Анализ производительности векторизованных алгоритмов

А.Е. Маслов[I], А.А. Зорин[I]

[I] АО Научно-технический центр «Радуга», г. Москва, Россия

Аннотация. Работа посвящена оценке эффективности применения векторизации для алгоритмов, встречающихся в различных задачах с целью повышения производительности. Определены рациональные случаи для применения SIMD расширения. Определены возможности достижения заявленного теоретического предела повышения производительности. Произведено сравнение применений SSE и AVX расширений для различных типов данных (double, float, комплексные float и double).

Ключевые слова: векторизация; SIMD; SSE; AVX; скалярное произведение; свертка; корреляция;

Введение

Большинство современных процессоров в серверных и персональных компьютерах имеют поддержку SIMD-расширений (Single instruction multiple data), которые могут быть применены для ускорения различных трудоемких операций. Основная идея данного расширения в одновременной обработке нескольких элементов за одну инструкцию. Наиболее распространенными SIMD-расширениями являются SSE, AVX и AVX-512, которые предлагают регистры размерами 128, 256 и 512 бит соответственно, т.е. каждое из расширений предлагает производить различное количество параллельных вычислений за одну операцию. Экспериментально подтверждено, что векторизованные (с применением SIMD-расширений) алгоритмы показывают прирост в скорости вычислений[1-4]. Однако, для каждого конкретного случая, улучшение скорости вычислений варьируются из-за различных факторов, таких как сложность векторизованного представления задачи, невозможность применения полностью векторизованного решения в связи с некратным числом данных размещающихся в SIMD регистрах, наличие промахов кэша.

Следует отметить, что применение технологии SIMD обеспечивает повышение энергоэффективности процессоров Intel, что подтверждается в работах [2] и [5].

Теоретическое увеличение скорости от применения векторизации зависит от типов данных, для которых применяется оптимизация, для чисел с плавающей точкой (float) возможно улучшение в 4 раза при использовании SSE, в 8 раз при использовании AVX и в 16 раз при использовании AVX-512. Однако такой прирост возможен лишь при отсутствии обращений к оперативной памяти (когда все необходимые данные находятся в кэше процессора). Для чисел с плавающей точкой двойной точности возможно ускорение в 2, 4 и 8 раз соответственно.

В настоящей работе рассматривается применение векторизации для нескольких, относительно простых алгоритмов, с целью определения возможности достижения теоретического увеличения производительности, а также с целью определить, насколько целесообразно применять векторизацию в различных случаях и какой прирост производительности может быть достигнут.

Постановка задачи

В настоящей работе рассматриваются функции, которые часто применяются в различных областях, в частности, в цифровой обработке сигналов.

1. Скалярное произведение. Пусть $\vec{a} = (a_1, \dots, a_n),\ \vec{b} = (b_1, \dots, b_n)$.
$$f(a,b) = \sum_{i=1}^{n} a_i b_i \qquad (1);$$
2. Свертка. Пусть $\vec{a} = (a_1, \dots, a_N),\ \vec{b} = (b_1, \dots, b_n), N > n$.
$$S(i,a,b) = \sum_{k=1}^{n} a_{i+k} b_k\ ,\ 0 \le i \le N - n - 1 \ (2);$$
3. Корреляционная свертка. Пусть $\vec{a} = (a_1, \dots, a_N),\ \vec{b} = (b_1, \dots, b_n), N > n$.
$$\rho(i,a,b) = \frac{\sum_{k=1}^{n} a_{i+k} b_k}{\sqrt{\sum_{k=1}^{n} a_{i+k} a_{i+k}} \cdot \sqrt{\sum_{k=1}^{n} b_k b_k}} \qquad (3);$$

Все указанные функции реализованы для следующих типов данных: float, double, комплексные float и double, имеют тесты, которые проверяют корректность работы, сравнивая результаты вычислений с эталонным результатом, а также проводят замер времени выполнения.

Все результаты работы получены на машине с ОС Windows 10, процессором Intel Core i7 - 4960X с поддержкой SSE и AVX инструкций, с 32 Гб оперативной памяти. Язык реализации: C++, с применением встроенных в компилятор функций (интринсики) для векторизации. Использование интринсиков позволяет компилятору применить некоторые дополнительные оптимизации. Компилятор MSVC, с ключом оптимизации (Favor Speed) (/Ox).

Скалярное произведение

Функция расчета скалярного произведения присутствует в стандартной библиотеке языка C++ (std::inner_product), в дальнейшем она используется для определения эталонного значения, для определения корректности результата.

В настоящей работе реализованы следующие варианты: стандартный алгоритм расчета скалярного произведения в соответствии с (1), SSE реализация скалярного произведения, AVX реализация скалярного произведения. Как было указано ранее, все функции работают со следующими типами данных: float, double, комплексные float и double. Для тестирования и замера времени проводилась серия тестов, в каждой из которых были взяты 2 набора

произвольных данных разных размеров. Вычисления повторялись многократно (3 млн. раз) для получения усредненного итогового времени.

Результаты для чисел типа float приведены на рисунке 1. Важно отметить, что данные в данном эксперименте полностью помещаются в кэше, таким образом обращение к оперативной памяти минимизированы, а исходя из представленного рисунка (рис.1 (а)), на котором изображено отношение времени выполнения векторизованного алгоритма к стандартной реализации, можно обнаружить, что прирост стремится к теоретическому максимуму.

