Индивидуальное задание № 2

«SIMD команды»

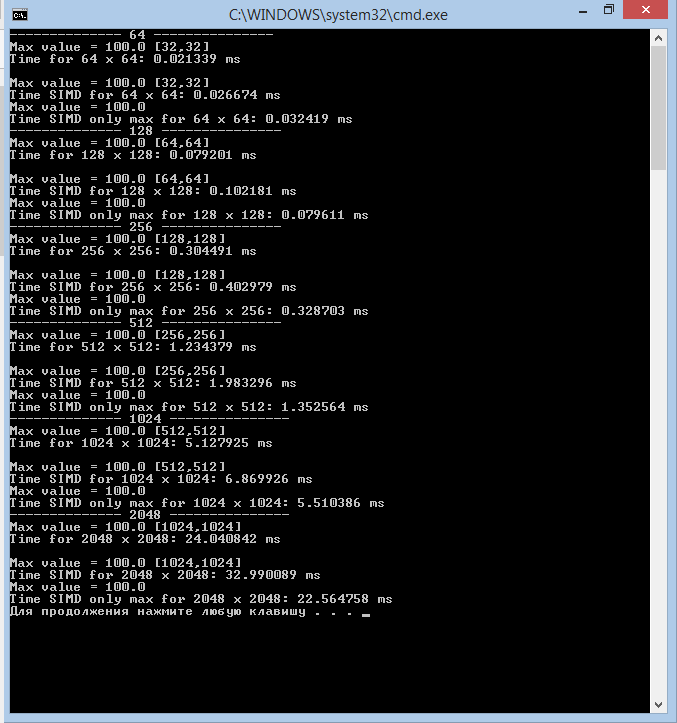
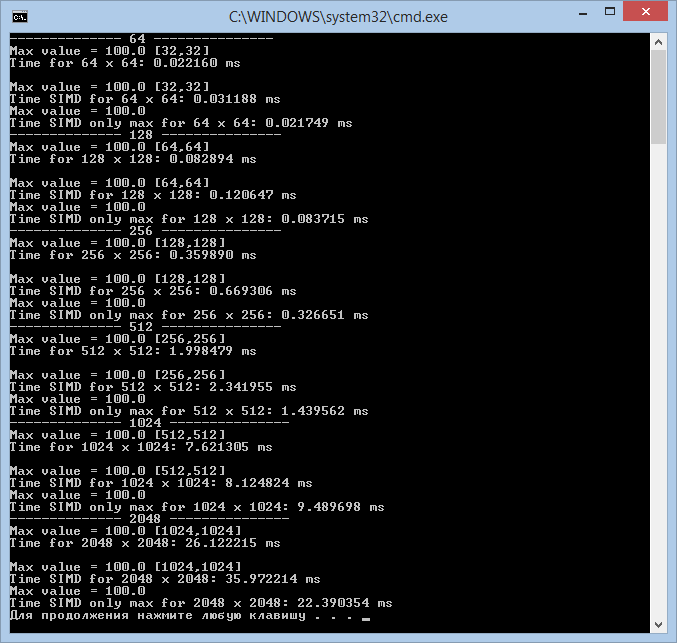
Выполнил:

