

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 №3

на тему:

**«ТЕХНОЛОГИЯ MMX»**

БГУИР 1-40-04-01

Выполнил студент группы 253505

Снежко Максим Андреевич

---

(дата, подпись студента)

Проверил

Калиновская Анастасия

Александровна

---

(дата, подпись преподавателя)

Минск 2024

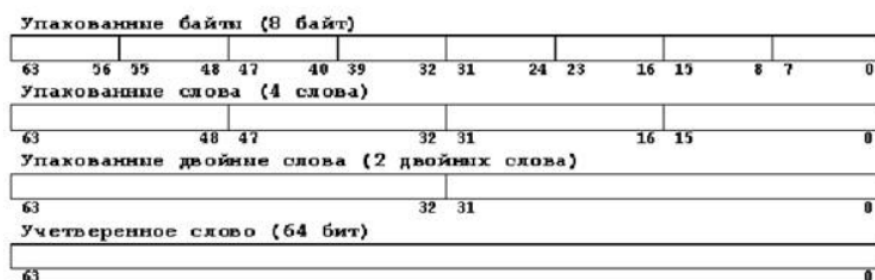
**Цель работы:** изучить технологию MMX, ее назначение, методы применения, а также выполнить задание в соответствии с номером варианта.

**Теоретические сведения:** Набор регистров MMX состоит из восьми 64-разрядных регистров MM0-MM7. Эти регистры могут использоваться только для выполнения вычислений над MMX типами данных и не могут использоваться для адресации памяти. Адресация операндов MMX-команды в памяти осуществляется, используя стандартные способы адресации и регистры общего назначения. Хотя регистры MMX определены в архитектуре IA-32 как отдельные регистры, они являются псевдонимами младших 64-разрядных частей регистров R0..R7 стека математического сопроцессора как изображено на рисунке 1.

#### *Типы данных MMX*

Технология MMX определяет следующие новые 64-разрядные типы данных (рисунок 2):

- упакованные байты – восемь байт, упакованные в одно 64-разрядное поле;
- упакованные слова – четыре слова, упакованные в одно 64-разрядное поле;
- упакованные двойные слова – два двойных слова, упакованные в одно 64-разрядное поле;
- учетверенное слово – одно 64-разрядное поле.



Технология MMX поддерживает новую арифметическую возможность, известную как арифметика с насыщением (Saturated Arithmetic). Арифметику с насыщением лучше всего определить, противопоставляя ее арифметике цикличности (Wraparound Arithmetic). В арифметике цикличности результаты, которые переполняются или антипереполняются, усекаются и возвращаются только самые младшие биты результата (только те которые входят в разрядную сетку соответствующего типа), т.е. перенос игнорируется. В режиме насыщения результаты операции, которые переполняются или антипереполняются, приводятся к соответствующим значениям границ диапазона для данного типа данных (таблица 1). Результат операции, который превышает верхнюю границу диапазона типа данных, насыщается к максимальному значению диапазона, а результат, который оказывается меньше нижней границы, – к минимальному значению диапазона. Этот метод обработки переполнения и антипереполнения применяется во многих приложениях. Например, когда результат превышает диапазон данных для знаковых байтов, он обрезается до 7fh для знаковых

байтов и 0fffh для байтов без знака. Если значение меньше диапазона, оно обрезается до 80h для знаковых байтов и 00h для байтов без знака.

Система команд MMX состоит из 57 команд, сгруппированных в следующие категории:

- команды пересылки данных;
- арифметические команды;
- команды сравнения;
- логические команды;
- команды сдвига;
- команды упаковки и распаковки;
- дополнительные команды;
- команда инициализации.

Команды преобразования преобразовывают элементы данных внутри упакованного типа данных. Команды packsswb и packssdw (упакованный со знаковой насыщенностью) преобразовывают знаковые слова в знаковые байты или знаковые двойные слова в знаковые слова в режиме знаковой насыщенности. Команда packuswb (упакованный насыщенностью без знака) преобразовывает знаковые слова в байты без знака в режиме насыщенности без знака. Команды punpckhbw, punpckhwd и punpckhdq (распаковать старшие упакованные данные) и punpcklbw, punpcklwd и punpckldq (распаковать младшие упакованные данные) преобразовывают байты в слова, слова в двойные слова или двойные слова в четверное слово.

Команды логического сдвига влево, логического сдвига вправо и арифметического сдвига право сдвигают каждый элемент на определенное число битов. Логические левые и правые сдвиги также дают возможность перемещать 64-разрядное поле как один блок, что помогает в преобразованиях типа данных и операциях выравнивания. Команды psllw, pslld (упакованный логический сдвиг влево) и psrlw, psrld (упакованный логический сдвиг вправо) выполняют логический левый или правый сдвиг и заполняют пустые старшие или младшие битовые позиции нулями. Эти команды поддерживают упакованные слова, упакованные двойные слова и четверное слово. Команды psraw и psrad (упакованный арифметический сдвиг вправо) выполняют арифметический сдвиг вправо, копируя знаковый разряд в пустые разрядные позиции на старшем конце операнда. Эти команды поддерживают упакованные слова и упакованные двойные слова.