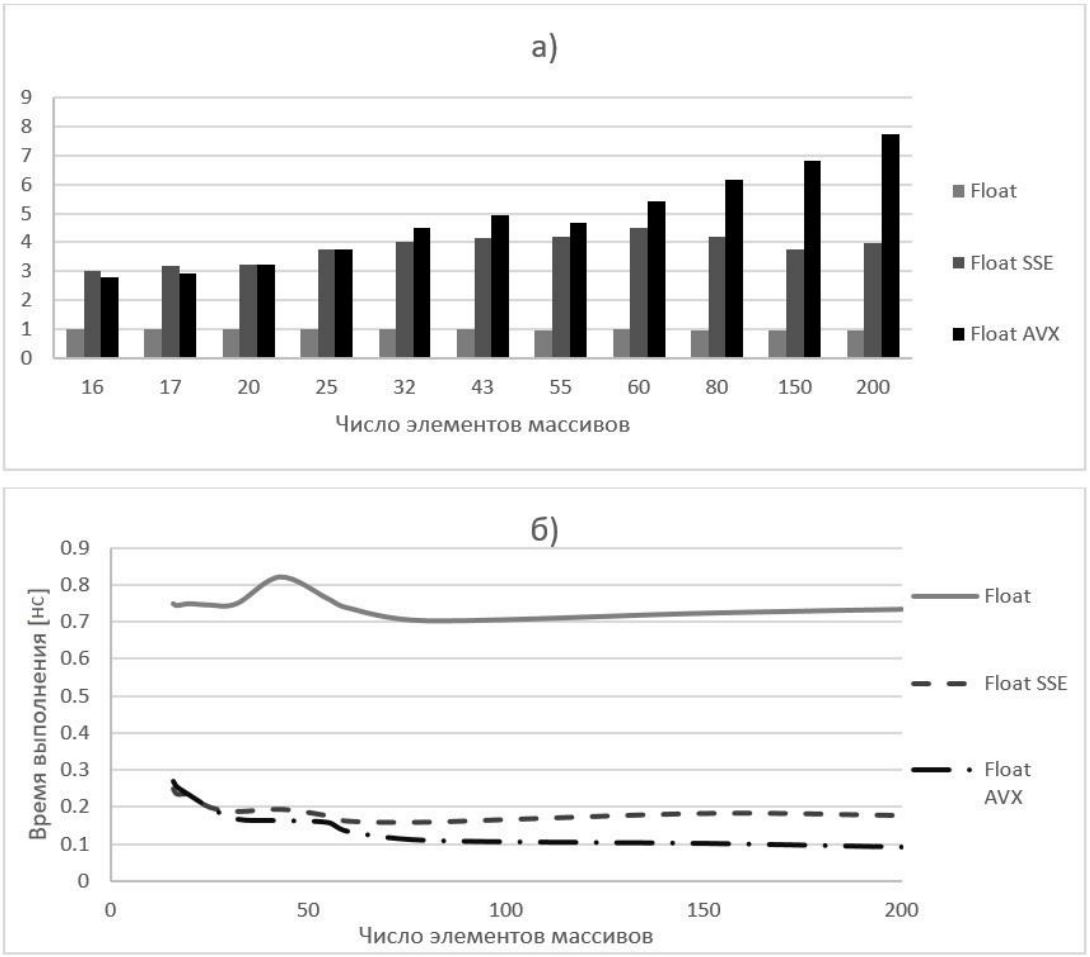

Рис. 1 – Гистограмма (а) и скорость выполнения одной итерации (б) скалярного произведения для различного числа элементов в массивах для чисел типа float

Также, достаточно наглядным представляется график зависимости времени, затраченного на одну итерацию цикла, от числа элементов в массиве (рис. 1 (б)). Следует отметить, что при небольшом количестве данных (массивы с числом элементов меньше 30) применение AVX расширения является нецелесообразным, если данных относительно много, то указанное расширение позволяет получить существенный прирост в скорости выполнения.

Независимо от числа данных, применения векторизации дает ощутимое ускорение (увеличение производительности в 3 раза и выше) и способно уменьшить временные затраты на различные расчеты.

Результаты для чисел с плавающей запятой двойной точности (тип double) представлены на рисунке 2, аналогично рисунку 1. Как и ожидалось, прирост производительности от использования SSE инструкций стремится к двум, в то время как производительность от AVX стремится к 3.5, а не к четырем. Это связано с тем, что некоторые 256 разрядные инструкции имеют задержку или выполняются как последовательность нескольких операций (в соответствии с официальной документацией на сайте производителя [6]).

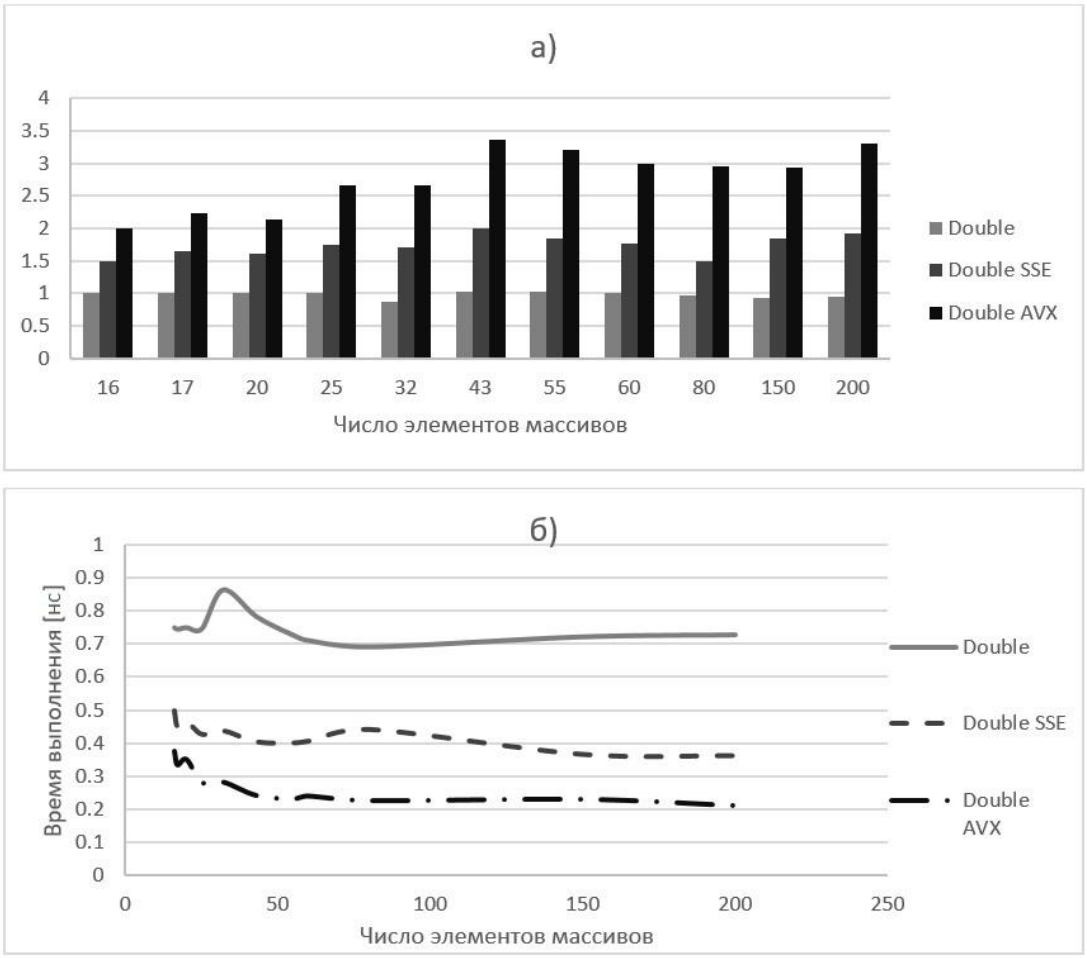

Рис. 2 - Гистограмма (а) и скорость выполнения одной итерации (б) скалярного произведения для различного числа элементов в массивах для чисел типа double

Результаты скалярного произведения для комплексных чисел представлены на рисунках 3 и 4. Аналогично вышеизложенным результатам, с ростом числа элементов в массивах, AVX расширение показывает лучшие результаты в сравнении с SSE.