Ст. гр. ПИ-13-6

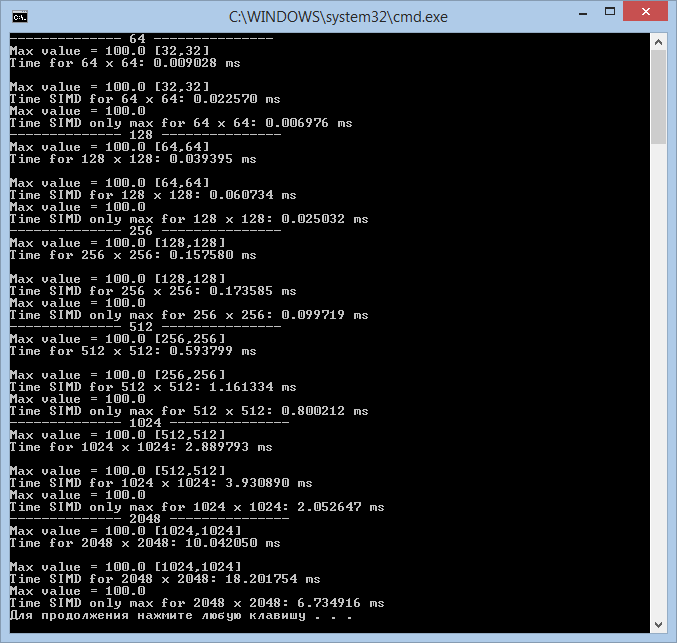
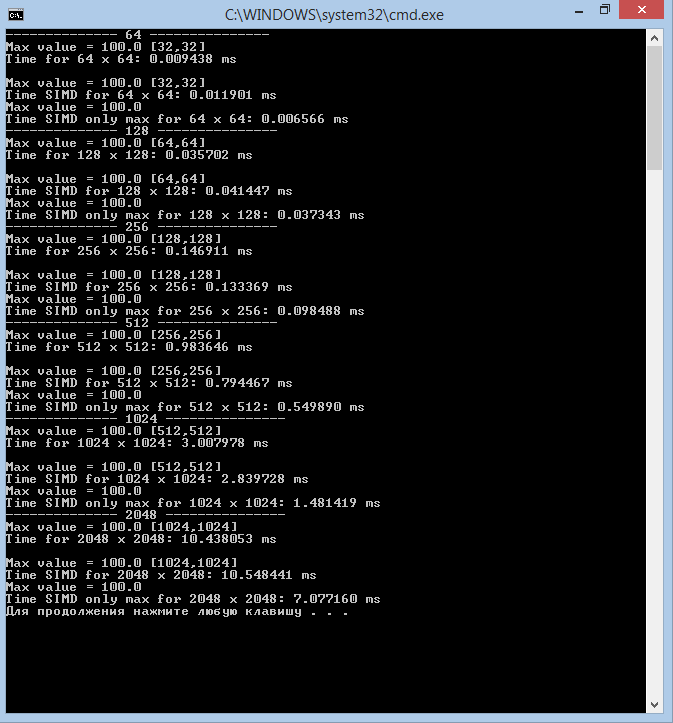
Кравченко Александр

1. Нахождение максимального элемента в матрице (индекс