER*)&ctr_1) != 0) {
30             d_prod = Dot_Prod_Ptr(arr_1, arr_2, arr_size); // using Pointer
31             QueryPerformanceCounter((LARGE_INTEGER*)&ctr_2);
32
33             // cout << "Beginning values: " << ctr_1 << endl; // remove comment if you would like to know arbitrary beginning and ending values
34             // cout << "Ending values: " << ctr_2 << endl;
35             // cout << "Frequency: " << freq;
36             // cout << "QueryPerformanceFrequency("LARGE_INTEGER")>freq;
37             // cout << "QueryPerformanceCounter resolution " << freq << "accs" << endl; // remove comment to see accs in resolution ... i.e. the frequency
38             // cout << "Total time: " << ((ctr_2 - ctr_1) * 1.0 / freq) << endl; // remove comment to see total time
39             total_time += ((ctr_2 - ctr_1) * 1.0 / freq);
40
41             // cout << "Total time: " << total_time << endl; // remove comment to see total time
42         }
43     }
44
45     cout << "Average time: " << (total_time / 10) << endl; // Note this time and graph it into the excel doc
46
47     system("pause");
48 }
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92

```

Fig 2. Source.cpp (code)

The dot product calculation is implemented via pointer arithmetic (Fig 3). In theory, pointer arithmetic is faster than indexing since indices require a multiplication of 4 for each increment whereas pointers forego the multiplication step. Therefore, I chose pointers over indexing.

Note: Auto-vectorization is disabled via `#pragma loop(no_vector)`.

Source.cpp (Fig 2) is the main function. The array size is `const 16` for the first calculation and will change to various sizes to measure different times. Two static arrays of the same size are declared. They are initialized to arbitrary numbers since the result of the dot product is irrelevant.

Note: `QueryPerformanceCounter` is used to measure the run-time of the dot product function.

Note: optimization, auto-vectorization, and auto-parallelization is disabled in Visual Studio.

The run-time is measured for the following array sizes; $N = 16, 32, 64, 128, 256, 512$, and 1024 .

Dot Product (Pointer)

Pointer arithmetic dot product. Note that we're passing float.

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and Search (Ctrl+Q). The title bar says "Dot_Prod_Pntr". The left sidebar displays the Solution Explorer with a single project named "Dot_Prod_Pntr". The main editor window contains the following C++ code:

```
1 ErlangDotProd(float* arr_1, float* arr_2, int size) {
2     float sum = 0.0;
3     float *a = arr_1;
4     float *b = arr_2;
5     for (a = arr_1[0], b = arr_2[0]; a < arr_1[size]; a++, b++) {
6         sum += (*a) * (*b);
7     }
8     return sum;
9 }
```

The bottom status bar shows "Ready".

Fig 3: Dot_Prod_Pntr.cpp (Code)

Dot Product (Pointer) [Compiler Generated Optimization]

The compiler optimizes the generated .asm code (Fig 4). We will then manually improve this code for the next tests. Parallelization and vectorization is enabled for these tests.

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Go, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and Search (Ctrl+F). The title bar indicates the project is "Dot_Prod_Ptr" and the file is "Dot_Prod_Ptr.cpp". The Solution Explorer on the left shows a single project named "Dot_Prod_Ptr" with files like "Dot_Prod_Ptr.h", "Dot_Prod_Ptr.cpp", and "Dot_Prod_Ptr.exe". The main window displays the assembly code for "Dot_Prod_Ptr.cpp". The assembly code is highly optimized, using intrinsics like _mm256_loadu_ps and _mm256_storeu_ps. It includes comments for compiler-specific pragmas and defines. The assembly uses registers like rsi, rdi, rbp, and r15, along with memory addresses and immediate values. The code is annotated with labels such as ".L1", ".L2", and ".L3". The bottom status bar shows "100% 0 No issues found".

Fig 4: Dot Prod Pntr (Code) [Compiler Generated Optimization]

Dot Product (Pointer) [Manually Optimized]

For pointer arithmetic dot product, the assembly code has been manually optimized (Fig 5). This was done by removing redundant/repetitive code and/or moved to out of the loop, so it's not repeated unnecessarily. We managed to reduce the 105 lines code down to 72 lines.

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "Optimized_Dot_Product". The Solution Explorer on the left shows a single project named "Optimized_Dot_Product" with one file "Op_Dot_Prod.cpp". The code editor displays assembly language generated by the Microsoft Optimizing Compiler Version 19.29.30816.0. The assembly code includes directives like .TITLE, .DATA, .CODE, .INCLUDE, and .GLOBAL. It also contains memory operations such as mov, add, sub, and cmp, along with various processor-specific instructions. The assembly code corresponds to the C++ code in "Op_Dot_Prod.cpp". The status bar at the bottom right shows "File 12 Ch 22 Col 23 TANG CRU".

Fig 5: Opt_Dot_Prod_Pntr.asm (Code) [Manually Optimized]

VDPPS

For further optimization, we are using vector instruction DPPS to computer the dot product. We are then using the same QueryPerformanceCounter function to measure the execution time.

```

1 //include <immintrin.h>
2 //include <stdio.h>
3
4 float Dot_Prod_Pntr(float* a, float* b, int n) {
5     __m256 ymm1, ymm2, result = _mm256_setzero_ps();
6     int i;
7     float product = 0.0f;
8
9     #pragma loop(1,parallel)(3)
10    for (int i = 0; i < n / 4; i++) {
11        ymm1 = _mm256_load_ps(a + i * 8);
12        ymm1 = _mm256_load_ps(a + i * 8);
13        ymm2 = _mm256_load_ps(b + i * 8);
14        ymm2 = _mm256_load_ps(b + i * 8);
15        result = _mm256_dp_ps(result, ymm1, ymm2, _MM_DENORMALS_ZERO_DISABLE);
16        result = _mm256_dp_ps(result, ymm1, ymm2, _MM_DENORMALS_ZERO_DISABLE);
17        result = _mm256_dp_ps(result, ymm1, ymm2, _MM_DENORMALS_ZERO_DISABLE);
18        result = _mm256_dp_ps(result, ymm1, ymm2, _MM_DENORMALS_ZERO_DISABLE);
19        product += (_mm256_cvt_ps(result));
20    }
21
22    //_mm256_store_ps(product, result);
23
24    return product;
25}

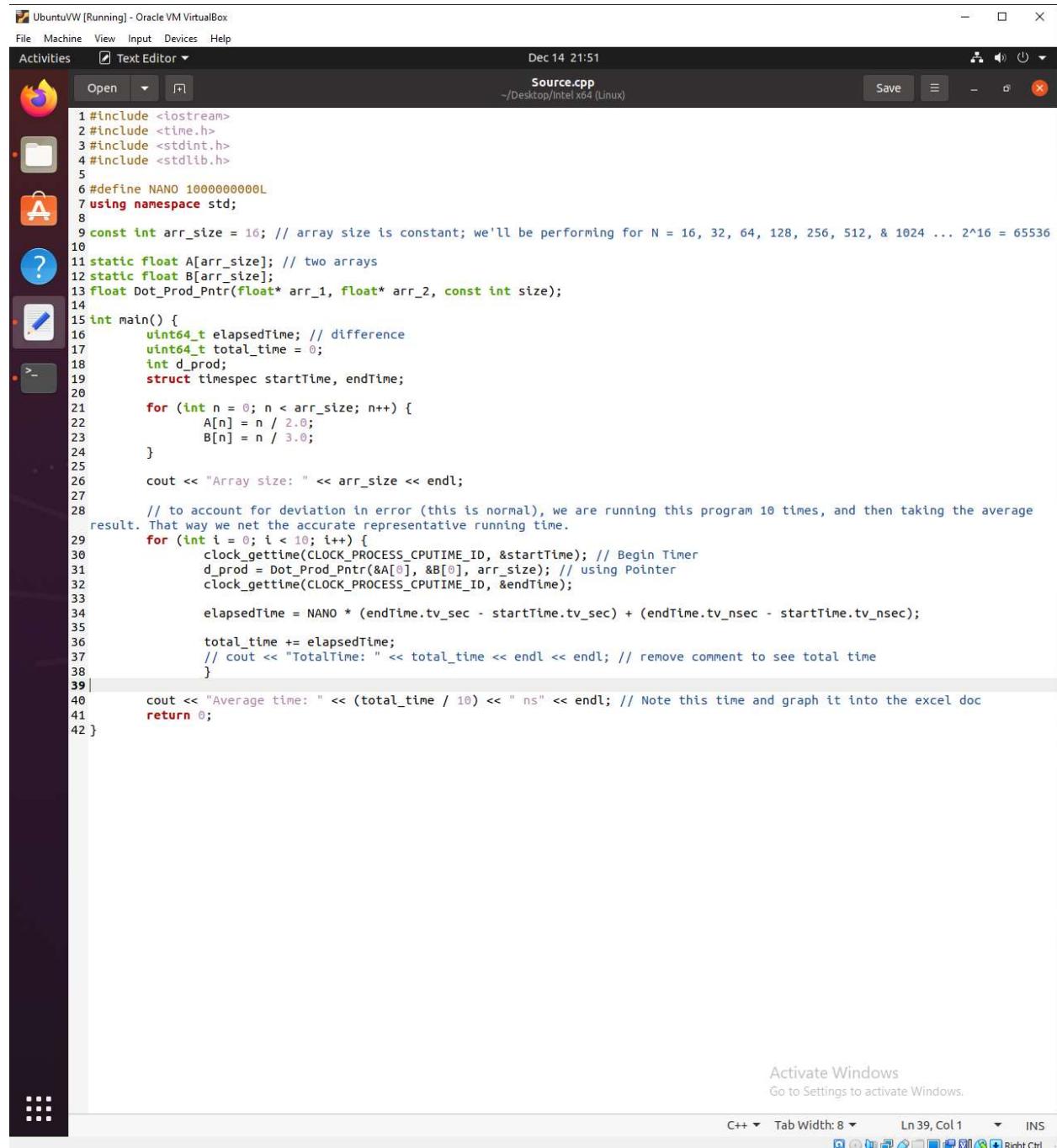
```

Fig 6: VDPPS_Dot_Prod_Pntr.cpp (Code)

Intel x64 (Linux GCC)

Dot Product (Pointer) [Non-Optimized]

For our run in Linux 64-bit, we are using the same C++ codes for the function of dot product. However, some modifications are needed for Source.cpp; we must use `clock_gettime` to measure the execution time. We use the `-O0` Optimization flag for no optimization.