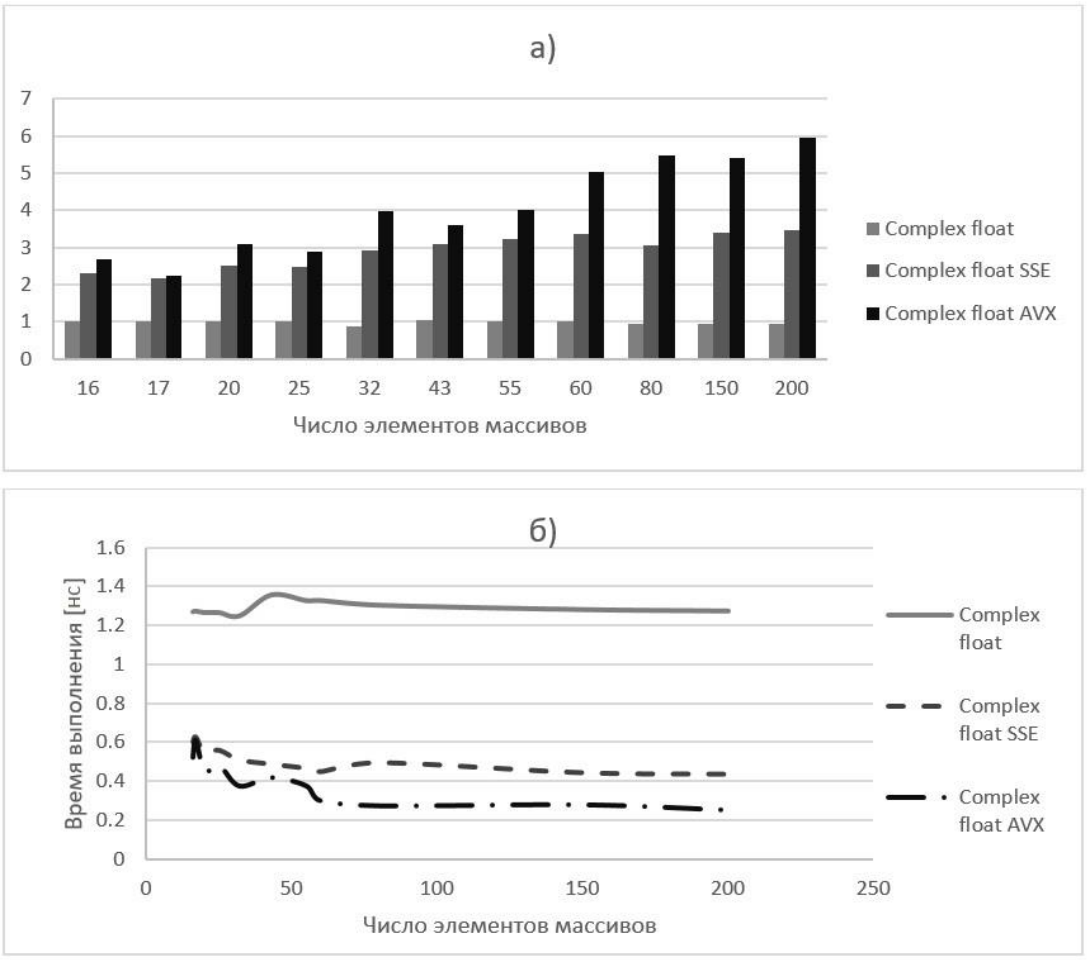

Рис. 3 - Гистограмма (а) и скорость выполнения одной итерации (б) скалярного произведения для различного числа элементов в массивах для чисел типа complex float

Поскольку в 128 разрядных регистрах можно разместить всего 4 значения типа float (т.е. 2 комплексных числа), то ожидалось, что при использовании SSE расширения скорость выполнения вырастет в 2 раза. Оптимизация вычисления производилась в несколько этапов:

– подготавливается промежуточный результат для сохранения действительной и мнимой частей перемножаемых чисел: $X_{re} = \{0, 0, 0, 0\}$, $X_{im} = \{0, 0, 0, 0\}$;

– на каждой итерации цикла в регистры записываются перемножаемые числа и одно из перемножаемых комплексных чисел с перестановленными мнимой и действительной частями: $A = \{a_{ire}, a_{iim}, a_{(i+1)re}, a_{(i+1)im}\}$, $B = \{b_{ire}, b_{iim}, b_{(i+1)re}, b_{(i+1)im}\}$, $B' = \{b_{iim}, b_{ire}, b_{(i+1)im}, b_{(i+1)re}\}$;

– на каждой итерации проводится суммирование произведений мнимых и действительных частей с накоплением результата:

$X_{re} = \{X_{re1} + A_1 * B_1, \ldots, X_{re4} + A_4 * B_4\}$, $X_{im} = \{X_{im1} + A_1 * B'_1, \ldots, X_{im4} + A_4 * B'_4\}$;

– после выхода из цикла производится последнее сложение промежуточных пар, для получения комплексного числа вида $a + ib = X_{re1} - X_{re2} + X_{re3} - X_{re4} + i*(X_{im1} + X_{im2} + X_{im3} + X_{im4})$.

В результате удалось получить выигрыш в скорости выполнения в 3 – 3.5 раза, аналогично для AVX расширения.

Для типа double ускорение от SSE инструкций составляет 40 %, в то время как использование AVX позволяет повысить производительность более чем в 2 раза.

Очевидно, что представленные результаты являются идеализированными, ведь на практике мало кому может понадобиться перемножать одну и ту же выборку большое количество раз.

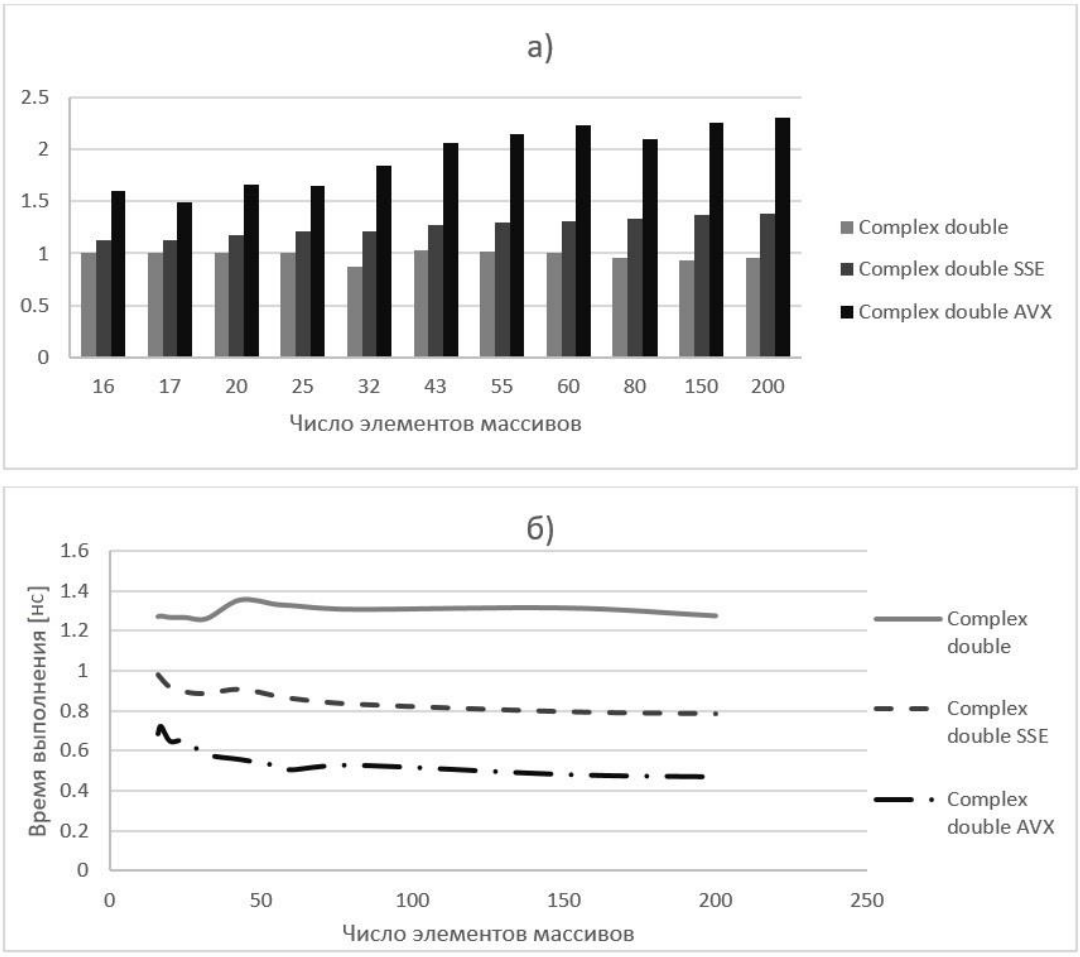

Рис. 4 - Гистограмма (а) и скорость выполнения одной итерации (б) скалярного произведения для различного числа элементов в массивах для чисел типа complex double

Свертка

Следующие результаты представлены для более реалистичного сценария работы, операции свертки, которая реализована в соответствии с (2). В данной работе применяется векторизация внутреннего цикла свертки [7].