строки, индекс столбца)

Debug 32-bit Debug 64-bit

Release 32-bit Release 64-bit

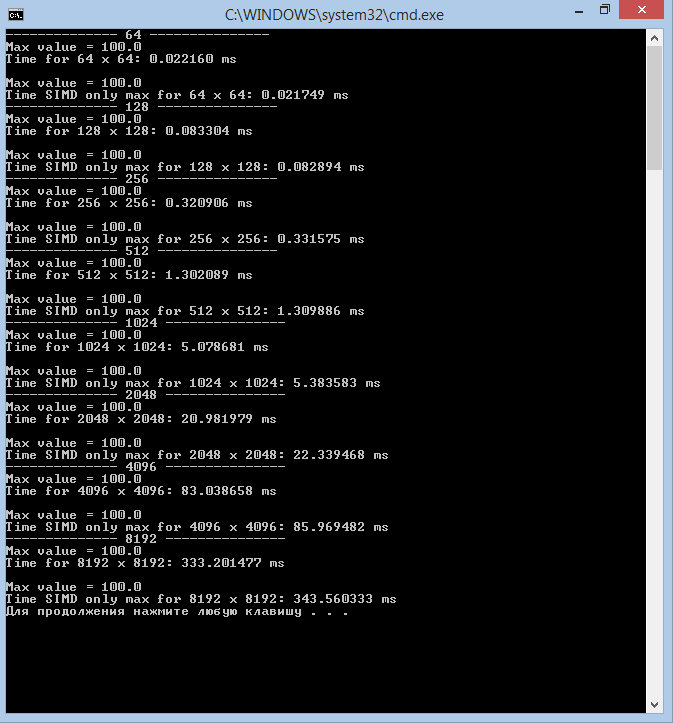
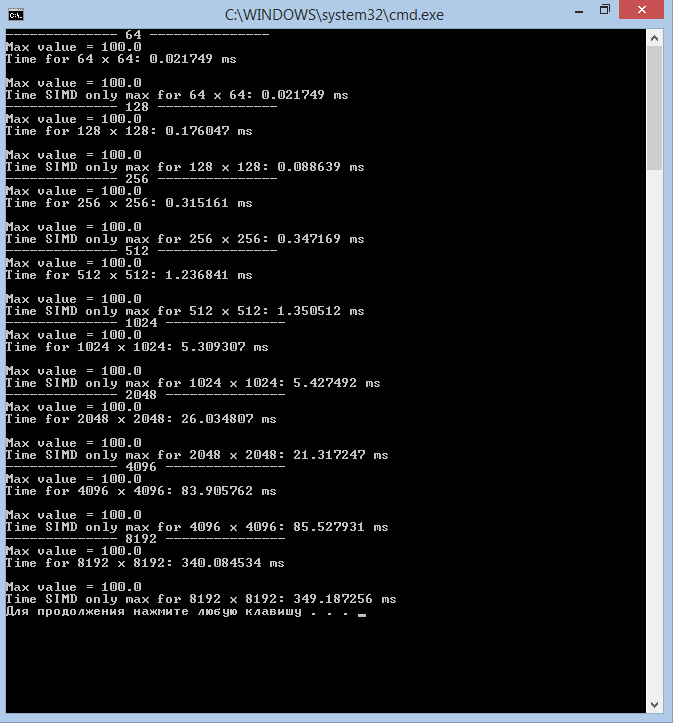
 