```

UbuntuVW [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor ▾ Dec 14 21:51
Source.cpp ~/Desktop/Intel x64 (Linux)
Save
Open ▾
1 #include <iostream>
2 #include <time.h>
3 #include <stdint.h>
4 #include <stdlib.h>
5
6 #define NANO 1000000000L
7 using namespace std;
8
9 const int arr_size = 16; // array size is constant; we'll be performing for N = 16, 32, 64, 128, 256, 512, & 1024 ... 2^16 = 65536
10
11 static float A[arr_size]; // two arrays
12 static float B[arr_size];
13 float Dot_Prod_Pntr(float* arr_1, float* arr_2, const int size);
14
15 int main() {
16     uint64_t elapsedTime; // difference
17     uint64_t total_time = 0;
18     int d_prod;
19     struct timespec startTime, endTime;
20
21     for (int n = 0; n < arr_size; n++) {
22         A[n] = n / 2.0;
23         B[n] = n / 3.0;
24     }
25
26     cout << "Array size: " << arr_size << endl;
27
28     // to account for deviation in error (this is normal), we are running this program 10 times, and then taking the average
29     // result. That way we get the accurate representative running time.
30     for (int i = 0; i < 10; i++) {
31         clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &startTime); // Begin Timer
32         d_prod = Dot_Prod_Pntr(&A[0], &B[0], arr_size); // using Pointer
33         clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &endTime);
34
35         elapsedTime = NANO * (endTime.tv_sec - startTime.tv_sec) + (endTime.tv_nsec - startTime.tv_nsec);
36
37         total_time += elapsedTime;
38         // cout << "TotalTime: " << total_time << endl << endl; // remove comment to see total time
39     }
40     cout << "Average time: " << (total_time / 10) << " ns" << endl; // Note this time and graph it into the excel doc
41     return 0;
42 }

```

Fig 7: Source.cpp (Code)

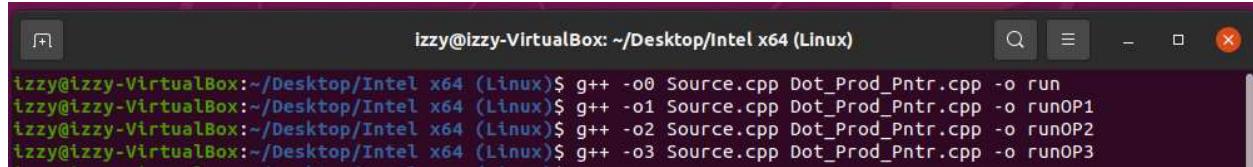
Dot Product (Pointer) [Compiler Optimization flags]

For optimization in a Linux environment, it is much more efficient to use the compiler flags. These are enabled by using “-o” alongside an optimization level. We’re using the flags that reduce execution time.

There are tradeoffs to using optimization glass, namely in compilation time and memory usage.

Optimization Flag	Optimization Level	Execution Time	Optimization Notes
-o0	O (Default)	+	None
-o1	1	-	Basic
-o2	2	--	Recommended
-o3	3 (Highest)	---	Not Guaranteed

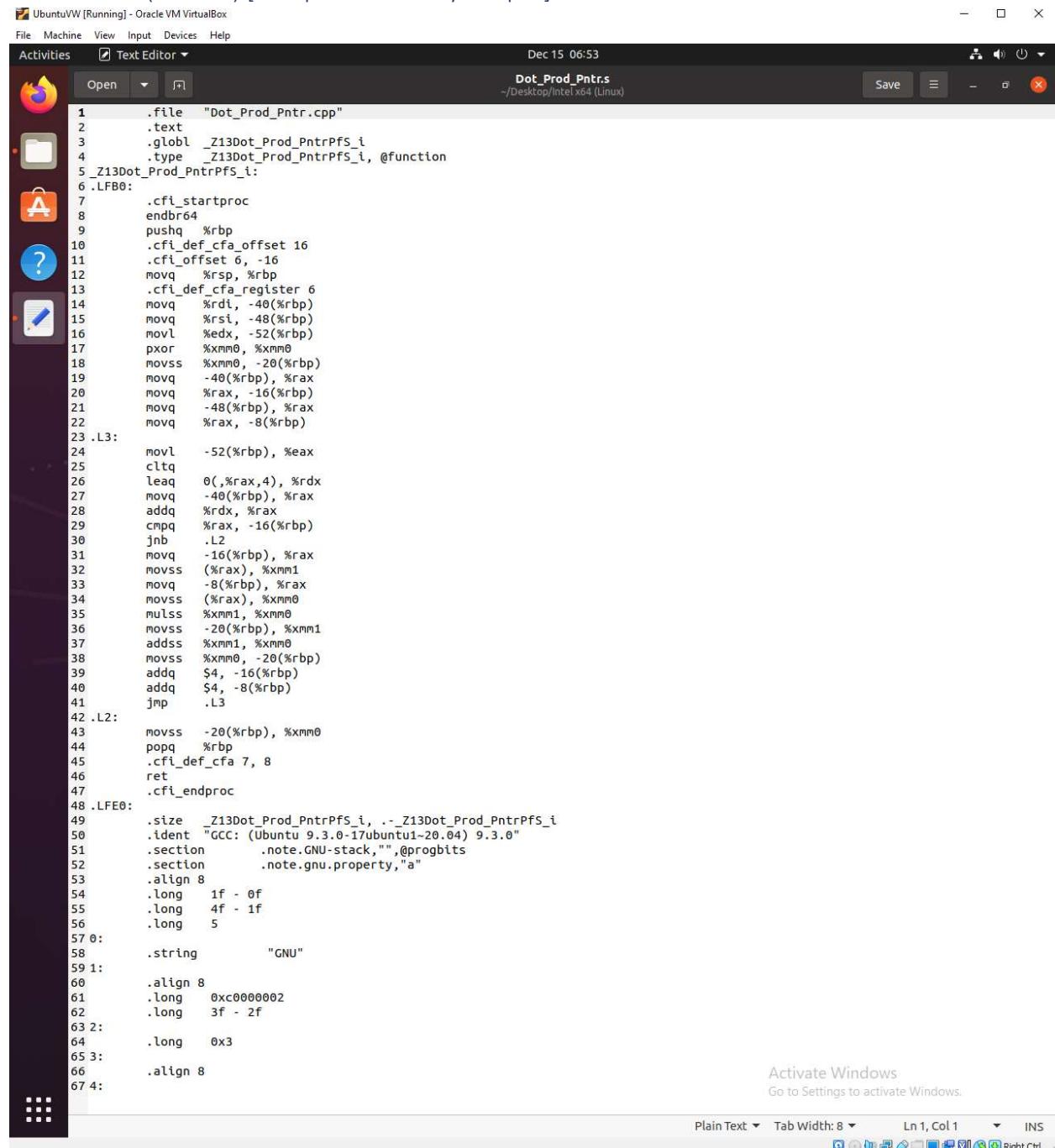
Fig 8: Table summary of Optimization Flags



```
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o1 Source.cpp Dot_Prod_Pntr.cpp -o runOP1
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o runOP2
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o3 Source.cpp Dot_Prod_Pntr.cpp -o runOP3
```

Fig 9: Compiling Dot Product Programs using the optimization flags

Dot Product (Pointer) [Compiler Assembly Output]



The screenshot shows a Linux desktop environment with a text editor window open. The window title is "Dot_Prod_Pntr.s" and the file path is "~/Desktop/Intel x64 (Linux)". The code displayed is assembly language generated by a compiler, specifically for the Intel x64 architecture. The assembly code includes various instructions like .file, .text, .globl, .type, .LFB0:, .L3:, .L2:, .LFE0:, .size, .ident, .section, .align, .long, and .string. The code implements a dot product operation using pointers and SIMD registers (xmm0-xmm1). The text editor has a dark theme and includes standard file operations like Open, Save, and Exit.

```

1     .file   "Dot_Prod_Pntr.cpp"
2     .text
3     .globl  _Z13Dot_Prod_PntrPfs_i
4     .type   _Z13Dot_Prod_PntrPfs_i, @function
5 _Z13Dot_Prod_PntrPfs_i:
6 .LFB0:
7     .cfi_startproc
8     endbr64
9     pushq   %rbp
10    .cfi_offset %rbp, -16
11    movq   %rsp, %rbp
12    .cfi_offset %rbp, -16
13    .cfi_register 6, %rbp
14    movq   %rdi, -40(%rbp)
15    movq   %rsi, -48(%rbp)
16    movl   %edx, -52(%rbp)
17    pxor   %xmm0, %xmm0
18    movss  %xmm0, -20(%rbp)
19    movq   %rax, -40(%rbp)
20    movq   %rax, -16(%rbp)
21    movq   %rax, -48(%rbp)
22    movq   %rax, -8(%rbp)
23 .L3:
24    movl   -52(%rbp), %eax
25    cltq
26    leaq   0(%rax,4), %rdx
27    movq   -40(%rbp), %rax
28    addq   %rdx, %rax
29    cmpq   %rax, -16(%rbp)
30    jnb    .L2
31    movq   -16(%rbp), %rax
32    movss  (%rax), %xmm1
33    movq   -8(%rbp), %rax
34    movss  (%rax), %xmm0
35    mulss  %xmm1, %xmm0
36    movss  -20(%rbp), %xmm1
37    addss  %xmm1, %xmm0
38    movss  %xmm0, -20(%rbp)
39    addq   $4, -16(%rbp)
40    addq   $4, -8(%rbp)
41    jmp    .L3
42 .L2:
43    movss  -20(%rbp), %xmm0
44    popq   %rbp
45    .cfi_offset %rbp, 8
46    ret
47    .cfi_endproc
48 .LFE0:
49     .size  _Z13Dot_Prod_PntrPfs_i, .-_Z13Dot_Prod_PntrPfs_i
50     .ident "GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0"
51     .section .note.GNU-stack,"@progbits"
52     .section .note.gnu.property,"a"
53     .align 8
54     .long   1f - 0f
55     .long   4f - 1f
56     .long   5
57 0:
58     .string  "GNU"
59 1:
60     .align 8
61     .long   0xc0000002
62     .long   3f - 2f
63 2:
64     .long   0x3
65 3:
66     .align 8
67 4:

```

Fig 10: Dot_Prod_Pntr.s (Code) [Compiler Generated Optimization]

Simulation

Intel x86 (Visual Studio Code)

Dot Product (Pointer) [Non-Optimized]

Note, for all simulations, we are running the dot product 10 times using a for loop. This is so we can account for error margins in execution time and get a precise measure by averaging them. We are accounting for array size N = 16 ... 2^16 and noting the execution time in seconds.

```
C:\Users\lzyy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\Dot_Product\Debug
Array size: 16
Run number: 1
Running time: 4e-07 secs
Run number: 2
Running time: 2e-07 secs
Run number: 3
Running time: 1e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 1e-07 secs
Run number: 7
Running time: 2e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 1e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.5e-07
Press any key to continue . . .
```

Fig 11: Dot_Prod_Pntr (16) w/o Qpar and arch

```
C:\Users\lzyy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\Dot_Product\Debug
Array size: 32
Run number: 1
Running time: 3e-07 secs
Run number: 2
Running time: 2e-07 secs
Run number: 3
Running time: 1e-07 secs
Run number: 4
Running time: 2e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 2e-07 secs
Run number: 7
Running time: 2e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 2e-07 secs
Run number: 10
Running time: 2e-07 secs
Average time: 1.8e-07
Press any key to continue . . .
```

Fig 11a: Dot_Prod_Pntr (32) w/o Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\Dot_Product\Debu...
Array size: 64
Run number: 1
Running time: 3e-07 secs
Run number: 2
Running time: 3e-07 secs
Run number: 3
Running time: 2e-07 secs
Run number: 4
Running time: 2e-07 secs
Run number: 5
Running time: 2e-07 secs
Run number: 6
Running time: 3e-07 secs
Run number: 7
Running time: 3e-07 secs
Run number: 8
Running time: 2e-07 secs
Run number: 9
Running time: 3e-07 secs
Run number: 10
Running time: 2e-07 secs
Average time: 2.5e-07
Press any key to continue . . .
```

Fig 11b: Dot_Prod_Pntr (64) w/o Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\Dot_Product\Debu...
Array size: 128
Run number: 1
Running time: 5e-07 secs
Run number: 2
Running time: 5e-07 secs
Run number: 3
Running time: 4e-07 secs
Run number: 4
Running time: 4e-07 secs
Run number: 5
Running time: 4e-07 secs
Run number: 6
Running time: 4e-07 secs
Run number: 7
Running time: 4e-07 secs
Run number: 8
Running time: 5e-07 secs
Run number: 9
Running time: 4e-07 secs
Run number: 10
Running time: 5e-07 secs
Average time: 4.4e-07
Press any key to continue . . .
```

Fig 11c: Dot_Prod_Pntr (128) w/o Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\Dot_Product\Debu...
Array size: 256
Run number: 1
Running time: 9e-07 secs
Run number: 2
Running time: 8e-07 secs
Run number: 3
Running time: 8e-07 secs
Run number: 4
Running time: 8e-07 secs
Run number: 5
Running time: 8e-07 secs
Run number: 6
Running time: 8e-07 secs
Run number: 7
Running time: 8e-07 secs
Run number: 8
Running time: 8e-07 secs
Run number: 9
Running time: 8e-07 secs
Run number: 10
Running time: 8e-07 secs
Average time: 8.1e-07
Press any key to continue . . .
```

Fig 11d: Dot_Prod_Pntr (256) w/o Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\Dot_Product\Debu...
Array size: 512
Run number: 1
Running time: 1.5e-06 secs
Run number: 2
Running time: 1.5e-06 secs
Run number: 3
Running time: 1.5e-06 secs
Run number: 4
Running time: 1.5e-06 secs
Run number: 5
Running time: 1.5e-06 secs
Run number: 6
Running time: 1.5e-06 secs
Run number: 7
Running time: 1.5e-06 secs
Run number: 8
Running time: 1.5e-06 secs
Run number: 9
Running time: 1.5e-06 secs
Run number: 10
Running time: 1.5e-06 secs
Average time: 1.5e-06
Press any key to continue . . .
```

Fig 11e: Dot_Prod_Pntr (512) w/o Qpar and arch

```
Array size: 1024
Run number: 1
Running time: 3e-06 secs
Run number: 2
Running time: 2.8e-06 secs
Run number: 3
Running time: 2.9e-06 secs
Run number: 4
Running time: 2.8e-06 secs
Run number: 5
Running time: 2.9e-06 secs
Run number: 6
Running time: 2.9e-06 secs
Run number: 7
Running time: 2.8e-06 secs
Run number: 8
Running time: 2.9e-06 secs
Run number: 9
Running time: 2.9e-06 secs
Run number: 10
Running time: 2.9e-06 secs
Average time: 2.88e-06
Press any key to continue . . .
```

Fig 11f: Dot_Prod_Pntr (1024) w/o Qpar and arch

```
Array size: 2048
Run number: 1
Running time: 5.9e-06 secs
Run number: 2
Running time: 5.8e-06 secs
Run number: 3
Running time: 5.7e-06 secs
Run number: 4
Running time: 5.8e-06 secs
Run number: 5
Running time: 5.8e-06 secs
Run number: 6
Running time: 5.7e-06 secs
Run number: 7
Running time: 5.8e-06 secs
Run number: 8
Running time: 5.8e-06 secs
Run number: 9
Running time: 5.8e-06 secs
Run number: 10
Running time: 5.7e-06 secs
Average time: 5.78e-06
Press any key to continue . . .
```

Fig 11g: Dot_Prod_Pntr (2^{11}) w/o Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\Dot_Product\Debu...
Array size: 4096
Run number: 1
Running time: 1.13e-05 secs
Run number: 2
Running time: 1.13e-05 secs
Run number: 3
Running time: 1.13e-05 secs
Run number: 4
Running time: 1.12e-05 secs
Run number: 5
Running time: 1.14e-05 secs
Run number: 6
Running time: 1.13e-05 secs
Run number: 7
Running time: 1.12e-05 secs
Run number: 8
Running time: 1.12e-05 secs
Run number: 9
Running time: 1.32e-05 secs
Run number: 10
Running time: 1.12e-05 secs
Average time: 1.146e-05
Press any key to continue . . .
```

Fig 11h: Dot_Prod_Pntr (2^{12}) w/o Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\Dot_Product\Debu...
Array size: 8192
Run number: 1
Running time: 2.26e-05 secs
Run number: 2
Running time: 2.26e-05 secs
Run number: 3
Running time: 2.24e-05 secs
Run number: 4
Running time: 2.24e-05 secs
Run number: 5
Running time: 2.24e-05 secs
Run number: 6
Running time: 2.24e-05 secs
Run number: 7
Running time: 2.24e-05 secs
Run number: 8
Running time: 2.25e-05 secs
Run number: 9
Running time: 2.24e-05 secs
Run number: 10
Running time: 2.3e-05 secs
Average time: 2.251e-05
Press any key to continue . . .
```

Fig 11i: Dot_Prod_Pntr (2^{13}) w/o Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\Dot_Product\Debu...
Array size: 16384
Run number: 1
Running time: 4.46e-05 secs
Run number: 2
Running time: 4.47e-05 secs
Run number: 3
Running time: 4.47e-05 secs
Run number: 4
Running time: 4.44e-05 secs
Run number: 5
Running time: 4.45e-05 secs
Run number: 6
Running time: 4.46e-05 secs
Run number: 7
Running time: 4.45e-05 secs
Run number: 8
Running time: 4.47e-05 secs
Run number: 9
Running time: 4.45e-05 secs
Run number: 10
Running time: 4.49e-05 secs
Average time: 4.461e-05
Press any key to continue . . .
```