Команды paddsb и paddsw (упакованное сложение с насыщенностью) и psubsb и psubsw (упакованное вычитание с насыщенностью) выполняют сложение или вычитание знаковых элементов данных операнда источника и операнда адресата и приводят результат к граничным значениям диапазона знакового типа данных. Эти команды поддерживают упакованные байты и упакованные слова. Команды paddusb и paddusw (упакованное сложение без знака с насыщенностью) и

pshufb и pshufw (упакованное вычитание без знака с насыщенностью) выполняют сложение или вычитание элементов данных без знака операнда источника и операнда адресата и приводят результат к граничным значениям диапазона типа данных без знака. Эти команды поддерживают упакованные байты и упакованные слова.

### Порядок выполнения работы:

- а) изучить программную модель MMX;
- б) изучить систему команд MMX;
- в) обработать массивы из 8 элементов по следующему выражению (в зависимости от варианта):

$$1) F[i] = (A[i] + B[i]) * C[i] + D[i], i = 1 \dots 8;$$

Используются следующие массивы:

**A, B и C** – 8 разрядные целые знаковые числа (`_int8`);

**D** – 16 разрядные целые знаковые числа (`_int16`).

**Ход работы:** В ходе выполнения задания был написан программный код на языке C++ с ассемблерными вставками. Исходный код представлен в листинге 1.

### Листинг 1 – Исходный код программы задания 1

```
#include <iostream>
#include "conio.h"

int main()
{
    __int8 A[8] = { 5, 9, 3, 14, 6, 10, 3, 8 };
    __int8 B[8] = { 3, 14, 11, 16, 12, 18, 5, 11 };
    __int8 C[8] = { 15, 11, 16, 21, 22, 25, 19, 24 };
    __int16 D[8] = { 10, 4, 7, 8, 10, 4, 11, 12 };
    __int16 F[8];

    __asm {
        movq mm0, [A]
        movq mm1, [B]
        movq mm2, [C]
        movq mm3, [D]

        punpcklbw mm4, mm0; Распаковать нижние 4 байта в mm4
        punpckhbw mm5, mm0; Распаковать верхние 4 байта в mm5

        pxor mm0, mm0; Очищаем регистры

        punpcklbw mm6, mm1; Распаковать нижние 4 байта в mm6
        punpckhbw mm7, mm1; Распаковать верхние 4 байта в mm7

        pxor mm1, mm1

        punpcklbw mm0, mm2; Распаковать нижние 4 байта в mm0
        punpckhbw mm1, mm2; Распаковать верхние 4 байта в mm1

        pxor mm2, mm2; Очищаем регистры
```

```

movq mm2, [D + 8]; Записываем в mm2 вторую половину D

psrlw mm0, 8; Смещаем на 8 чтобы потом умножение работало корректно
psrlw mm1, 8;
psrlw mm4, 8
psrlw mm6, 8
psrlw mm5, 8
psrlw mm7, 8

paddsw mm4, mm6
pxor mm6, mm6
movq mm6, mm4

pmullw mm4, mm0; Умножаем верхнюю часть первой половины чисел
pmulhw mm6, mm0

paddsw mm6, mm4

paddsw mm6, mm3 // ласт сложение
movq[F], mm6 // Добавляем первые 4 числа в результат

paddsw mm5, mm7
pxor mm7, mm7
movq mm7, mm5

pmullw mm7, mm1
pmulhw mm5, mm1

paddsw mm7, mm5

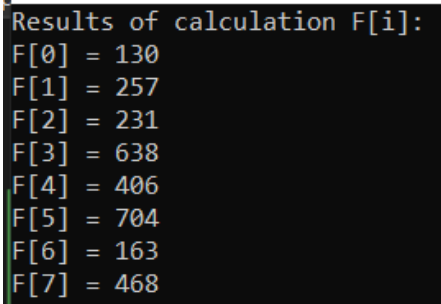
paddsw mm7, mm2 // ласт сложение

movq[F + 8], mm7 // Добавляем последние 4 числа в результат
}

std::cout << "Results of calculation F[i]: \n";
for (int i = 0; i < 8; ++i) {
    std::cout << "F[" << i << "] = " << F[i] << '\n';
}
return 0;
}

```

На рисунке 1 представлен результат выполнения программы.



```

Results of calculation F[i]:
F[0] = 130
F[1] = 257
F[2] = 231
F[3] = 638
F[4] = 406
F[5] = 704
F[6] = 163
F[7] = 468

```

Рисунок 1 – Результат выполнения программы

**Выводы:** в результате лабораторной работы было получено представление о принципах работы с технологией MMX, о ее назначении, типах данных системе команд, а также выполнено практическое задание.