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **t (мс)** | **Debug X32** | | **Debug X64** | | **Release X32** | | **Release X64** | |
| **N** | **Stand** | **SIMD** | **Stand** | **SIMD** | **Stand** | **SIMD** | **Stand** | **SIMD** |
| **64** | **0,021** | 0,026 | **0,022** | 0,031 | **0,009** | 0,022 | **0,009** | 0,011 |
| **128** | **0,079** | 0,102 | **0,0828** | 0,120 | **0,039** | 0,06 | **0,035** | 0,041 |
| **256** | **0,305** | 0,403 | **0,358** | 0,669 | **0,15** | 0,17 | 0,146 | **0,133** |
| **512** | **1,234** | 1,983 | **1,99** | 2,34 | **0,59** | 1,16 | 0,98 | **0,7946** |
| **1024** | **5,128** | 6,870 | **7,62** | 8,12 | **2,89** | 3,93 | 3,00 | **2,83** |
| **2048** | **24,04** | 32,99 | **26,12** | 35,97 | **10,04** | 18,2 | **10,43** | 10,548 |

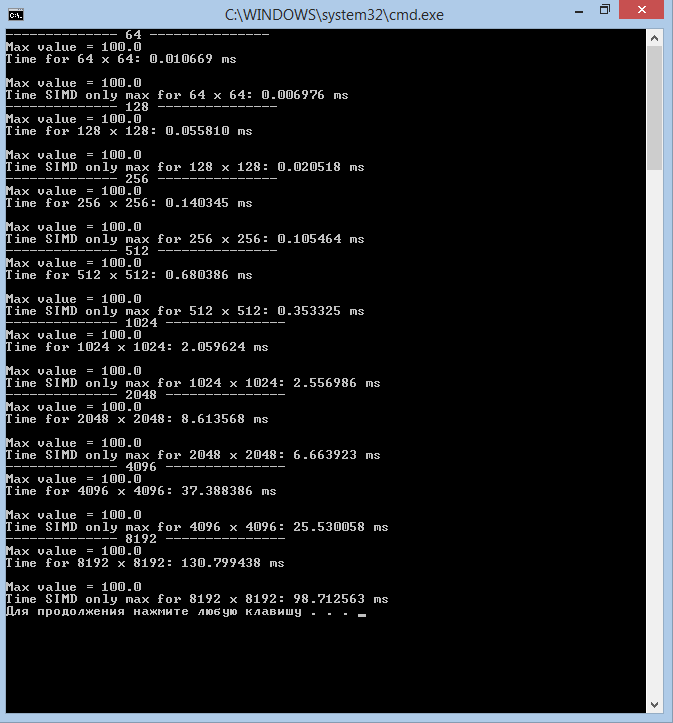
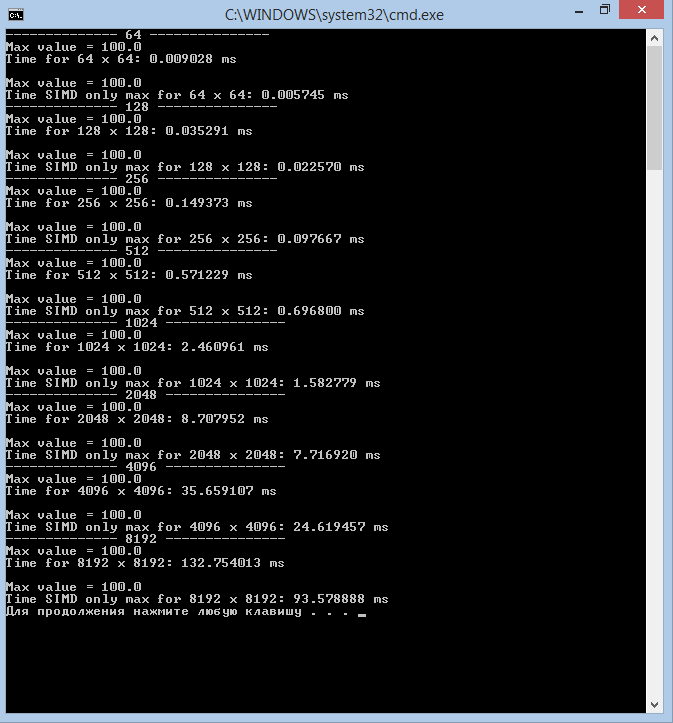
SIMD команды, примененные в моем алгоритме поиска максимального числа и его местоположения работали быстрее только в Release x64 размерностью массива в 256, 512, 1024. Скорее всего тенденция продолжится и после. Проблема в скорости из-за неправильного применения SIMD команд в алгоритме.

Теперь сравним время, где поиск только максимальных элементов.