Fig 11j: Dot_Prod_Pntr (2^{14}) w/o Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\Dot_Product\Debu...
Array size: 32768
Run number: 1
Running time: 9.02e-05 secs
Run number: 2
Running time: 9.17e-05 secs
Run number: 3
Running time: 8.99e-05 secs
Run number: 4
Running time: 9.02e-05 secs
Run number: 5
Running time: 9.03e-05 secs
Run number: 6
Running time: 8.99e-05 secs
Run number: 7
Running time: 8.99e-05 secs
Run number: 8
Running time: 9.04e-05 secs
Run number: 9
Running time: 9.04e-05 secs
Run number: 10
Running time: 8.99e-05 secs
Average time: 9.028e-05
Press any key to continue . . .
```

Fig 11k: Dot_Prod_Pntr (2^{15}) w/o Qpar and arch

```
Array size: 65536
Run number: 1
Running time: 0.0001781 secs
Run number: 2
Running time: 0.0001804 secs
Run number: 3
Running time: 0.000179 secs
Run number: 4
Running time: 0.0001802 secs
Run number: 5
Running time: 0.000179 secs
Run number: 6
Running time: 0.0001789 secs
Run number: 7
Running time: 0.0001788 secs
Run number: 8
Running time: 0.0001836 secs
Run number: 9
Running time: 0.0001791 secs
Run number: 10
Running time: 0.000179 secs
Average time: 0.00017961
Press any key to continue . . .
```

Fig 11l: Dot_Prod_Pntr (2^{16}) w/o Qpar and arch

Dot Product (Pointer) [Compiler Generated Optimization]
Optimizations enabled.

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\Dot_Product\Debu...
Array size: 16
Run number: 1
Running time: 3e-07 secs
Run number: 2
Running time: 1e-07 secs
Run number: 3
Running time: 2e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 2e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 0 secs
Run number: 9
Running time: 2e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.4e-07
Press any key to continue . . .
```

Fig 12: Dot_Prod_Pntr (16) w/ Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\Dot_Product\Debu...
Array size: 32
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 1e-07 secs
Run number: 3
Running time: 1e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 1e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 1e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.1e-07
Press any key to continue . . .
```

Fig 12a: Dot_Prod_Pntr (32) w/ Qpar and arch

```
Array size: 64
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 2e-07 secs
Run number: 3
Running time: 2e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 1e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 2e-07 secs
Run number: 9
Running time: 2e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.5e-07
Press any key to continue . . .
```

Fig 12b: Dot_Prod_Pntr (64) w/ Qpar and arch

```
Array size: 128
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 2e-07 secs
Run number: 3
Running time: 2e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 2e-07 secs
Run number: 6
Running time: 2e-07 secs
Run number: 7
Running time: 2e-07 secs
Run number: 8
Running time: 2e-07 secs
Run number: 9
Running time: 2e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.8e-07
Press any key to continue . . .
```

Fig 12c: Dot_Prod_Pntr (128) w/ Qpar and arch

```
Array size: 256
Run number: 1
Running time: 3e-07 secs
Run number: 2
Running time: 3e-07 secs
Run number: 3
Running time: 3e-07 secs
Run number: 4
Running time: 2e-07 secs
Run number: 5
Running time: 3e-07 secs
Run number: 6
Running time: 2e-07 secs
Run number: 7
Running time: 2e-07 secs
Run number: 8
Running time: 3e-07 secs
Run number: 9
Running time: 3e-07 secs
Run number: 10
Running time: 3e-07 secs
Average time: 2.7e-07
Press any key to continue . . .
```

Fig 12d: Dot_Prod_Pntr (256) w/ Qpar and arch

```
Array size: 512
Run number: 1
Running time: 5e-07 secs
Run number: 2
Running time: 5e-07 secs
Run number: 3
Running time: 4e-07 secs
Run number: 4
Running time: 5e-07 secs
Run number: 5
Running time: 5e-07 secs
Run number: 6
Running time: 4e-07 secs
Run number: 7
Running time: 5e-07 secs
Run number: 8
Running time: 4e-07 secs
Run number: 9
Running time: 4e-07 secs
Run number: 10
Running time: 4e-07 secs
Average time: 4.5e-07
Press any key to continue . . .
```

Fig 12e: Dot_Prod_Pntr (512) w/ Qpar and arch

```
Array size: 1024
Run number: 1
Running time: 9e-07 secs
Run number: 2
Running time: 8e-07 secs
Run number: 3
Running time: 8e-07 secs
Run number: 4
Running time: 8e-07 secs
Run number: 5
Running time: 8e-07 secs
Run number: 6
Running time: 1.1e-06 secs
Run number: 7
Running time: 8e-07 secs
Run number: 8
Running time: 9e-07 secs
Run number: 9
Running time: 8e-07 secs
Run number: 10
Running time: 8e-07 secs
Average time: 8.5e-07
Press any key to continue . . .
```

Fig 12f: Dot_Prod_Pntr (1024) w/ Qpar and arch

```
Array size: 2048
Run number: 1
Running time: 1.7e-06 secs
Run number: 2
Running time: 1.8e-06 secs
Run number: 3
Running time: 1.6e-06 secs
Run number: 4
Running time: 1.6e-06 secs
Run number: 5
Running time: 1.6e-06 secs
Run number: 6
Running time: 1.6e-06 secs
Run number: 7
Running time: 1.6e-06 secs
Run number: 8
Running time: 1.6e-06 secs
Run number: 9
Running time: 1.5e-06 secs
Run number: 10
Running time: 1.5e-06 secs
Average time: 1.61e-06
Press any key to continue . . .
```

Fig 12g: Dot_Prod_Pntr (2^11) w/ Qpar and arch

```
Array size: 4096
Run number: 1
Running time: 3.3e-06 secs
Run number: 2
Running time: 3.2e-06 secs
Run number: 3
Running time: 3e-06 secs
Run number: 4
Running time: 3.1e-06 secs
Run number: 5
Running time: 3.1e-06 secs
Run number: 6
Running time: 3e-06 secs
Run number: 7
Running time: 3e-06 secs
Run number: 8
Running time: 3e-06 secs
Run number: 9
Running time: 3e-06 secs
Run number: 10
Running time: 3e-06 secs
Average time: 3.07e-06
Press any key to continue . . .
```

Fig 12h: Dot_Prod_Pntr (2^{12}) w/ Qpar and arch

```
Array size: 8192
Run number: 1
Running time: 6e-06 secs
Run number: 2
Running time: 5.9e-06 secs
Run number: 3
Running time: 6e-06 secs
Run number: 4
Running time: 5.9e-06 secs
Run number: 5
Running time: 6e-06 secs
Run number: 6
Running time: 5.9e-06 secs
Run number: 7
Running time: 5.9e-06 secs
Run number: 8
Running time: 6e-06 secs
Run number: 9
Running time: 5.9e-06 secs
Run number: 10
Running time: 5.9e-06 secs
Average time: 5.94e-06
Press any key to continue . . .
```

Fig 12i: Dot_Prod_Pntr (2^{13}) w/ Qpar and arch

```
Array size: 16384
Run number: 1
Running time: 1.19e-05 secs
Run number: 2
Running time: 1.19e-05 secs
Run number: 3
Running time: 1.19e-05 secs
Run number: 4
Running time: 1.19e-05 secs
Run number: 5
Running time: 1.19e-05 secs
Run number: 6
Running time: 1.19e-05 secs
Run number: 7
Running time: 1.18e-05 secs
Run number: 8
Running time: 1.19e-05 secs
Run number: 9
Running time: 1.19e-05 secs
Run number: 10
Running time: 1.18e-05 secs
Average time: 1.19e-05
Press any key to continue . . .
```

Fig 12j: Dot_Prod_Pntr (2¹⁴) w/ Qpar and arch

```
Array size: 32768
Run number: 1
Running time: 2.46e-05 secs
Run number: 2
Running time: 2.47e-05 secs
Run number: 3
Running time: 2.49e-05 secs
Run number: 4
Running time: 2.41e-05 secs
Run number: 5
Running time: 2.41e-05 secs
Run number: 6
Running time: 2.4e-05 secs
Run number: 7
Running time: 2.38e-05 secs
Run number: 8
Running time: 2.38e-05 secs
Run number: 9
Running time: 2.4e-05 secs
Run number: 10
Running time: 2.39e-05 secs
Average time: 2.419e-05
Press any key to continue . . .
```

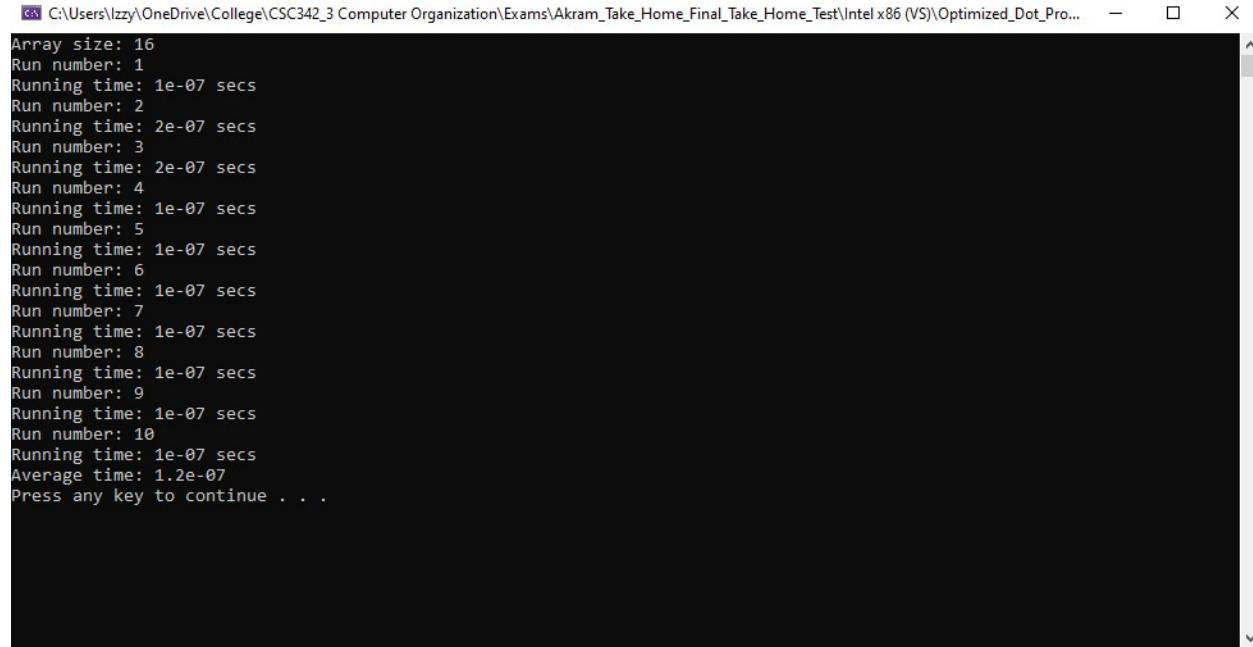
Fig 12k: Dot_Prod_Pntr (2¹⁵) w/ Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\Dot_Product\Debu...
Array size: 65536
Run number: 1
Running time: 4.83e-05 secs
Run number: 2
Running time: 4.78e-05 secs
Run number: 3
Running time: 4.79e-05 secs
Run number: 4
Running time: 4.85e-05 secs
Run number: 5
Running time: 4.75e-05 secs
Run number: 6
Running time: 4.77e-05 secs
Run number: 7
Running time: 4.8e-05 secs
Run number: 8
Running time: 5.25e-05 secs
Run number: 9
Running time: 4.78e-05 secs
Run number: 10
Running time: 4.78e-05 secs
Average time: 4.838e-05
Press any key to continue . . .
```