Результаты для чисел типа float приведены на рисунке 5, где *n* – размер меньшего из массивов, т.е. тот с которым проводится свертка. При таких условиях работы прирост к скорости становится заметно ниже. Также следует отметить, что прирост производительности от использования AVX расширения в сравнении с SSE невелик, а для малых *n* и вовсе отсутствует. Это обуславливается затратами на загрузку данных в SIMD регистры при постоянном обращении к оперативной памяти, происходят частые промахи кэша, поскольку данные в нем не умещаются. Данный вывод подтверждается тривиальностью операций, т.е. при скалярном произведении производится небольшое число вычислительных операций, при частой загрузке данных.

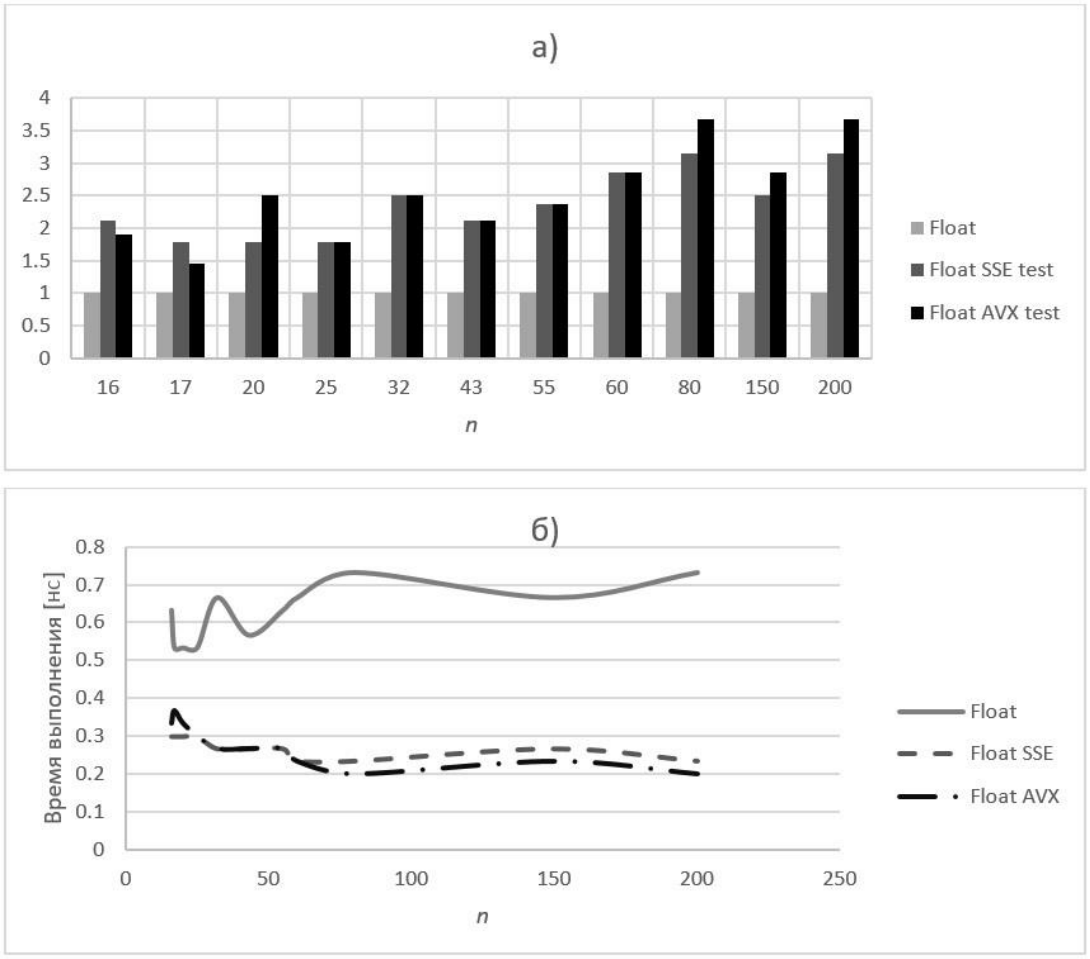

Рис. 5 - Гистограмма (а) и скорость выполнения одной итерации (б) свертки для различного *n* для чисел типа float

Для чисел типа double получены аналогичные результаты, с оговоркой на то, что AVX реализация не дает ощутимого преимущества при указанных *n*, улучшение в сравнении с SSE на 7,6 процента. Результаты тестов приведены на рисунке 6, аналогично ранее приведенным данным.

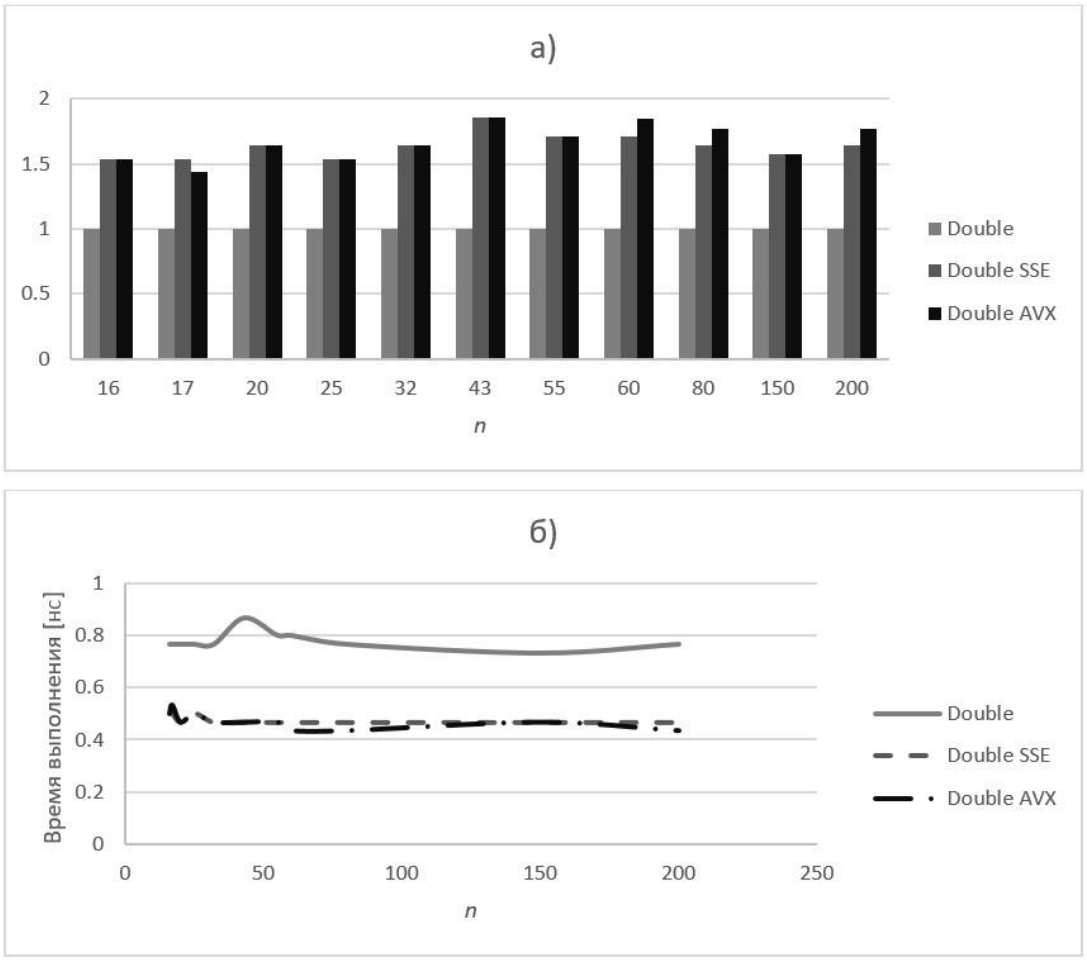

Рис. 6 - Гистограмма (а) и скорость выполнения одной итерации (б) свертки для различного *n* для чисел типа double

Свертка комплексных чисел характерна большим количеством вычислительных операций. Таким образом можно сделать предположение, что выигрыш от использования векторизации для типов complex float и complex double должен быть выше.