Debug 32-bit Debug 64-bit

Release 32-bit Release 64-bit

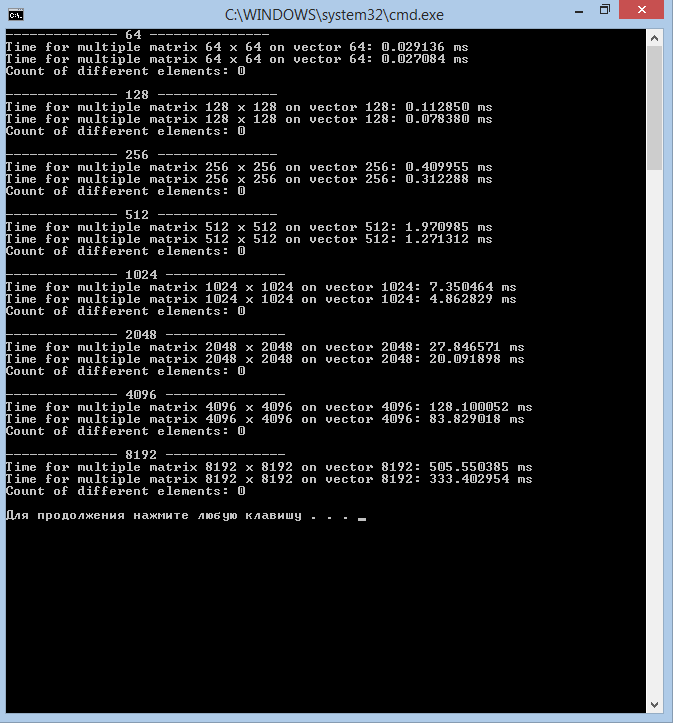
 

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **t (мс)** | **Debug X32** | | **Debug X64** | | **Release X32** | | **Release X64** | |
| **N** | **Stand** | **SIMD** | **Stand** | **SIMD** | **Stand** | **SIMD** | **Stand** | **SIMD** |
| **64** | 0,022 | **0,021** | 0,021 | 0,021 | 0,010 | **0,0069** | 0,009 | **0,0057** |
| **128** | 0,0833 | **0,0828** | 0,17 | **0,08** | 0,55 | **0,020** | 0,035 | **0,022** |
| **256** | **0,32** | 0,33 | **0,31** | 0,34 | 0,140 | **0,105** | 0,149 | **0,097** |
| **512** | **1,302** | 1,309 | **1,23** | 1,35 | 0,68 | **0,35** | **0,57** | 0,696 |
| **1024** | **5,08** | 5,39 | **5,3** | 5,4 | **2,05** | 2,55 | 2,46 | **1,58** |
| **2048** | **20,98** | 22,33 | 26,03 | **21,31** | 8,61 | **6,66** | 8,70 | **7,716** |
| **4096** | **83,03** | 85,97 | **83,9** | 85,52 | 37,38 | **25,53** | 35,65 | **24,619** |
| **8192** | **333,20** | 343,56 | **340,0** | 349,18 | 130,79 | **98,7** | 132,75 | **93,578** |

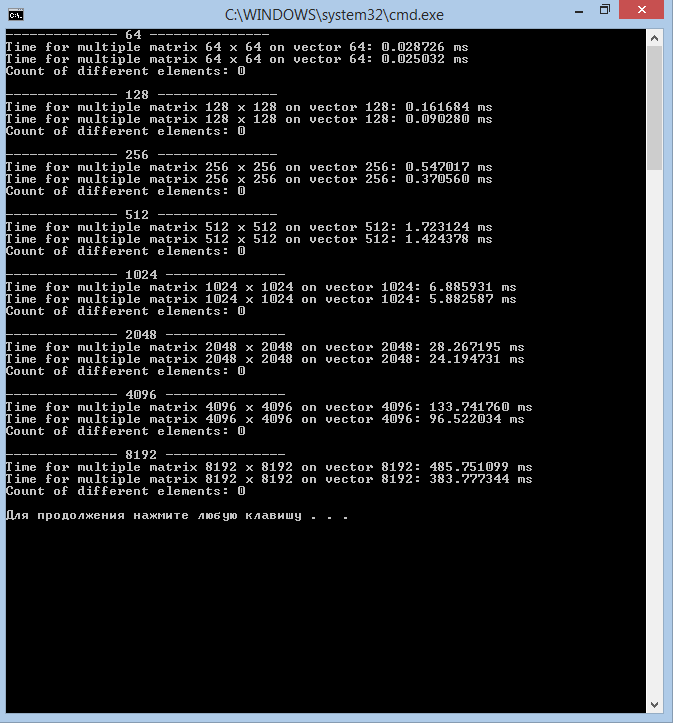
Алгоритм с SIMD командами для поиска максимального числа работают в режиме Debug одинаково с обычным поиском максимального, а в режиме Release заметно быстрее ( до 30% скорости).