Fig 12l: Dot_Prod_Pntr (2^{16}) w/ Qpar and arch

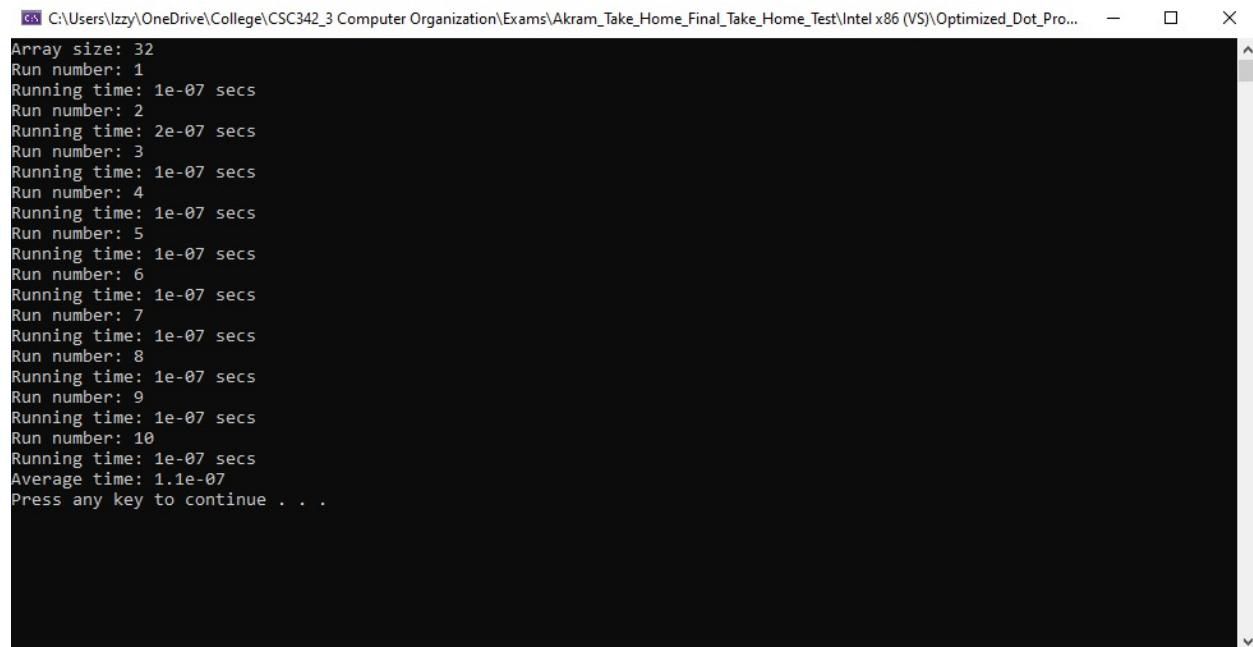
Dot Product (Pointer) [Manually Optimized]

Manually optimized compiler generated .asm file



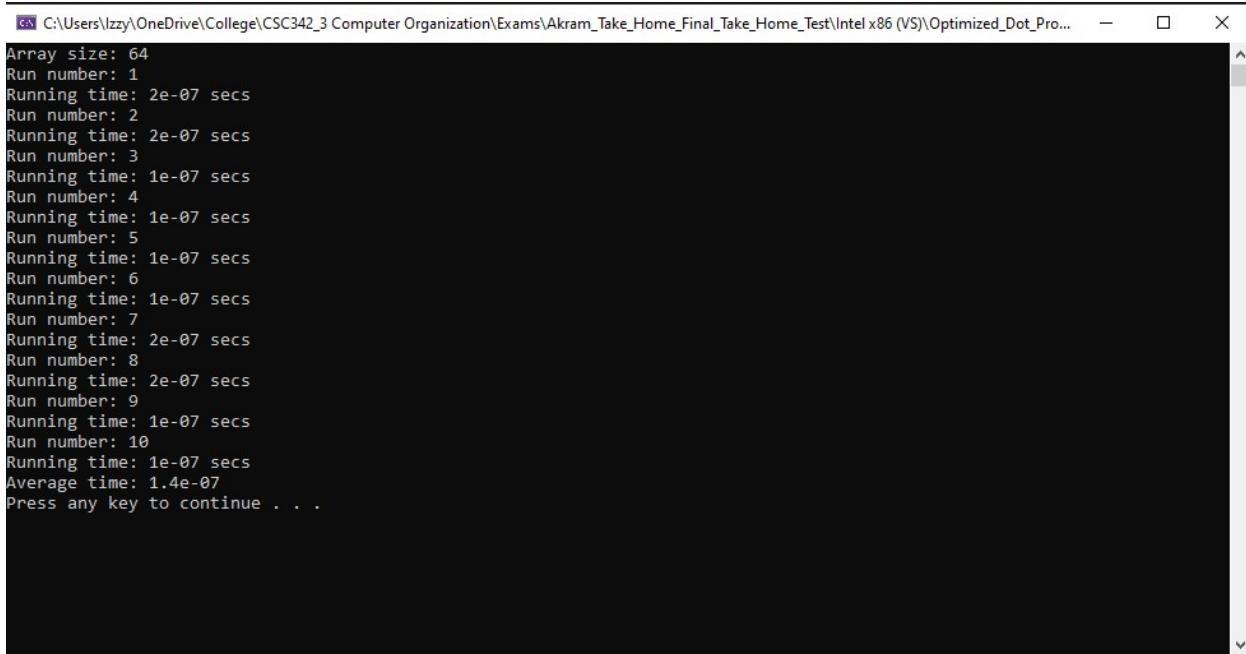
```
C:\Users\lzyy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\Optimized_Dot_Pro...
Array size: 16
Run number: 1
Running time: 1e-07 secs
Run number: 2
Running time: 2e-07 secs
Run number: 3
Running time: 2e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 1e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 1e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.2e-07
Press any key to continue . . .
```

Fig 13: Op_Dot_Prod_Pntr (16) w/ Qpar and arch & Manual optimization



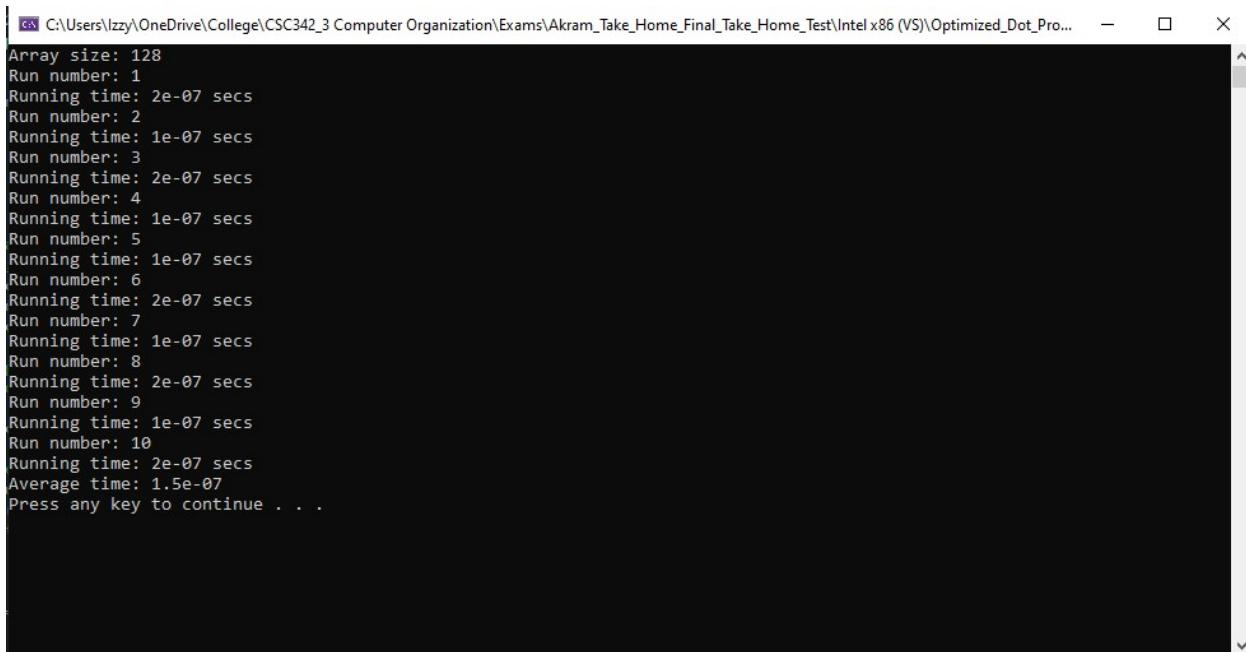
```
C:\Users\lzyy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\Optimized_Dot_Pro...
Array size: 32
Run number: 1
Running time: 1e-07 secs
Run number: 2
Running time: 2e-07 secs
Run number: 3
Running time: 1e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 1e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 1e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.1e-07
Press any key to continue . . .
```

Fig 13a: Op_Dot_Prod_Pntr (32) w/ Qpar and arch & Manual optimization



```
Array size: 64
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 2e-07 secs
Run number: 3
Running time: 1e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 1e-07 secs
Run number: 7
Running time: 2e-07 secs
Run number: 8
Running time: 2e-07 secs
Run number: 9
Running time: 1e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.4e-07
Press any key to continue . . .
```

Fig 13b: Op_Dot_Prod_Pntr (64) w/ Qpar and arch & Manual optimization



```
Array size: 128
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 1e-07 secs
Run number: 3
Running time: 2e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 2e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 2e-07 secs
Run number: 9
Running time: 1e-07 secs
Run number: 10
Running time: 2e-07 secs
Average time: 1.5e-07
Press any key to continue . . .
```

Fig 13c: Op_Dot_Prod_Pntr (128) w/ Qpar and arch & Manual optimization

```
Array size: 256
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 3e-07 secs
Run number: 3
Running time: 2e-07 secs
Run number: 4
Running time: 3e-07 secs
Run number: 5
Running time: 2e-07 secs
Run number: 6
Running time: 2e-07 secs
Run number: 7
Running time: 2e-07 secs
Run number: 8
Running time: 3e-07 secs
Run number: 9
Running time: 2e-07 secs
Run number: 10
Running time: 2e-07 secs
Average time: 2.3e-07
Press any key to continue . . .
```

Fig 13d: Op_Dot_Prod_Pntr (256) w/ Qpar and arch & Manual optimization

```
Array size: 512
Run number: 1
Running time: 4e-07 secs
Run number: 2
Running time: 4e-07 secs
Run number: 3
Running time: 5e-07 secs
Run number: 4
Running time: 5e-07 secs
Run number: 5
Running time: 5e-07 secs
Run number: 6
Running time: 4e-07 secs
Run number: 7
Running time: 5e-07 secs
Run number: 8
Running time: 4e-07 secs
Run number: 9
Running time: 5e-07 secs
Run number: 10
Running time: 5e-07 secs
Average time: 4.6e-07
Press any key to continue . . .
```

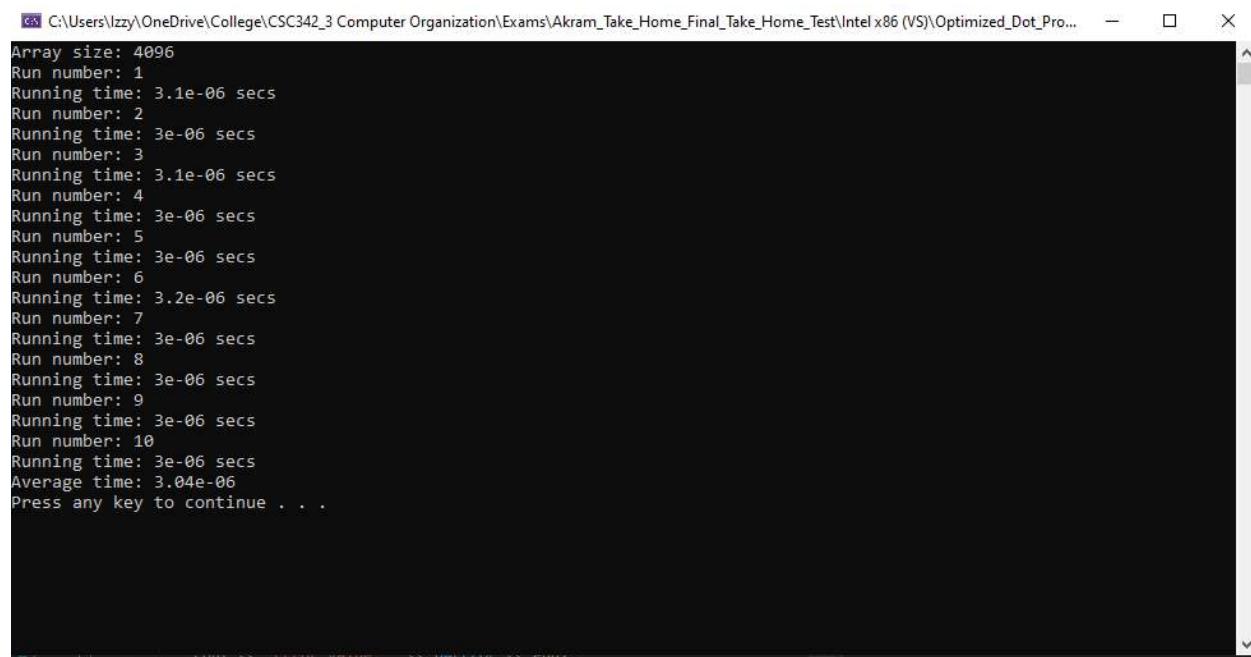
Fig 13e: Op_Dot_Prod_Pntr (512) w/ Qpar and arch & Manual optimization