Несмотря на то, что 256 битная реализация превосходит 128 битную, для комплексных чисел типа float, разница не столь велика (рис. 7). Таким образом, если имеется алгоритм с применением SSE расширения, переход на 256 битную реализацию предоставляет улучшение на 6,2 процента.

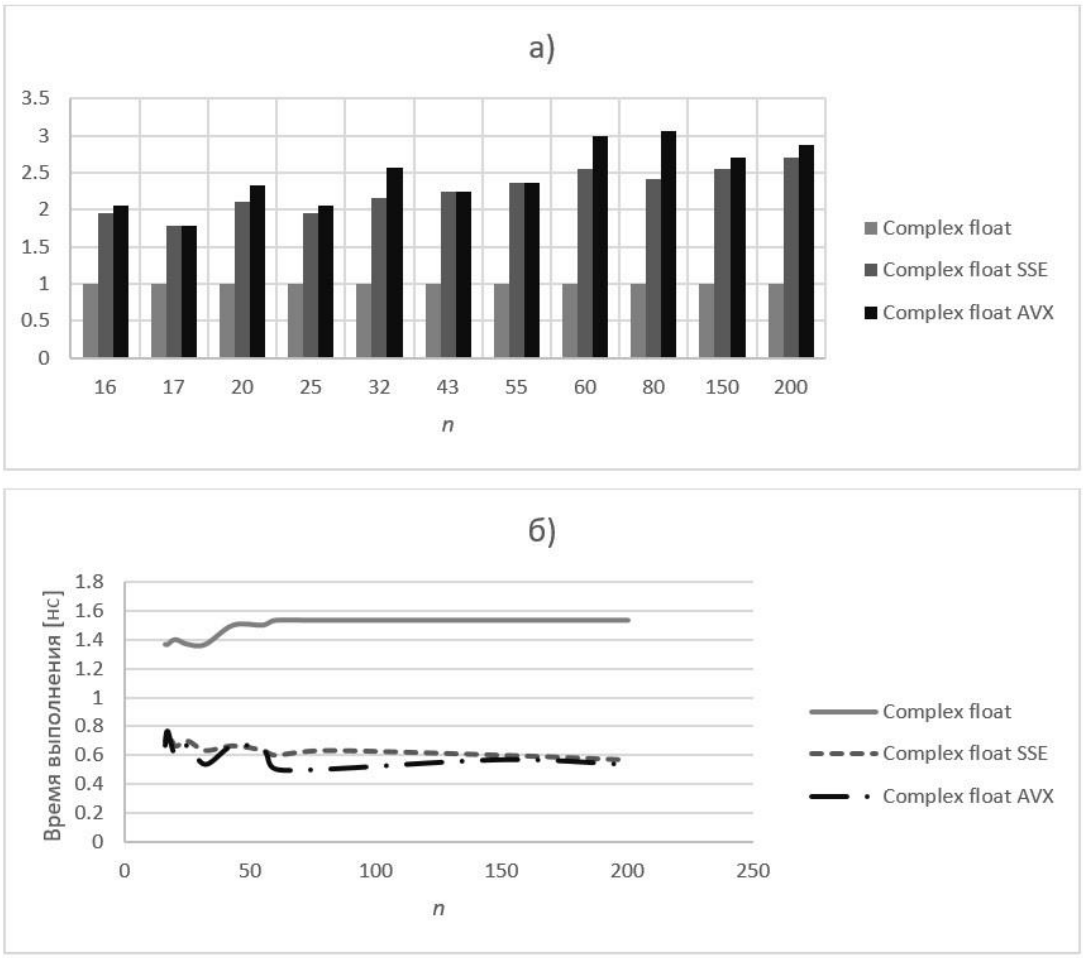

Рис. 7 - Гистограмма (а) и скорость выполнения одной итерации (б) свертки для различного *n* для чисел типа complex float

Для комплексных чисел типа double прирост от использования SSE и AVX расширений составляет 30 и 40 процентов соответственно. Использование 256 разрядных регистров здесь является наиболее предпочтительным.

В целях эксперимента была произведена попытка привести значения типа double к значениям типа float с последующей сверткой. В результате, время работы алгоритма не улучшилось, поскольку затраты на приведение (несмотря на использование технологии SIMD для данной операции) достаточно велики.

Для получения более существенного преимущества, возможно, следует избрать другой подход к векторизации и аналогично проверить его производительность.

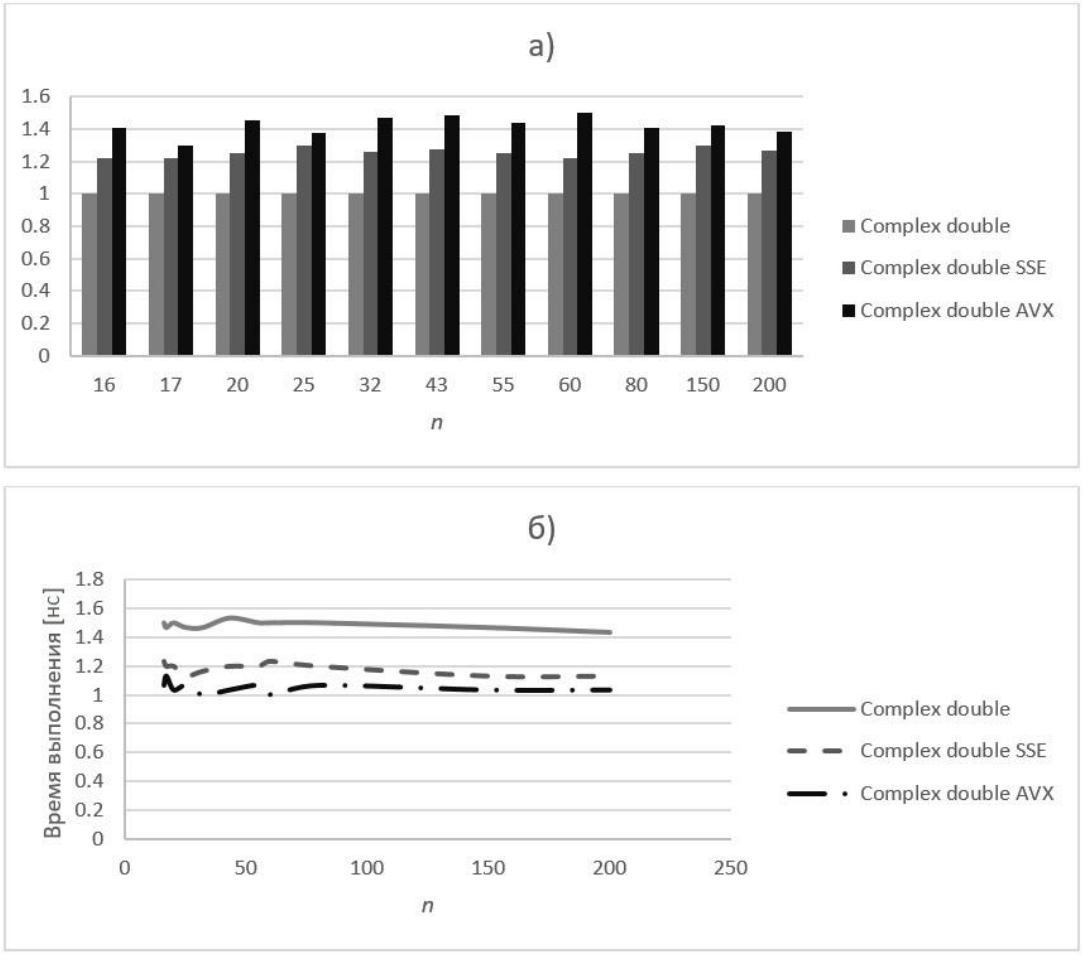

Рис. 8 - Гистограмма (а) и скорость выполнения одной итерации (б) свертки для различного *n* для чисел типа complex double