1. Умножение матрицы на вектор

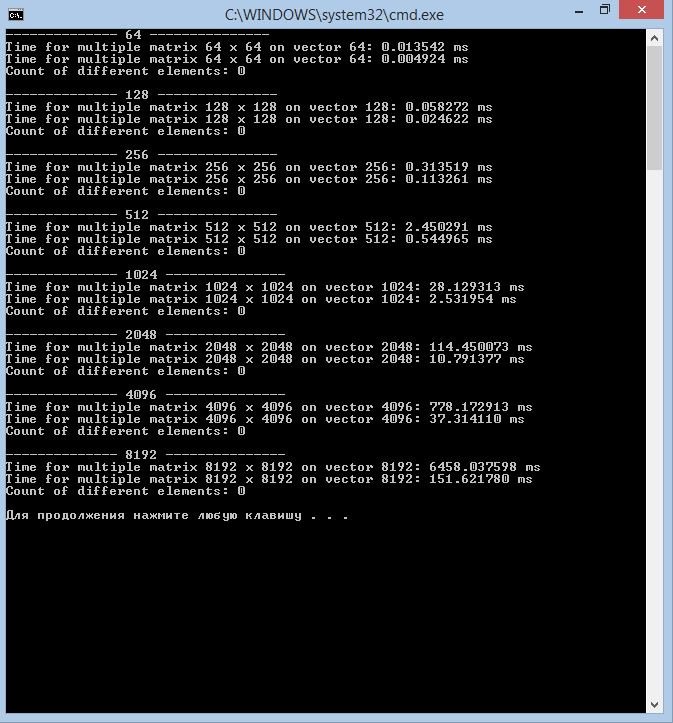
Debug 32-bit



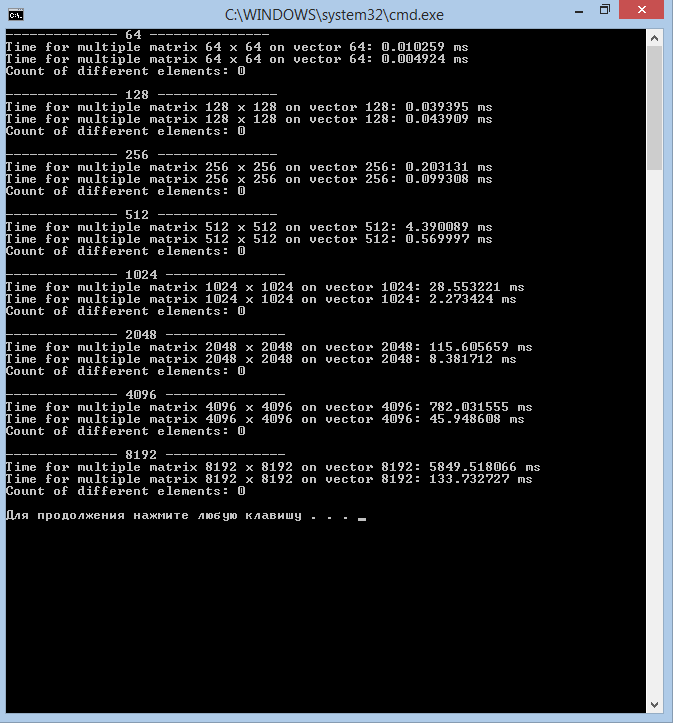
Debug 64-bit



Release 32-bit



Release 64-bit

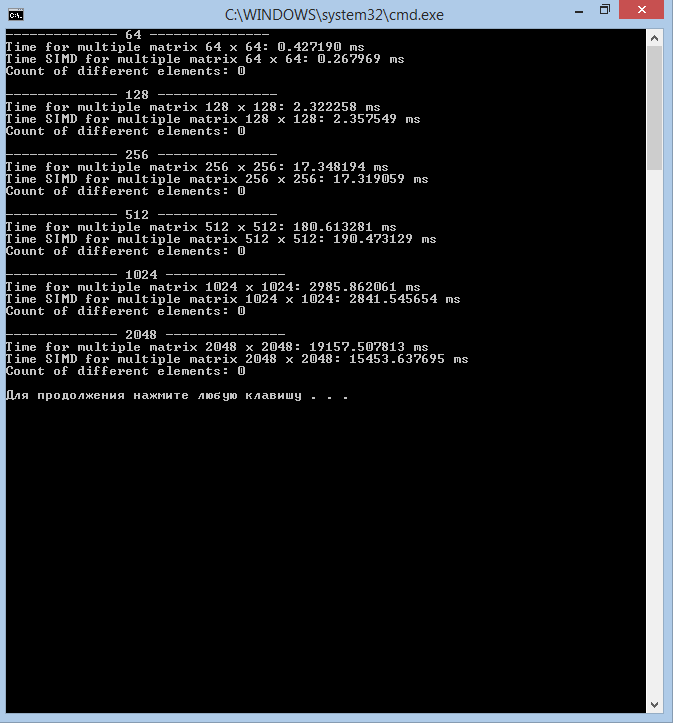


|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **t (мс)** | **Debug X32** | | **Debug X64** | | **Release X32** | | **Release X64** | |
| **N** | **Stand** | **SIMD** | **Stand** | **SIMD** | **Stand** | **SIMD** | **Stand** | **SIMD** |
| **64** | 0,029 | **0,027** | 0,028 | **0,025** | 0,013 | **0,0049** | 0,010 | **0,0049** |
| **128** | 0,112 | **0,078** | 0,16 | **0,09** | 0,058 | **0,024** | **0,039** | 0,043 |
| **256** | 0,409 | **0,31** | 0,54 | **0,37** | 0,31 | **0,113** | 0,20 | **0,099** |
| **512** | 1,97 | **1,27** | 1,72 | **1,42** | 2,45 | **0,54** | 4,39 | **0,56** |
| **1024** | 7,35 | **4,86** | 6,88 | **5,88** | 28,12 | **2,53** | 28,55 | **2,27** |
| **2048** | 27,84 | **20,09** | 28,26 | **24,19** | 114,45 | **10,79** | 115,6 | **8,38** |
| **4096** | 128,1 | **83,82** | 133,74 | **96,52** | 778,17 | **37,3** | 782,03 | **45,94** |