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\Optimized_Dot_Pro...
Array size: 1024
Run number: 1
Running time: 1.3e-06 secs
Run number: 2
Running time: 1e-06 secs
Run number: 3
Running time: 8e-07 secs
Run number: 4
Running time: 8e-07 secs
Run number: 5
Running time: 8e-07 secs
Run number: 6
Running time: 8e-07 secs
Run number: 7
Running time: 7e-07 secs
Run number: 8
Running time: 8e-07 secs
Run number: 9
Running time: 8e-07 secs
Run number: 10
Running time: 8e-07 secs
Average time: 8.6e-07
Press any key to continue . . .
```

Fig 13f: *Op_Dot_Prod_Pntr (1024) w/ Qpar and arch & Manual optimization*

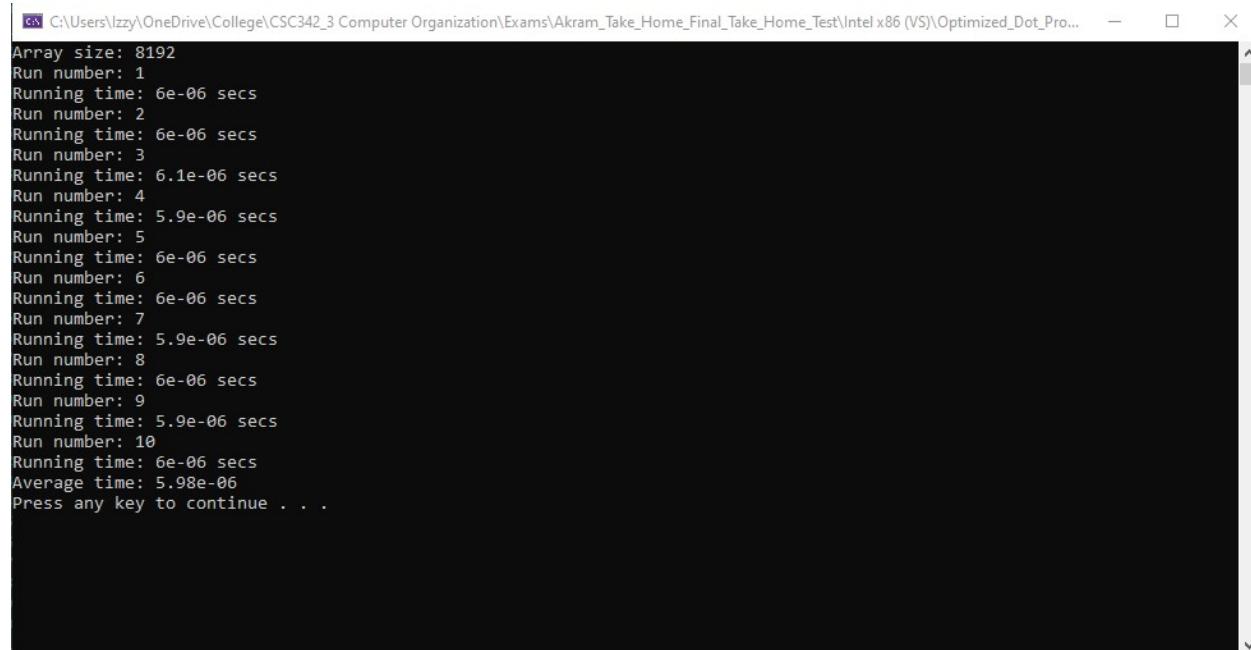
```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\Optimized_Dot_Pro...
Array size: 2048
Run number: 1
Running time: 1.6e-06 secs
Run number: 2
Running time: 1.5e-06 secs
Run number: 3
Running time: 1.6e-06 secs
Run number: 4
Running time: 1.5e-06 secs
Run number: 5
Running time: 1.5e-06 secs
Run number: 6
Running time: 1.5e-06 secs
Run number: 7
Running time: 1.5e-06 secs
Run number: 8
Running time: 1.6e-06 secs
Run number: 9
Running time: 1.6e-06 secs
Run number: 10
Running time: 1.5e-06 secs
Average time: 1.54e-06
Press any key to continue . . .
```

Fig 13g: *Op_Dot_Prod_Pntr (2^11) w/ Qpar and arch & Manual optimization*



```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\Optimized_Dot_Pro...
Array size: 4096
Run number: 1
Running time: 3.e-06 secs
Run number: 2
Running time: 3e-06 secs
Run number: 3
Running time: 3.e-06 secs
Run number: 4
Running time: 3e-06 secs
Run number: 5
Running time: 3e-06 secs
Run number: 6
Running time: 3.2e-06 secs
Run number: 7
Running time: 3e-06 secs
Run number: 8
Running time: 3e-06 secs
Run number: 9
Running time: 3e-06 secs
Run number: 10
Running time: 3e-06 secs
Average time: 3.04e-06
Press any key to continue . . .
```

Fig 13h: *Op_Dot_Prod_Pntr (2^12) w/ Qpar and arch & Manual optimization*



```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\Optimized_Dot_Pro...
Array size: 8192
Run number: 1
Running time: 6e-06 secs
Run number: 2
Running time: 6e-06 secs
Run number: 3
Running time: 6.1e-06 secs
Run number: 4
Running time: 5.9e-06 secs
Run number: 5
Running time: 6e-06 secs
Run number: 6
Running time: 6e-06 secs
Run number: 7
Running time: 5.9e-06 secs
Run number: 8
Running time: 6e-06 secs
Run number: 9
Running time: 5.9e-06 secs
Run number: 10
Running time: 6e-06 secs
Average time: 5.98e-06
Press any key to continue . . .
```

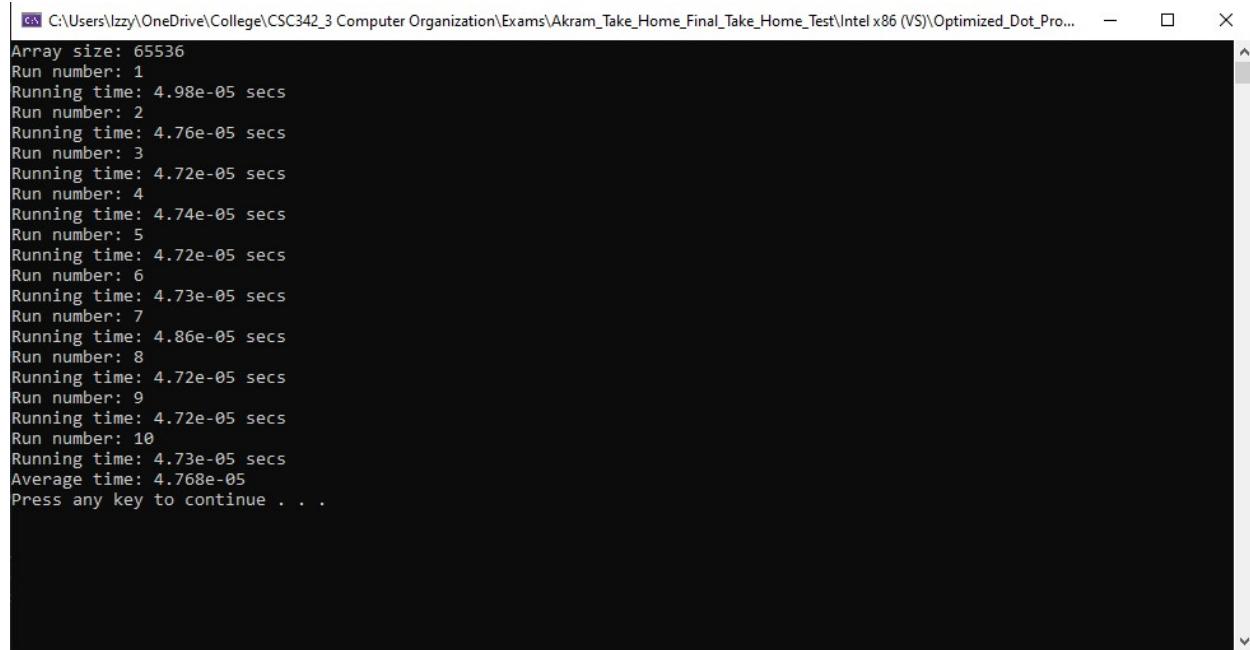
Fig 13i: *Op_Dot_Prod_Pntr (2^13) w/ Qpar and arch & Manual optimization*

```
Array size: 16384
Run number: 1
Running time: 1.19e-05 secs
Run number: 2
Running time: 1.19e-05 secs
Run number: 3
Running time: 1.18e-05 secs
Run number: 4
Running time: 1.19e-05 secs
Run number: 5
Running time: 1.19e-05 secs
Run number: 6
Running time: 1.19e-05 secs
Run number: 7
Running time: 1.18e-05 secs
Run number: 8
Running time: 1.19e-05 secs
Run number: 9
Running time: 1.18e-05 secs
Run number: 10
Running time: 1.19e-05 secs
Average time: 1.187e-05
Press any key to continue . . .
```

Fig 13j: Op_Dot_Prod_Pntr (2^{14}) w/ Qpar and arch & Manual optimization

```
Array size: 32768
Run number: 1
Running time: 2.51e-05 secs
Run number: 2
Running time: 2.44e-05 secs
Run number: 3
Running time: 2.41e-05 secs
Run number: 4
Running time: 2.38e-05 secs
Run number: 5
Running time: 2.39e-05 secs
Run number: 6
Running time: 2.38e-05 secs
Run number: 7
Running time: 2.39e-05 secs
Run number: 8
Running time: 2.38e-05 secs
Run number: 9
Running time: 2.38e-05 secs
Run number: 10
Running time: 2.38e-05 secs
Average time: 2.404e-05
Press any key to continue . . .
```

Fig 13k: Op_Dot_Prod_Pntr (2^{15}) w/ Qpar and arch & Manual optimization

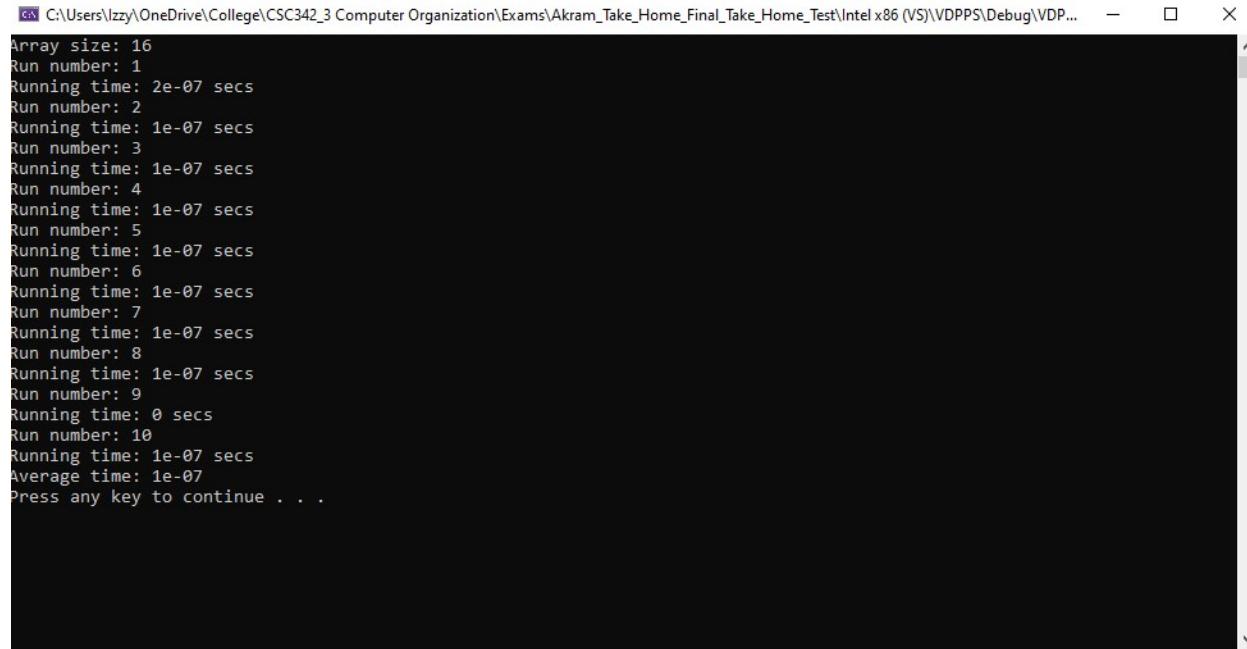


The screenshot shows a terminal window with the following text output:

```
Array size: 65536
Run number: 1
Running time: 4.98e-05 secs
Run number: 2
Running time: 4.76e-05 secs
Run number: 3
Running time: 4.72e-05 secs
Run number: 4
Running time: 4.74e-05 secs
Run number: 5
Running time: 4.72e-05 secs
Run number: 6
Running time: 4.73e-05 secs
Run number: 7
Running time: 4.86e-05 secs
Run number: 8
Running time: 4.72e-05 secs
Run number: 9
Running time: 4.72e-05 secs
Run number: 10
Running time: 4.73e-05 secs
Average time: 4.768e-05
Press any key to continue . . .
```

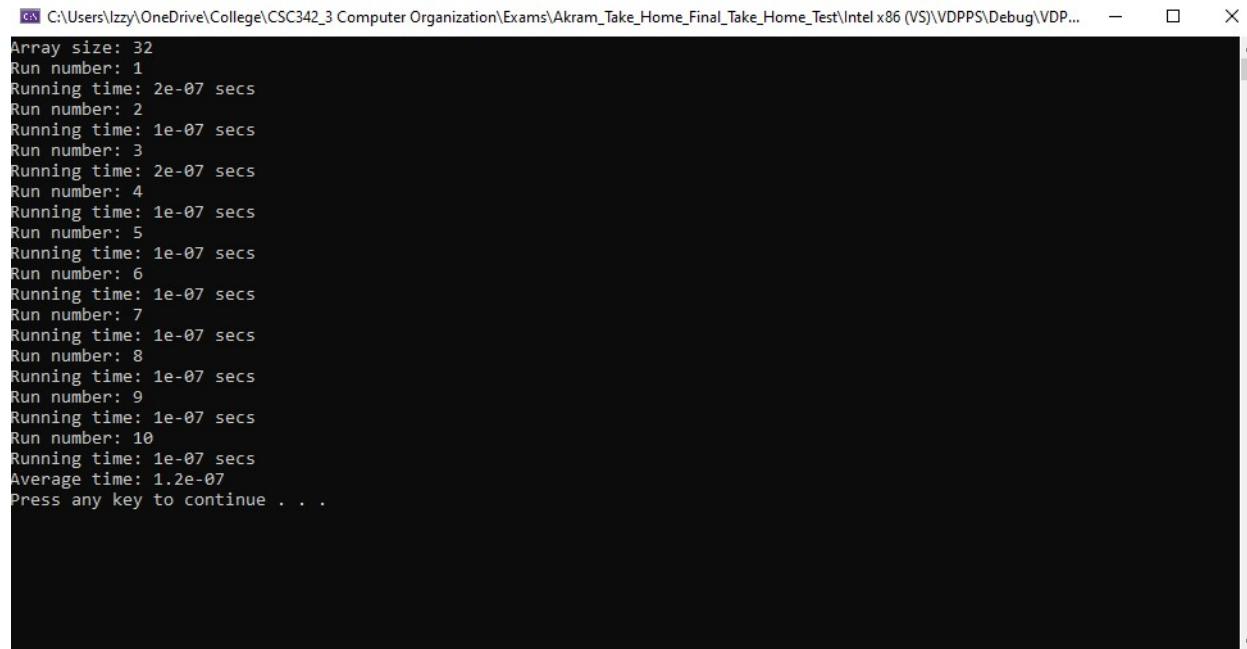
Fig 13l: Op_Dot_Prod_Pntr (2^{16}) w/ Qpar and arch & Manual optimization

VDPPS