Корреляционная свертка

Дальнейшим шагом в анализе является применение алгоритма, который реализует большее число вычислительных операций за одну итерацию. Примером данной функции является вычисление коэффициента корреляции, которая реализована в соответствии с (3). Результаты для чисел типа float указаны на рисунке 9, где *n* – величина массива с которым производится корреляция. Очевидно, что при использовании AVX расширения вновь наблюдается относительно невысокий прирост производительности, однако, с ростом *n*, использование 256 разрядного расширения становится более предпочтительным.

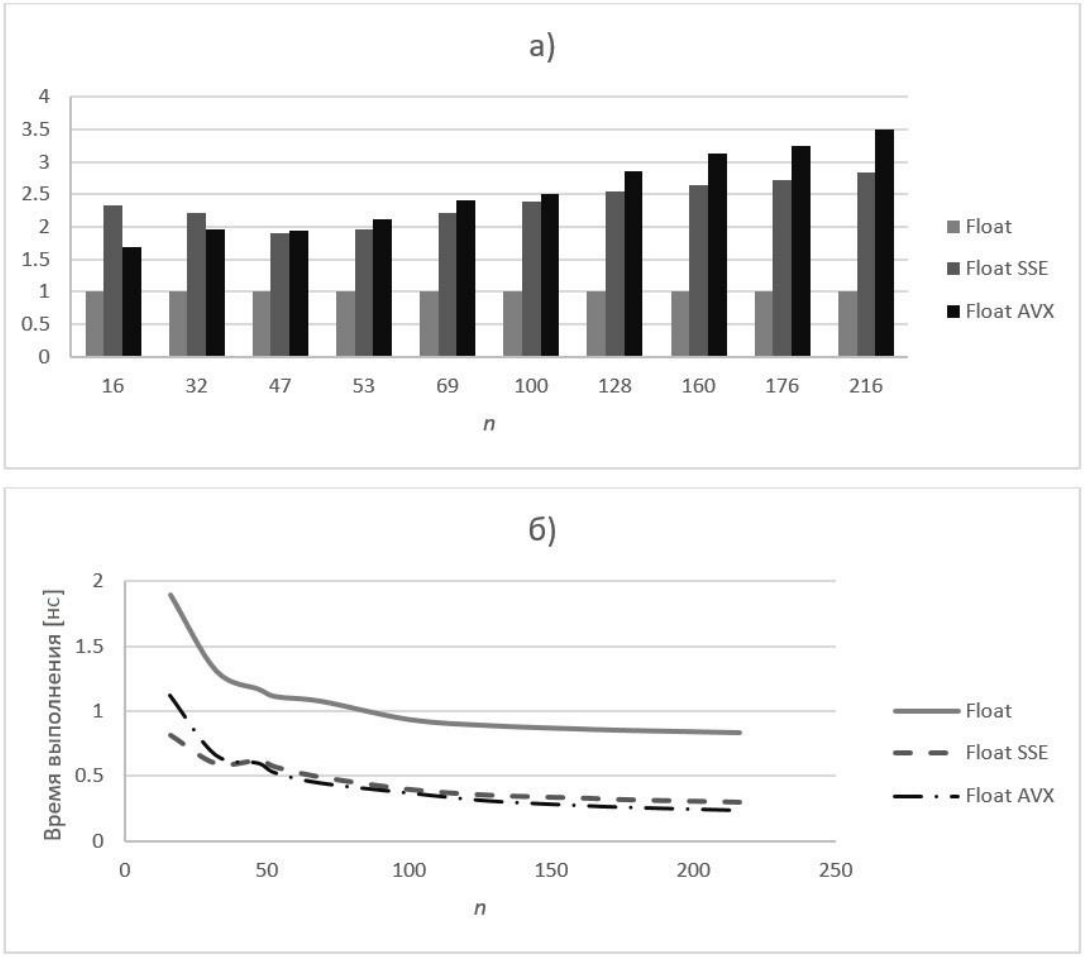

Рис. 9 - Гистограмма (а) и скорость выполнения одной итерации (б) корреляционной свертки для различного *n* для чисел типа float

Результаты корреляции для чисел типа double приведены на рисунке 10. Поскольку в SIMD регистрах размещается меньшее количество значений данного типа, времени на загрузку данных в регистры затрачивается меньше, в результате чего, большую часть времени алгоритм проводит “полезную” работу. Исходя из результатов, можно отметить, что в данном случае, AVX расширение показывает существенный прирост в производительности.

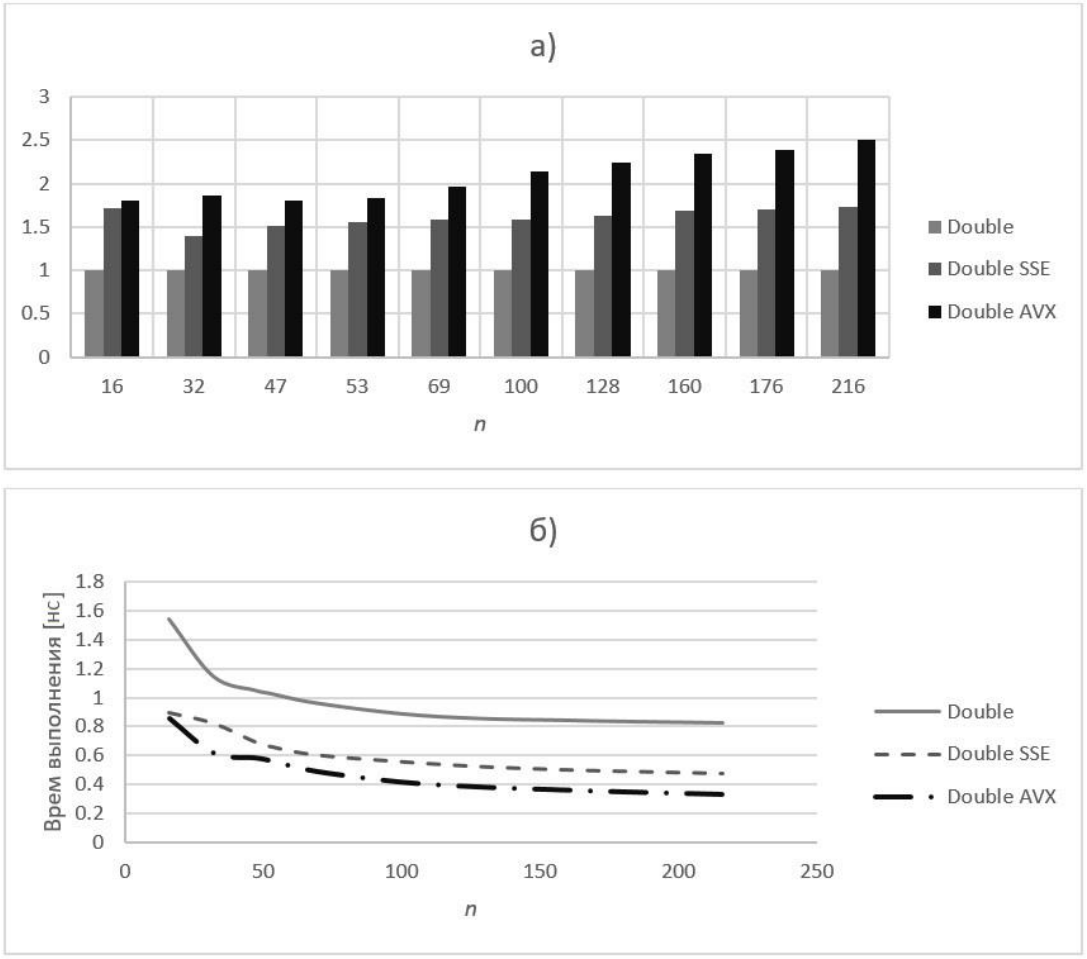

Рис. 10 - Гистограмма (а) и скорость выполнения одной итерации (б) корреляционной свертки для различного *n* для чисел типа double