| **8192** | 505,55 | **333,40** | 485,75 | **383,77** | 6458 | **151,62** | 5849,5 | **133,7** |

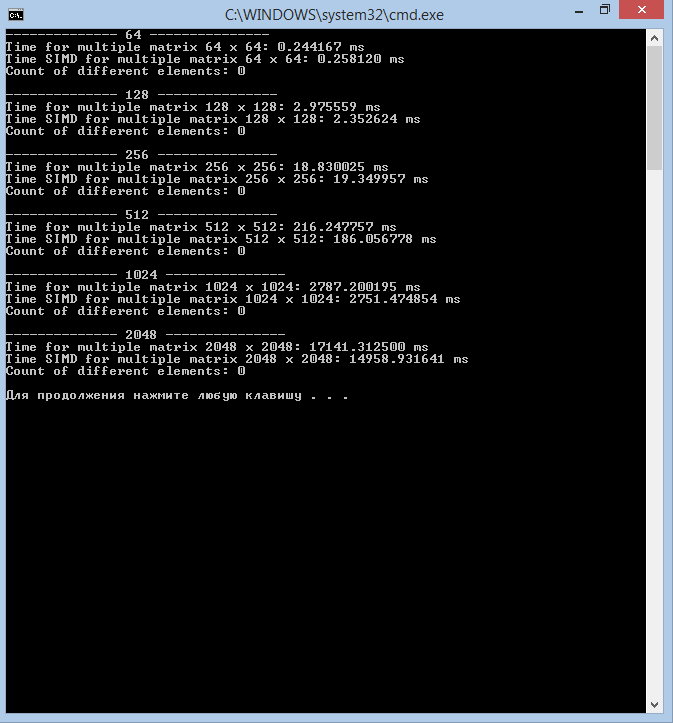
Видно, что использование SIMD команд для умножения матрицы на вектор очень прибавляет к скорости, особенно в режиме Release (до 50 раз).

1. Умножение матрицы на матрицу

Release 32-bit



Release 64-bit



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **t (мс)** | **Release X32** | | **Release X64** | |
| **N** | **Stand** | **SIMD** | **Stand** | **SIMD** |
| **64** | 0,427 | **0,267** | **0,24** | 0,25 |
| **128** | **2,32** | 2,35 | **2,97** | 2,35 |
| **256** | 17,34 | **17,31** | **18,83** | 19,34 |
| **512** | 180,61 | 190 | 216,24 | **186,05** |
| **1024** | 2985 | **2841** | 2787 | **2751** |
| **2048** | 19157 | **15453** | 17141 | **14958** |

Видно, что с увеличением размерности матриц алгоритм с SIMD командами работает быстрее (до 1.5 раз).

1. Вывод: по ходу выполнения индивидуального домашнего задания я убедился, что переход на использование SIMD команд при обработке массивов дает очень хорошую прибавку к экономии времени.