```
Array size: 16
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 1e-07 secs
Run number: 3
Running time: 1e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 1e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 0 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1e-07
Press any key to continue . . .
```

Fig 14: VDPPS_Dot_Prod_Pntr (16) w/ Qpar and arch



```
Array size: 32
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 1e-07 secs
Run number: 3
Running time: 2e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 1e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 1e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.2e-07
Press any key to continue . . .
```

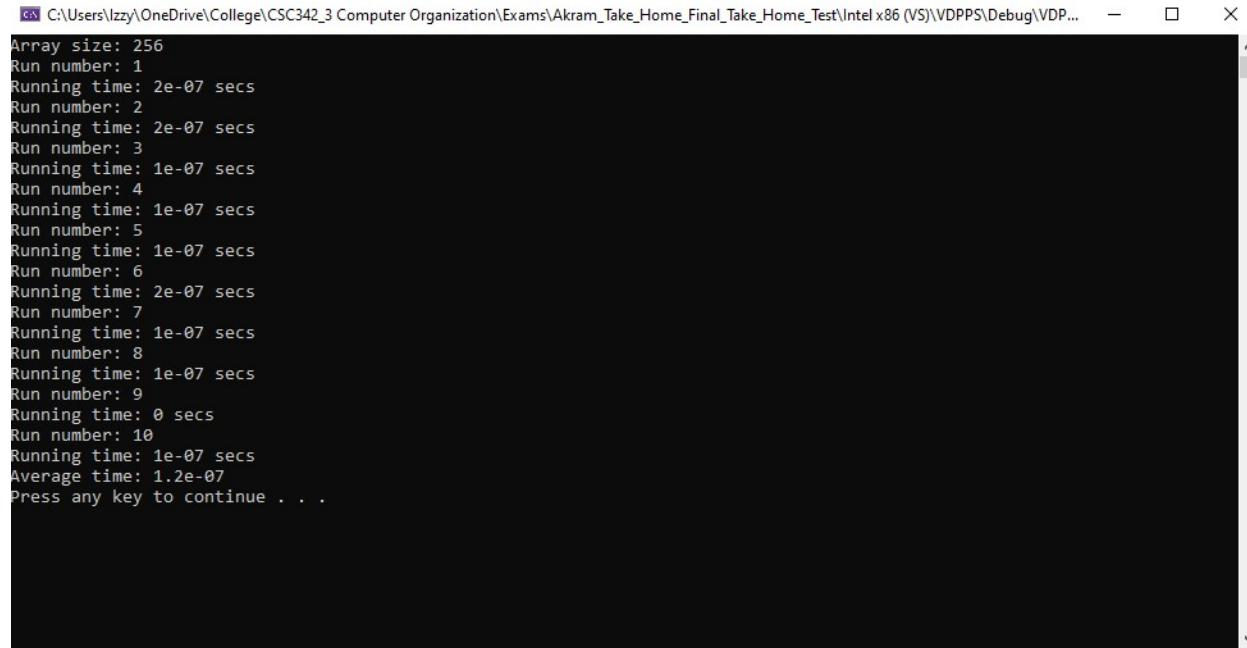
Fig 14a: VDPPS_Dot_Prod_Pntr (32) w/ Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\VDPPS\Debug\VDPPS_Dot_Prod_Pntr.exe
Array size: 64
Run number: 1
Running time: 1e-07 secs
Run number: 2
Running time: 1e-07 secs
Run number: 3
Running time: 1e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 1e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 1e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1e-07
Press any key to continue . . .
```

Fig 14b: VDPPS_Dot_Prod_Pntr (64) w/ Qpar and arch

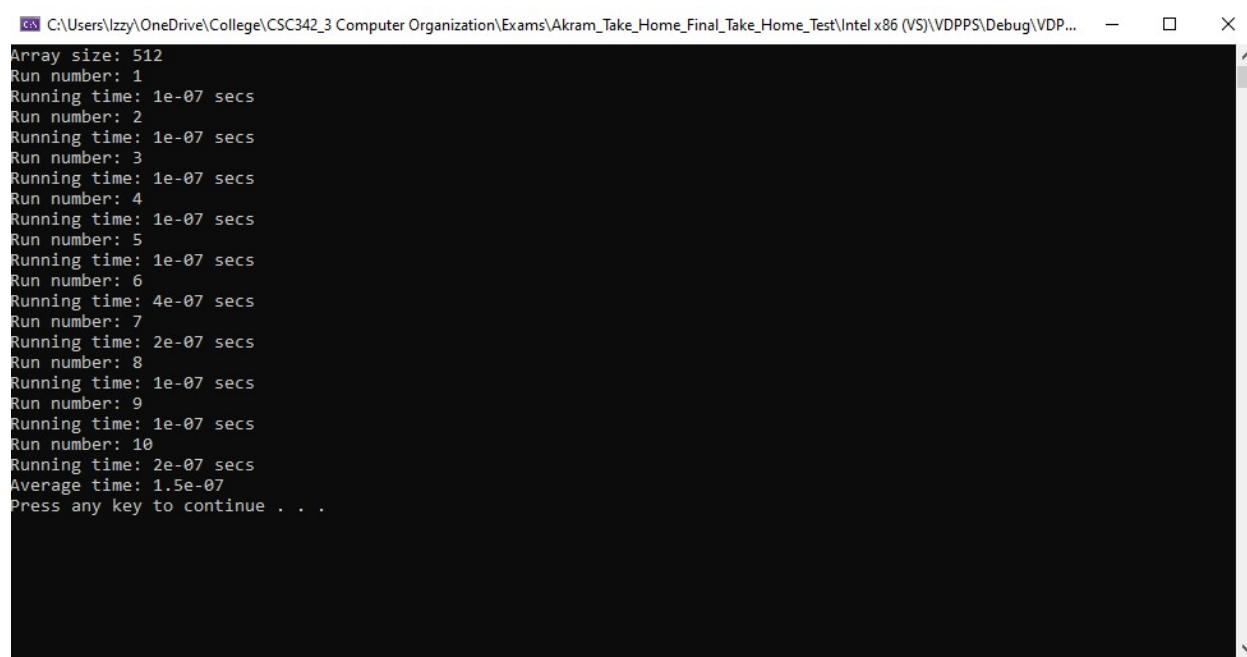
```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\VDPPS\Debug\VDPPS_Dot_Prod_Pntr.exe
Array size: 128
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 1e-07 secs
Run number: 3
Running time: 1e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 2e-07 secs
Run number: 6
Running time: 1e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 1e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.2e-07
Press any key to continue . . .
```

Fig 14c: VDPPS_Dot_Prod_Pntr (128) w/ Qpar and arch



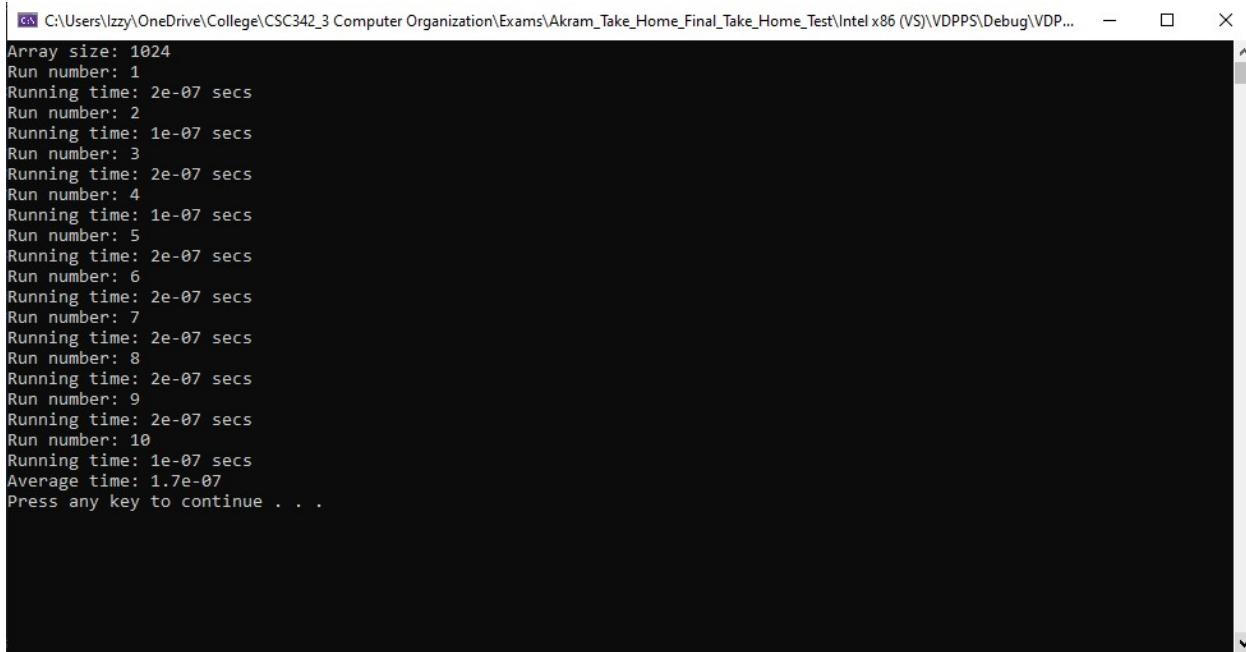
```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\VDPPS\Debug\VDPPS_Dot_Prod_Pntr.exe
Array size: 256
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 2e-07 secs
Run number: 3
Running time: 1e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 2e-07 secs
Run number: 7
Running time: 1e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 0 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.2e-07
Press any key to continue . . .
```

Fig 14d: VDPPS_Dot_Prod_Pntr (256) w/ Qpar and arch



```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\VDPPS\Debug\VDPPS_Dot_Prod_Pntr.exe
Array size: 512
Run number: 1
Running time: 1e-07 secs
Run number: 2
Running time: 1e-07 secs
Run number: 3
Running time: 1e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 1e-07 secs
Run number: 6
Running time: 4e-07 secs
Run number: 7
Running time: 2e-07 secs
Run number: 8
Running time: 1e-07 secs
Run number: 9
Running time: 1e-07 secs
Run number: 10
Running time: 2e-07 secs
Average time: 1.5e-07
Press any key to continue . . .
```

Fig 14e: VDPPS_Dot_Prod_Pntr (512) w/ Qpar and arch



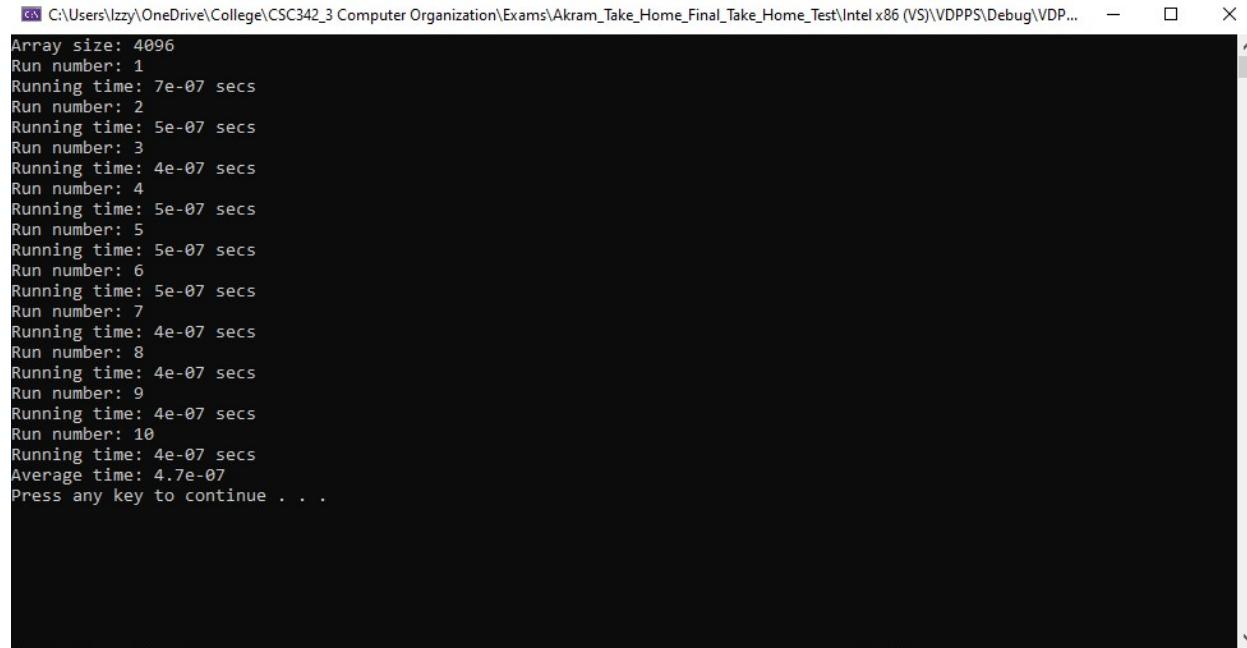
```
C:\Users\lizzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\VDPPS\Debug\VDPPS_Dot_Prod_Pntr.exe
Array size: 1024
Run number: 1
Running time: 2e-07 secs
Run number: 2
Running time: 1e-07 secs
Run number: 3
Running time: 2e-07 secs
Run number: 4
Running time: 1e-07 secs
Run number: 5
Running time: 2e-07 secs
Run number: 6
Running time: 2e-07 secs
Run number: 7
Running time: 2e-07 secs
Run number: 8
Running time: 2e-07 secs
Run number: 9
Running time: 2e-07 secs
Run number: 10
Running time: 1e-07 secs
Average time: 1.7e-07
Press any key to continue . . .
```