Определение коэффициента корреляции комплексных чисел проводилось в соответствии с [8]. Результаты представлены на рисунках 11 – 12.

Как и в ранее представленных результатах, для комплексных чисел типа float, прирост производительности от применения технологии AVX незначителен. Однако применение SSE расширения показывает существенное улучшение производительности, в сравнении с реализацией без использования технологии SIMD.

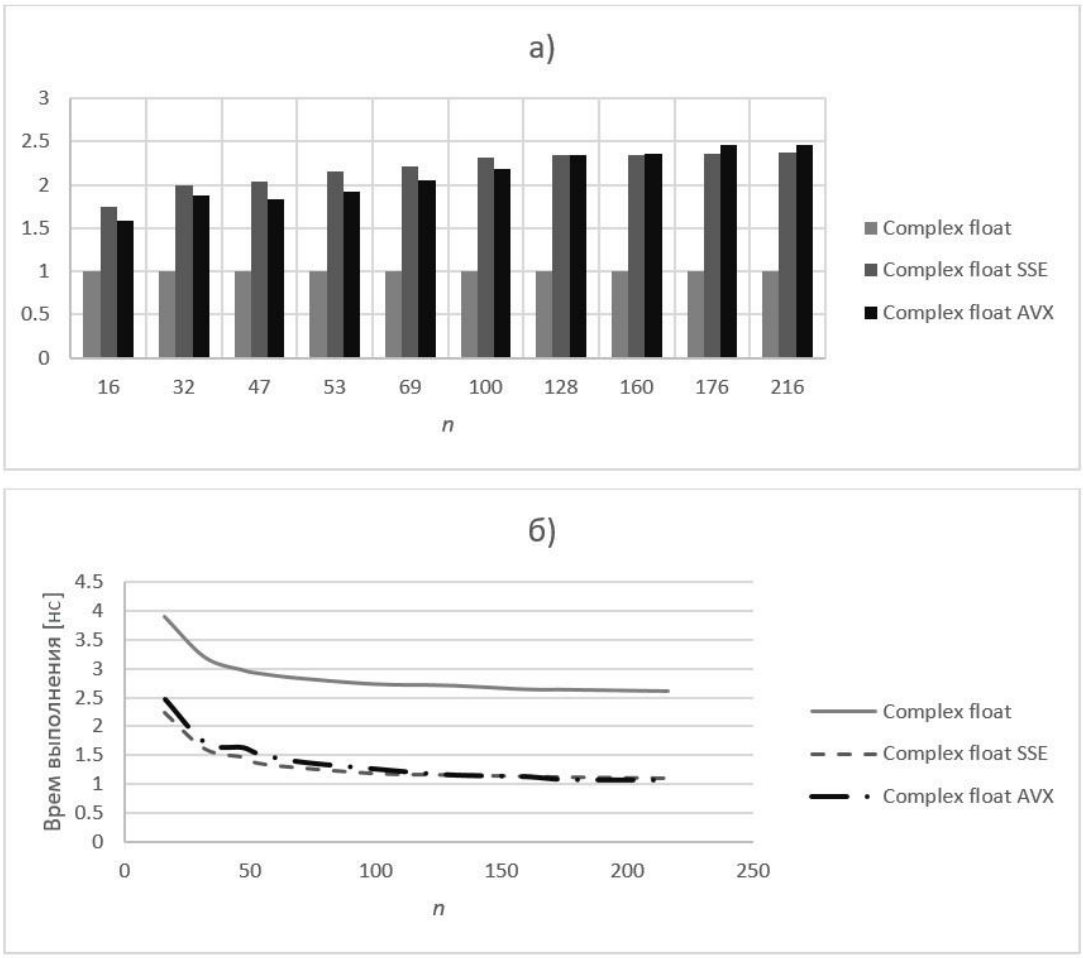

Рис. 11 - Гистограмма (а) и скорость выполнения одной итерации (б) корреляционной свертки для различного *n* для чисел типа complex float

Для комплексных чисел типа double, прирост от использования AVX расширения оказывается на 15% выше, чем при использовании SSE. При этом, как и во всех ранее полученных данных, наблюдается то, что с ростом *n*, векторизация показывает более лучшие результаты. Теоретически, помимо частых обращений к оперативной памяти, это может быть обусловлено не мгновенным переходом процессора на пониженные частоты, при использовании сложных SIMD инструкций, для поддержания допустимого температурного режима процессора.

