

Министерство образования Республики Беларусь  
Учреждение образования «Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей  
Кафедра информатики  
Дисциплина «Архитектура вычислительных систем»

## **ОТЧЕТ**

к лабораторной работе №4

на тему:

**«ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ РАСШИРЕНИЙ  
SSE/SSE2»**

БГУИР 1-40-04-01

Выполнил студент группы 253504  
Новиков Валерий Андреевич

---

(дата, подпись студента)

Проверила ассистент кафедры  
информатики  
Калиновская Анастасия  
Александровна

---

(дата, подпись преподавателя)

Минск 2024

## ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

### *Расширение SSE*

SSE (англ. Streaming SIMD Extensions, потоковое SIMD-расширение процессора) — это набор SIMD инструкций, разработанный Intel, и впервые представленный в процессорах серии Pentium III.

Технология SSE позволяет преодолеть основную проблему MMX — при использовании MMX невозможно одновременно использовать инструкции сопроцессора, так как его регистры используются и для MMX и для работы FPU.

Расширение позволяет выполнять векторные (пакетные) и скалярные инструкции.

Векторные инструкции реализуют операции сразу над четырьмя комплектами операндов. Скалярные инструкции работают только с одним комплектом операндов — младшим 32-битным словом.

SSE включает в архитектуру процессора восемь 128-битных регистров xmm0...xmm7, каждый из которых трактуется как 4 последовательных значения с плавающей точкой одинарной точности. Расширение позволяет выполнять векторные (пакетные) и скалярные инструкции. *Векторные инструкции* реализуют операции сразу над четырьмя комплектами операндов. *Скалярные инструкции* работают только с одним комплектом операндов — младшим 32-битным словом.

Реализация блоков SIMD осуществляется распараллеливанием вычислительного процесса между данными. То есть когда через один блок проходит поочередно множество потоков данных.

### *Расширение SSE2*

SSE2 (англ. Streaming SIMD Extensions 2, потоковое SIMD-расширение процессора) — это

SIMD (англ. Single Instruction, Multiple Data, Одна инструкция — множество данных) набор инструкций, разработанный Intel, и впервые представленный в процессорах серии Pentium 4.

SSE2 использует те же восемь 128-битных регистров xmm0...xmm7 что и расширение SSE, каждый из которых трактуется как 2 последовательных значения с плавающей точкой двойной точности. SSE2 включает в себя набор инструкций, которые производят операции со скалярными и упакованными типами данных. Также SSE2 содержит инструкции для потоковой обработки целочисленных данных в тех же 128-битных xmm регистрах, что делает это расширение более

предпочтительным для целочисленных вычислений, нежели использование набора инструкций MMX.

### ***Команды SSE2***

При описании операндов инструкций использованы следующие обозначения:

- mmx – любой из восьми 64-х разрядных регистров MMX.
- xmm – любой из восьми 128-ми разрядных регистров.
- r32 – любой 32-х разрядный регистр общего назначения: EAX, EBX и так далее.
- m128, m64, m32, m8 – элемент памяти соответствующего размера в битах.
- imm8 – непосредственный способ адресации, число имеющее размер байта, например, константа сдвига.

Если в качестве операнда указано только имя регистра или только элемент памяти, то это означает, что операнд может находиться только в регистре или только в ОЗУ. Если же указано сочетание обозначений имени регистра и элемента памяти, разделенное наклонной скобкой, например, xmm/m128 то операнд может находиться либо в регистре, либо в ОЗУ

## **ПРАКТИЧЕСКАЯ ЧАСТЬ**

**Цель работы:** Вариант 19. Изучить программную модель SSE, изучить систему команд SSE, обработать массивы из 8 элементов по следующему выражению:  $F[i] = (A[i] + B[i]) * (C[i] + D[i])$ ,  $i = 1 \dots 8$ ;

**Ход работы:** на рисунке 1 представлены регистры XMM до выполнения программы, на рисунке 2 представлены входные данные, на рисунке 3 представлены регистры MMX после выполнения программы, на рисунке 4 представлены результаты программы.

Листинг 1 – Исходный код программы

```
__asm {
    xorps xmm0, xmm0
    xorps xmm1, xmm1
    xorps xmm2, xmm2
    xorps xmm3, xmm3
    xorps xmm4, xmm4
    xorps xmm5, xmm5
```

```

xorps xmm6, xmm6

movups xmm0, A
punpcklbw xmm0, xmm7
movups xmm1, B
punpcklbw xmm1, xmm7
movups xmm2, C
punpcklbw xmm2, xmm7
movups xmm3, D
pmullw xmm1, xmm2
addps xmm0, xmm1
addps xmm0, xmm3
movups F, xmm0
}

```

```

XMM0 = 0000000000000000-0000000000000000
XMM1 = 0000000000000000-0000000000000000
XMM2 = 0000000000000000-0000000000000000
XMM3 = 0000000000000000-0000000000000000
XMM4 = 0000000000000000-0000000000000000
XMM5 = 0000000000000000-0000000000000000
XMM6 = 0000000000000000-0000000000000000
XMM7 = 0000000000000000-0000000000000000

```

Рисунок 1 – Регистры XMM до выполнения программы

```

__int8 A[8] = { 1, 2, 3, 4, 5, 6, 7, 8 };
__int8 B[8] = { 3, 3, 5, 7, 2, 1, 8, 5 };
__int8 C[8] = { 2, 2, 1, 1, 3, 2, 1, 2 };
__int16 D[8] = { 3, 2, 1, 2, 3, 4, 5, 1 };

```

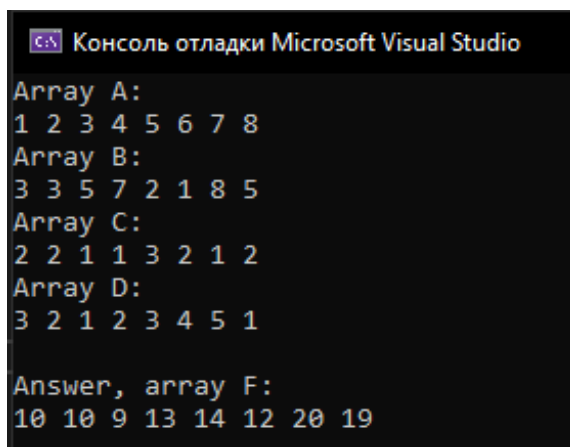
Рисунок 2 – Входные данные

```

XMM0 = 00130014000C000E-000D0009000A000A
XMM1 = 000A000800020006-0007000500060006
XMM2 = 0002000100020003-0001000100020002
XMM3 = 0001000500040003-0002000100020003
XMM4 = 0000000000000000-0000000000000000
XMM5 = 0000000000000000-0000000000000000
XMM6 = 0000000000000000-0000000000000000
XMM7 = 0000000000000000-0000000000000000

```

Рисунок 3 – Регистры XMM после выполнения программы



```
cs: Консоль отладки Microsoft Visual Studio
Array A:
1 2 3 4 5 6 7 8
Array B:
3 3 5 7 2 1 8 5
Array C:
2 2 1 1 3 2 1 2
Array D:
3 2 1 2 3 4 5 1
Answer, array F:
10 10 9 13 14 12 20 19
```

Рисунок 4 – Результат вычислений

**Вывод:** в результате лабораторной работы была изучена программная модель SSE и выполнена поставленная задача.