Fig 14f: VDPPS_Dot_Prod_Pntr (1024) w/ Qpar and arch



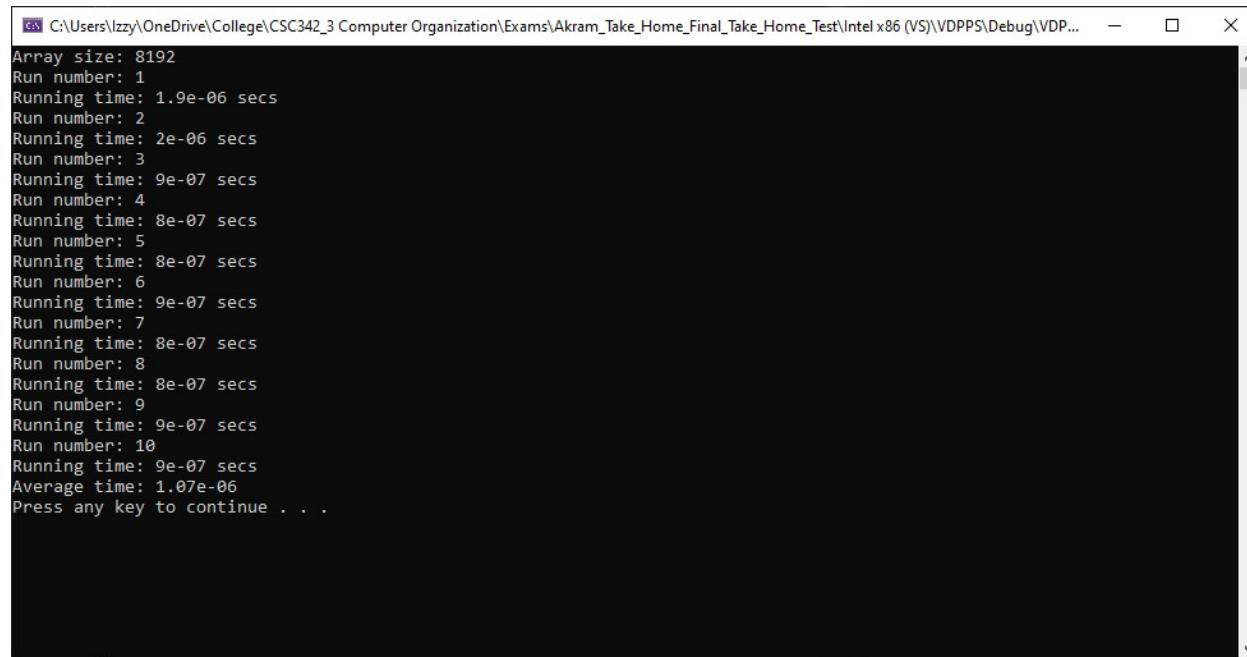
```
C:\Users\lizzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\intel x86 (VS)\VDPPS\Debug\VDPPS_Dot_Prod_Pntr.exe
Array size: 2048
Run number: 1
Running time: 5e-07 secs
Run number: 2
Running time: 4e-07 secs
Run number: 3
Running time: 3e-07 secs
Run number: 4
Running time: 3e-07 secs
Run number: 5
Running time: 3e-07 secs
Run number: 6
Running time: 3e-07 secs
Run number: 7
Running time: 3e-07 secs
Run number: 8
Running time: 3e-07 secs
Run number: 9
Running time: 2e-07 secs
Run number: 10
Running time: 3e-07 secs
Average time: 3.2e-07
Press any key to continue . . .
```

Fig 14g: VDPPS_Dot_Prod_Pntr (2^11) w/ Qpar and arch



```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\VDPPS\Debug\VDPPS_Dot_Prod_Pntr.exe

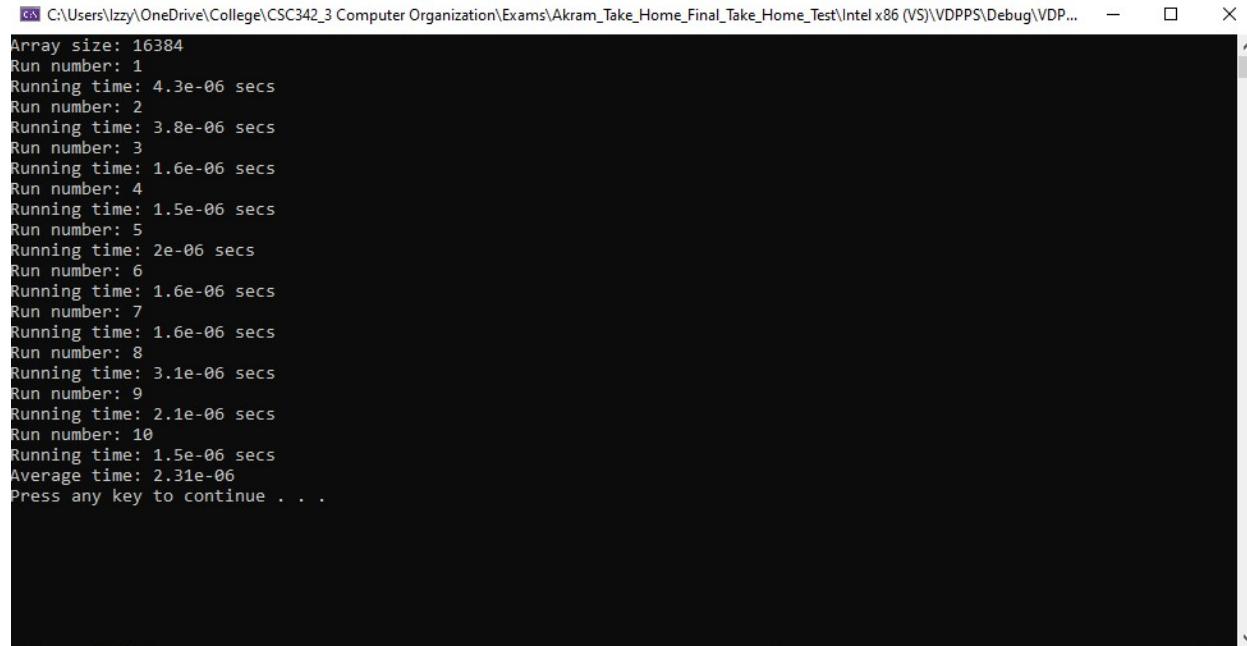
Array size: 4096
Run number: 1
Running time: 7e-07 secs
Run number: 2
Running time: 5e-07 secs
Run number: 3
Running time: 4e-07 secs
Run number: 4
Running time: 5e-07 secs
Run number: 5
Running time: 5e-07 secs
Run number: 6
Running time: 5e-07 secs
Run number: 7
Running time: 4e-07 secs
Run number: 8
Running time: 4e-07 secs
Run number: 9
Running time: 4e-07 secs
Run number: 10
Running time: 4e-07 secs
Average time: 4.7e-07
Press any key to continue . . .
```

Fig 14h: VDPPS_Dot_Prod_Pntr (2^{12}) w/ Qpar and arch

```
C:\Users\Izzy\OneDrive\College\CSC342_3 Computer Organization\Exams\Akram_Take_Home_Final_Take_Home_Test\Intel x86 (VS)\VDPPS\Debug\VDPPS_Dot_Prod_Pntr.exe

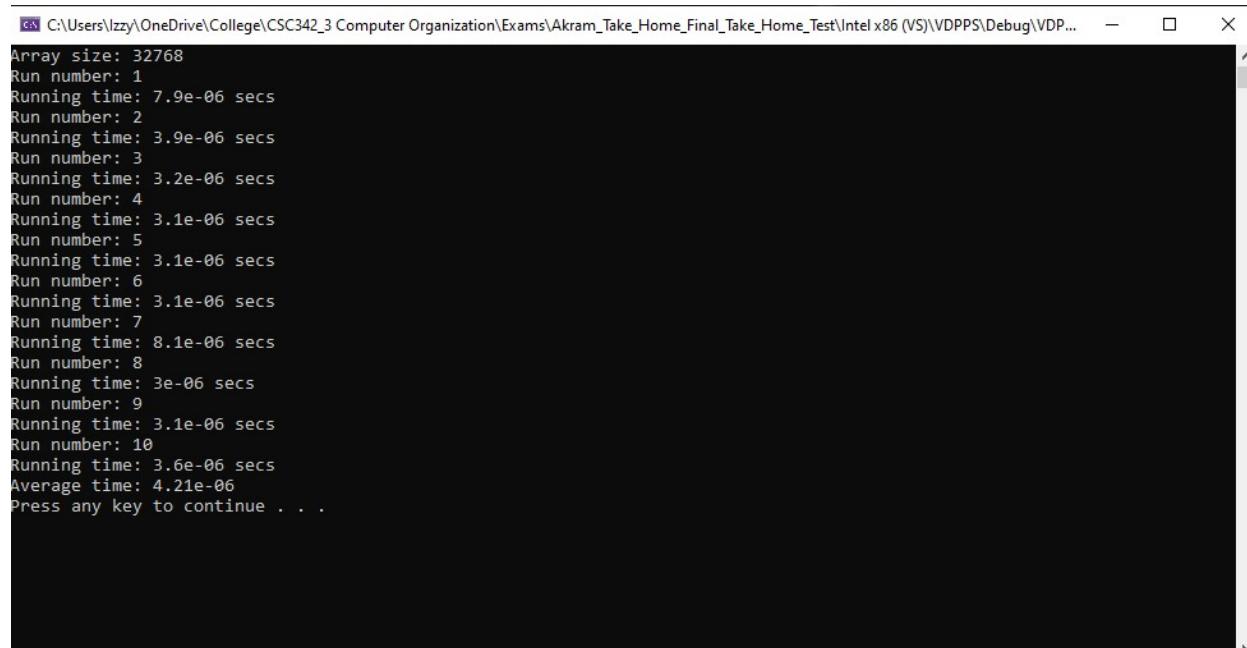
Array size: 8192
Run number: 1
Running time: 1.9e-06 secs
Run number: 2
Running time: 2e-06 secs
Run number: 3
Running time: 9e-07 secs
Run number: 4
Running time: 8e-07 secs
Run number: 5
Running time: 8e-07 secs
Run number: 6
Running time: 9e-07 secs
Run number: 7
Running time: 8e-07 secs
Run number: 8
Running time: 8e-07 secs
Run number: 9
Running time: 9e-07 secs
Run number: 10
Running time: 9e-07 secs
Average time: 1.07e-06
Press any key to continue . . .
```

Fig 14i: VDPPS_Dot_Prod_Pntr (2^{13}) w/ Qpar and arch



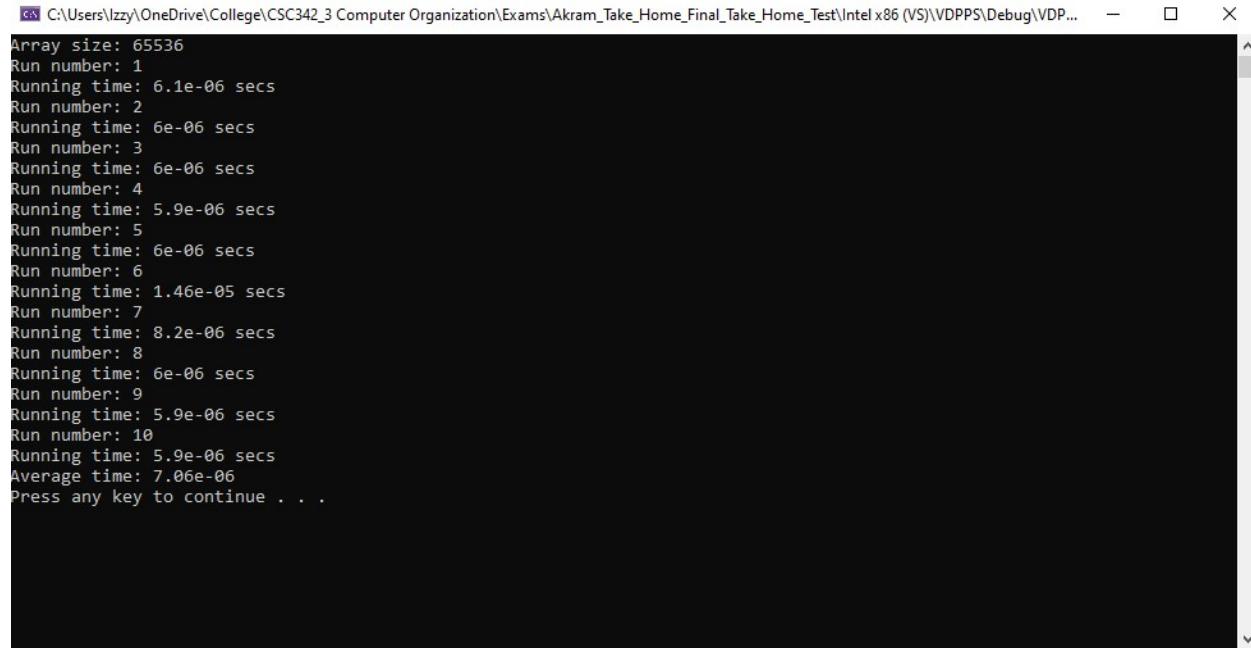
```
Array size: 16384
Run number: 1
Running time: 4.3e-06 secs
Run number: 2
Running time: 3.8e-06 secs
Run number: 3
Running time: 1.6e-06 secs
Run number: 4
Running time: 1.5e-06 secs
Run number: 5
Running time: 2e-06 secs
Run number: 6
Running time: 1.6e-06 secs
Run number: 7
Running time: 1.6e-06 secs
Run number: 8
Running time: 3.1e-06 secs
Run number: 9
Running time: 2.1e-06 secs
Run number: 10
Running time: 1.5e-06 secs
Average time: 2.31e-06
Press any key to continue . . .
```