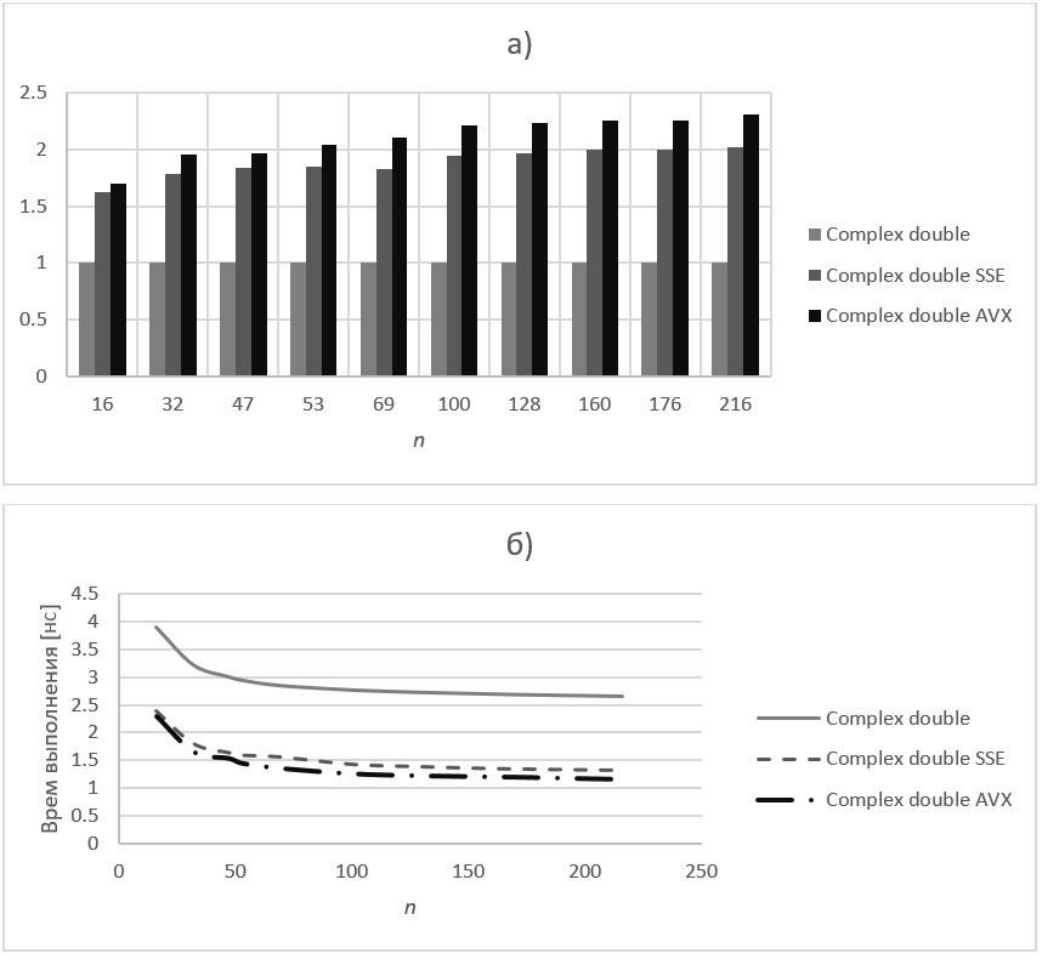

Рис. 12 - Гистограмма (а) и скорость выполнения одной итерации (б) корреляционной свертки для различного *n* для чисел типа complex double

Заключение

В рамках рассматриваемой работы были реализованы различные векторизованные алгоритмы, с проверкой результатов вычислений на корректность и на скорость выполнения. По полученным данным, можно выдвинуть следующие выводы и рекомендации по применению векторизации:

- Применение векторизации позволяет ускорить оптимальные и отлаженные “невекторизованные” алгоритмы;

- Применение векторизации нецелесообразно, при малом количестве вычислительных операций на одну итерацию цикла при частой загрузке данных, из-за промахов кэша, в результате чего, прирост производительности оказывается несущественным;

- Применение векторизации показывает лучшие результаты при больших выборках данных;

- Заявленный теоретический предел в повышении производительности достижим при полном размещении данных в кэше, в данных условиях возможно повысить производительность вычислений до 8 раз (для чисел типа float), при использовании технологии AVX;

- Разница между SSE и AVX расширениями может оказаться несущественной, при использовании данных типа float;

Литература

1. J. M. Cebrián, M. Jahre and L. Natvig, "Optimized hardware for suboptimal software: The case for SIMD-aware benchmarks," 2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2014, pp. 66-75, doi: 10.1109/ISPASS.2014.6844462.
2. Jakobs, T., Naumann, B. & Rünger, G. Performance and energy consumption of the SIMD Gram–Schmidt process for vector orthogonalization. J Supercomput 76, 1999–2021 (2020). https://doi.org/10.1007/s11227-019-02839-0
3. Cui, C., Zhang, X., Jin, Z. (2019). Performance Analysis of Existing SIMD Architectures. In: Xu, W., Xiao, L., Li, J., Zhu, Z. (eds) Computer Engineering and Technology. NCCET 2019. Communications in Computer and Information Science, vol 1146. Springer, Singapore. https://doi.org/10.1007/978-981-15-1850-8_4
4. L. Zhang, X. Yang and W. Yu, "Acceleration study for the FDTD method using SSE and AVX instructions," 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), 2012, pp. 2342-2344, doi: 10.1109/CECNet.2012.6201608.
5. J. M. Cebrián, L. Natvig and J. C. Meyer, "Improving Energy Efficiency through Parallelization and Vectorization on Intel Core i5 and i7 Processors," 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, 2012, pp. 675-684, doi: 10.1109/SC.Companion.2012.93.
6. Intel Intrinsics Guide [Электронный ресурс]. URL: https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html# (дата обращения 05.07.2022).
7. Shahbahrami A, Juurlink B, Vassiliadis S (2005) Efficient vectorization of the FIR filter. In: Proc b16th annual workshop on circuits, systems and signal processing (ProRISC2005), November, pp 432–437
8. Šverko, Z.; Vrankić, M.; Vlahinić, S.; Rogelj, P. Complex Pearson Correlation Coefficient for EEG Connectivity Analysis. Sensors 2022, 22, 1477. https://doi.org/10.3390/s22041477.

Маслов Александр Евгеньевич. АО Научно-технический Центр «Радуга», г. Москва, Россия. Инженер-программист, кандидат технических наук. Количество печатных работ: 12. Область научных интересов: информационные технологии, электротехника.

Зорин Андрей Александрович. АО Научно-технический Центр «Радуга», г. Москва, Россия. Руководитель отдела разработки программного обеспечения. Область научных интересов: информационные технологии, электротехника, цифровая обработки сигналов, оптимизационные задачи. e-mail: a_zorin@ntc-raduga.ru.

Performance analysis of vectorized algorithms

A.E. Maslov[I], A.A. Zorin[I]

[I] Join-stock company Scientific and technical center «Raduga», Moscow, Russia

Abstract. This paper is devoted to evaluating the efficiency of vectorization for algorithms, which are used in various tasks in order to improve performance. Rational use cases for the SIMD extension are determined. The possibilities of achieving the declared theoretical limit of performance increase are determined. Comparison of use of SSE and AVX extensions for various data types (double, float, complex float and double) is made.

Keywords: vectorization; SIMD; SSE; AVX; dot product; convolution; correlation;

References

1. J. M. Cebrián, M. Jahre and L. Natvig, "Optimized hardware for suboptimal software: The case for SIMD-aware benchmarks," 2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2014, pp. 66-75, doi: 10.1109/ISPASS.2014.6844462.

2. Jakobs, T., Naumann, B. & Rünger, G. Performance and energy consumption of the SIMD Gram–Schmidt process for vector orthogonalization. J Supercomput 76, 1999–2021 (2020). https://doi.org/10.1007/s11227-019-02839-0

3. Cui, C., Zhang, X., Jin, Z. (2019). Performance Analysis of Existing SIMD Architectures. In: Xu, W., Xiao, L., Li, J., Zhu, Z. (eds) Computer Engineering and Technology. NCCET 2019. Communications in Computer and Information Science, vol 1146. Springer, Singapore. https://doi.org/10.1007/978-981-15-1850-8_4

4. L. Zhang, X. Yang and W. Yu, "Acceleration study for the FDTD method using SSE and AVX instructions," 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), 2012, pp. 2342-2344, doi: 10.1109/CECNet.2012.6201608.

5. J. M. Cebrián, L. Natvig and J. C. Meyer, "Improving Energy Efficiency through Parallelization and Vectorization on Intel Core i5 and i7 Processors," 2012 SC Companion: High Performance

Computing, Networking Storage and Analysis, 2012, pp. 675-684, doi: 10.1109/SC.Companion.2012.93.

6. Intel Intrinsics Guide. Available at: https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html# (accessed July 5, 2022).

7. Shahbahrami A, Juurlink B, Vassiliadis S (2005) Efficient vectorization of the FIR filter. In: Proc b16th annual workshop on circuits, systems and signal processing (ProRISC2005), November, pp 432–437

8. Šverko, Z.; Vrankić, M.; Vlahinić, S.; Rogelj, P. Complex Pearson Correlation Coefficient for EEG Connectivity Analysis. Sensors 2022, 22, 1477. https://doi.org/10.3390/s22041477.

A.E. Maslov. PhD, Join-stock company, Scientific and technical center «Raduga», 5-th Donskoy proezd 15, Moscow, 119991, Russia.

A.A. Zorin. Join-stock company, Scientific and technical center «Raduga», 5-th Donskoy proezd 15, Moscow, 119991, Russia. e-mail: a_zorin@ntc-raduga.ru