Fig 14j: VDPPS_Dot_Prod_Pntr (2^{14}) w/ Qpar and arch



```
Array size: 32768
Run number: 1
Running time: 7.9e-06 secs
Run number: 2
Running time: 3.9e-06 secs
Run number: 3
Running time: 3.2e-06 secs
Run number: 4
Running time: 3.1e-06 secs
Run number: 5
Running time: 3.1e-06 secs
Run number: 6
Running time: 3.1e-06 secs
Run number: 7
Running time: 8.1e-06 secs
Run number: 8
Running time: 3e-06 secs
Run number: 9
Running time: 3.1e-06 secs
Run number: 10
Running time: 3.6e-06 secs
Average time: 4.21e-06
Press any key to continue . . .
```

Fig 14k: VDPPS_Dot_Prod_Pntr (2^{15}) w/ Qpar and arch



```
Array size: 65536
Run number: 1
Running time: 6.1e-06 secs
Run number: 2
Running time: 6e-06 secs
Run number: 3
Running time: 6e-06 secs
Run number: 4
Running time: 5.9e-06 secs
Run number: 5
Running time: 6e-06 secs
Run number: 6
Running time: 1.46e-05 secs
Run number: 7
Running time: 8.2e-06 secs
Run number: 8
Running time: 6e-06 secs
Run number: 9
Running time: 5.9e-06 secs
Run number: 10
Running time: 5.9e-06 secs
Average time: 7.06e-06
Press any key to continue . . .
```

Fig 14l: VDPPS_Dot_Prod_Pntr (2^{16}) w/ Qpar and arch

Intel x64 (Linux GCC)

Dot Product (Pointer) [Non-Optimized]

```
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 16
Average time: 296 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 32
Average time: 320 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 64
Average time: 383 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 128
Average time: 571 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 256
Average time: 910 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 512
Average time: 2509 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 1024
Average time: 2958 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 2048
Average time: 7105 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 4096
Average time: 11191 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 8192
Average time: 23876 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 16384
Average time: 45647 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 32768
Average time: 89139 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O0 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 65536
Average time: 181782 ns
```

Figs 15-15l: Dot_Prod_Pntr (-O0 optimization)

Dot Product (Pointer) [Compiler Optimization flags]

```
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 16
Average time: 277 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 32
Average time: 317 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 64
Average time: 378 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 128
Average time: 569 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 256
Average time: 923 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 512
Average time: 1589 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 1024
Average time: 2984 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 2048
Average time: 7525 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 4096
Average time: 11232 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 8192
Average time: 23162 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 16384
Average time: 46657 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 32768
Average time: 89234 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O1 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 65536
Average time: 181430 ns
```

Figs 16-16l: Dot_Prod_Pntr (-O1 optimization)

```
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 16
Average time: 247 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 32
Average time: 292 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 64
Average time: 399 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 128
Average time: 574 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 256
Average time: 915 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 512
Average time: 1580 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 1024
Average time: 2968 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 2048
Average time: 8987 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 4096
Average time: 11243 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 8192
Average time: 24490 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 16384
Average time: 45626 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 32768
Average time: 90930 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -o2 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 65536
Average time: 181395 ns
```

Figs 17-17l: Dot_Prod_Pntr (-o2 optimization)

```
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 16
Average time: 248 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 32
Average time: 292 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 64
Average time: 403 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 128
Average time: 584 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 256
Average time: 930 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 512
Average time: 1591 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 1024
Average time: 2997 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 2048
Average time: 5715 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 4096
Average time: 11315 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 8192
Average time: 22272 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 16384
Average time: 44288 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 32768
Average time: 88366 ns
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ g++ -O3 Source.cpp Dot_Prod_Pntr.cpp -o run
izzy@izzy-VirtualBox:~/Desktop/Intel x64 (Linux)$ ./run
Array size: 65536
Average time: 177113 ns
```

Fig 18-18l: Dot_Prod_Pntr (-O3 optimization)

Results

Array Size	Dot_Prod_Pntr (Non-Opt) time (secs)	Dot_Product_Pntr (Opt) time (secs)	Dot_Prod_Pntr (Man Opt) time (secs)	VDPPS_Dot_Prod_Pntr time (secs)
16	1.50E-07	1.40E-07	1.20E-07	1.00E-07
32	1.80E-07	1.10E-07	1.10E-07	1.20E-07
64	2.50E-07	1.50E-07	1.40E-07	1.00E-07
128	4.40E-07	1.80E-07	1.50E-07	1.20E-07
256	8.10E-07	2.70E-07	2.30E-07	1.20E-07
512	1.50E-06	4.50E-07	4.60E-07	1.50E-07
1024	2.88E-06	8.50E-07	5.10E-07	1.70E-07
2048	5.78E-06	1.61E-06	9.65E-07	3.20E-07
4096	1.15E-05	3.07E-06	1.77E-06	4.70E-07
8192	2.26E-05	5.94E-06	3.51E-06	1.07E-06
16384	4.46E-05	1.19E-05	7.11E-06	2.31E-06
32768	9.03E-05	2.42E-05	1.42E-05	4.21E-06
65536	1.80E-04	4.84E-05	2.77E-05	7.06E-06

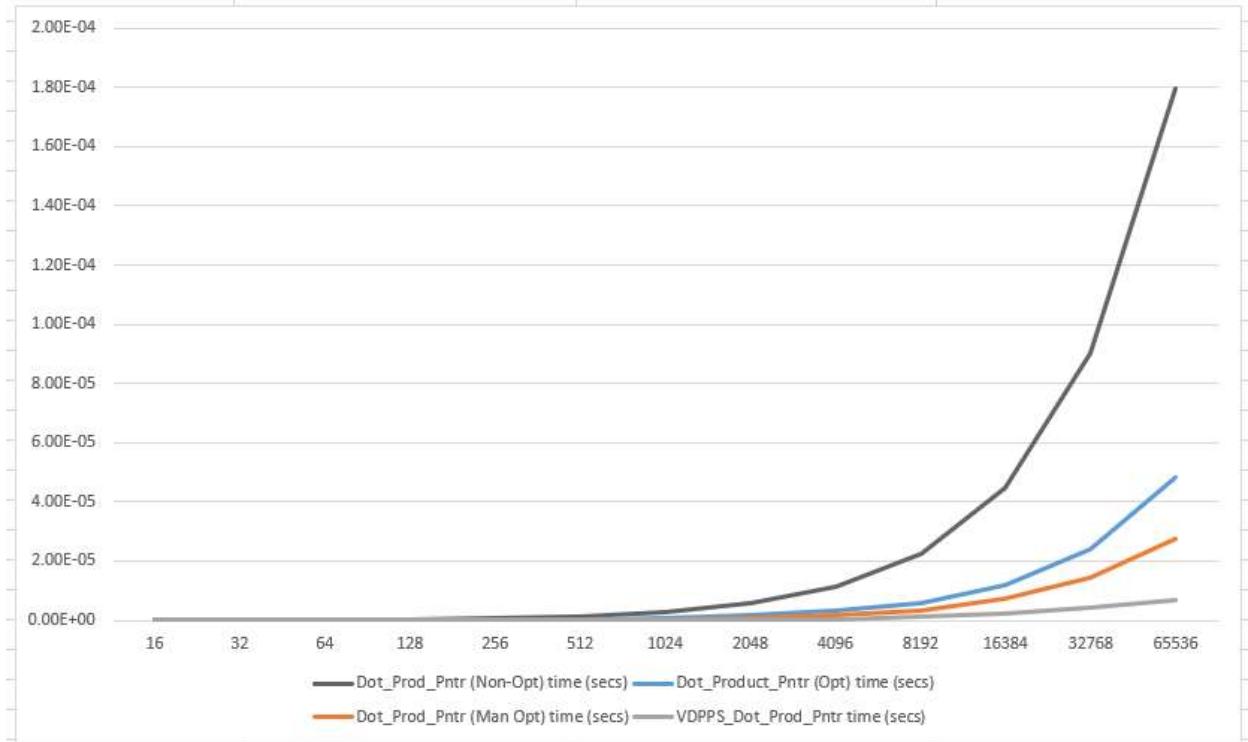


Fig 19: Extraction times comparison of Dot_Prod_Pntr in VS

Array Size	Dot_Prod_Pntr (-o0) time (ns)	Dot_Prod_Pntr (-o1) time (ns)	Dot_Prod_Pntr (-o2) time (ns)	Dot_Prod_Pntr (-o3) time (ns)
16	6.55E-07	5.43E-07	5.14E-07	5.12E-07
32	7.55E-07	6.07E-07	5.63E-07	5.34E-07
64	8.22E-07	8.17E-07	6.40E-07	7.31E-07
128	1.10E-06	1.01E-06	7.18E-07	5.69E-07
256	1.82E-06	1.65E-06	1.42E-06	1.01E-06
512	2.56E-06	2.27E-06	1.88E-06	1.91E-06
1024	4.20E-06	3.58E-06	2.21E-06	2.23E-07
2048	2.55E-05	4.21E-06	3.99E-06	3.23E-06
4096	3.46E-05	2.59E-05	1.65E-05	1.52E-05
8192	6.48E-05	3.28E-05	2.57E-05	3.24E-05
16384	1.22E-04	6.35E-05	4.91E-05	5.03E-05
32768	2.33E-04	1.63E-04	1.26E-04	1.32E-04
65536	4.82E-04	2.83E-04	2.25E-04	2.26E-04

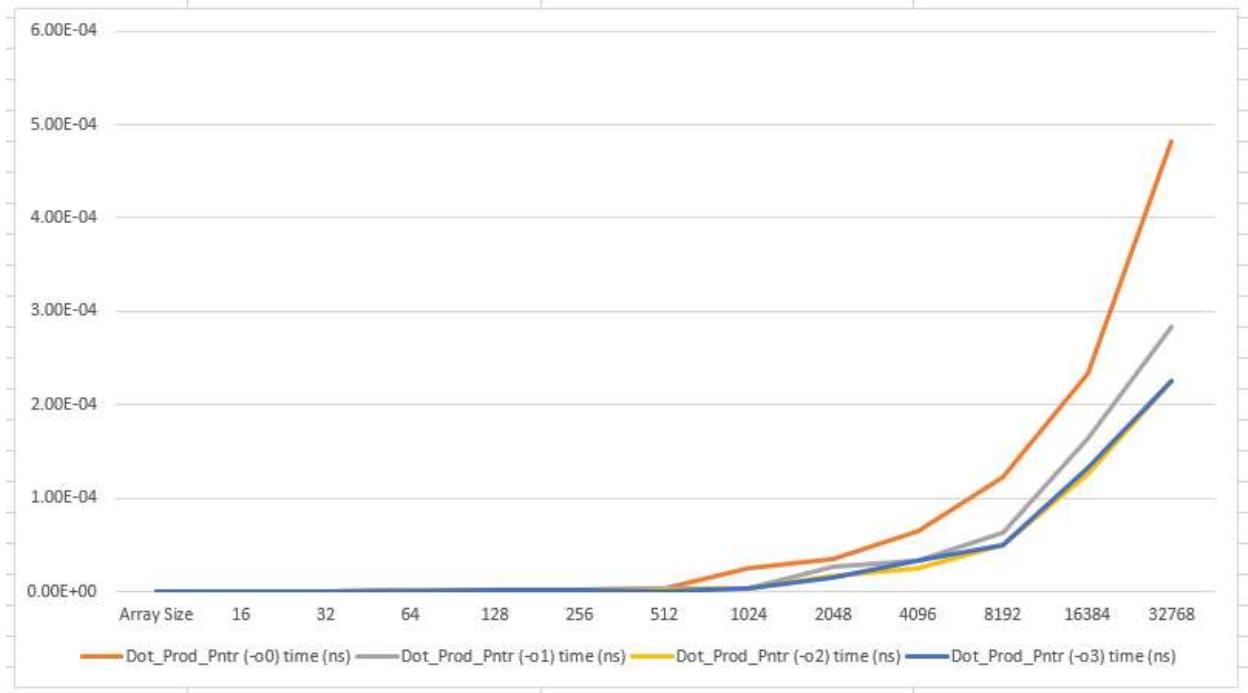


Fig 19: Extraction times comparison of Dot_Prod_Pntr optimization level in Linux

Conclusion

Looking at the execution time outputs from our exercise in VS (figure 18), we note that there is a discernable difference between:

- Dot_Prod_Pnt (Non-Opt)
- Dot_Prod_Pntr (Opt)
- Dot_Prod_Pntr (Man Opt)
- VDPPS_Dot_Prod_Pntr

With the time being fast with each optimization. We notice that our manually optimized assembly is faster than the compiler generated one. But ultimately VDPPS is noticeably faster which isn't surprising.

On the Linux side, we notice a decrease in compilation time with each optimization level:

- -o0
- -o1
- -o2
- -o3

Given these results (figure 19), applying one level of optimization drastically improves the execution time. We can also conclude that optimization level 2 is faster than optimization level 1. However, the difference between optimization levels 2 and 3 are not as drastic, which was expected.

Optimization level 2 is a much better option in reducing the execution time (since there is a tradeoff with memory usage). Optimization level 1 is also a good choice if maximum optimization is not needed.