**Форматы данных и команды их обработки процессоров Intel Pentium III и Intel Pentium IV**

**АННОТАЦИЯ**

В курсовой работе представлены данные о всевозможных форматах данных процессоров Intel Pentium III и Intel Pentium IV. Так же приведён полный список команд, реализованных в данных процессорах, с кратким описанием для обработки этих данных. Более подробно рассмотрены команды блока XMM: SSE – Pentium III и SSE2 – Pentium IV, с подробным описанием: для данных команд предсталено описание синтксиса, правила построения машинного кода, принцип действия (для многих команд принцип действия для большей наглядности представлен графически, ввиде схем), воздействие команды на флаги процессора, возможные возникаемые исключения во время выполнения команд.

**СОДЕРЖАНИЕ**

Введение. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1. Типы данных процессоров Pentium 3, Pentium 4 . . . . . . . . . . . . 5

2. Математический сопроцессор x87 . . . . . . . . . . . . . . . . 9

3. Технология MMX . . . . . . . . . . . . . . . . . . . . . . . 12

4. Расширение SSE и SSE2 — блок XMM. . . . . . . . . . . . . . . 14

5. Команды обработки данных. . . . . . . . . . . . . . . . . . . 15

6. Команды блока XMM (SSE и SSE2) . . . . . . . . . . . . . . . . 31

6.1. Команды блока XMM (SSE – Pentium 3) . . . . . . . . . . . 33

6.2. Команды блока XMM (SSE2 – Pentium 4) . . . . . . . . . . . 50

Литература . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89

**ВВЕДЕНИЕ**

 Целью курсовой работы «Форматы данных и команды их обработки процессоров Intel Pentium III, Intel Pentium IV» является поытка наиболее полно показать отличительные признаки современных процессоров. В данной работе автор не ставил себе задачу расписать каждую команду обработки данных существующую в данных процессорах, ведь отличительная особенность процессоров Intel заключается в том, что наиболее поздние модификации полностью совместимы с более ранними. В процессорах Intel Pentium III, Intel Pentium IV нововведением стали два блока XMM (eXtended MultiMedia) – это SSE (Streaming SIMD (Single Istruction Multiply Data) Extensions) – введённый в процессоре Intel Pentium III и SSE2 – введённый в процессоре Intel Pentium IV. Поэтому базовые команды (существующие в более ранних модификациях Intel, начиная с 8086), команды математического сопроцессора (FPU) и команды блока MMX, появившегося впервые в процессоре Intel Pentium рассмотрены лишь обзорно. Так как расширение 3Dnow! блока MMX, введённое фирмой AMD в процессорах K6-2 на данный момент отсутствует в процессорах фирмы Intel, то оно совсем не рассмотренно в курсовой работе.

1. Типы данных процессоров Pentium 3, Pentium 4

История 32-разрядных процессоров Intel Pentium 3 и Pentium 4 началась с процессора Intel386. Он вобрал в себя все черты своих 16-разрядных предшественников 8086/88 и 80286 для обеспечения совместимости с громадным объемом программного обеспечения, существовавшего на момент его появления. Процессоры могут оперировать с 8-,16- и 32-битными операндами, строками байт, слов и двойных слов, а также с битами, битовыми полями и строками бит.

Рассматриваемые процессоры непосредственно поддерживают (используют в качестве операндов) знаковые и беззнаковые целые числа, строки байт, цифр и символов, битовые строки, указатели и числа с плавающей точкой. В семействе х86 принято, что слова записываются в двух смежных байтах памяти, начиная с младшего. Адресом слова является адрес его младшего байта. Двойные слова записываются в четырех смежных байтах, опять-таки начиная с младшего байта, адрес которого и является адресом двойного слова. Этот порядок называется Little-Endian Memory Format. В других семействах процессоров применяют и обратный порядок — Big-Endian Memory Format, в котором адресом слова (двойного слова) является адрес его старшего байта, а младшие байты располагаются в последующих адресах. Для взаимного преобразования форматов слова имеется инструкция XCHG, двойного слова — BSWAP (процессор 486 и выше).

На рис.1. приведены форматы данных, обрабатываемых целочисленным блоком АЛУ всех 32-разрядных процессоров:

• Бит (Bit) — единица информации. Бит в памяти задается базой (адресом слова) и смещением (номером бита в слове).

• Битовое поле (Bit Field) — группа до 32 смежных бит, располагающихся не более чем в 4 байтах.

• Битовая строка (Bit String) — набор смежных бит длиной до 4 Гбит.

• Байт (Byte) — 8 бит.

• Числа без знака: байт/слово/двойное/учетверенное слово (Unsigned Byte/ Word/Double Word/Quade Word), 8/16/32/64 бит.

• Целые числа со знаком: байт/слово/двойное/учетверенное слово (Integer Byte/ Word/Double Word/Quade Word). Единичное значение самого старшего бита (знак) является признаком отрицательного числа, которое хранится в дополнительном коде.

• Двоично-десятичные числа (BCD — Binary Coded Decimal):

• 8-разрядные упакованные (Packed BCD), содержащие два десятичных разряда в одном байте;

• 8-разрядные неупакованные (Unpacked BCD), содержащие один десятичный разряд в байте (значение бит 7:4 при сложении и вычитании несущественно, при умножении и делении они должны быть нулевыми).

• Строки байт, слов и двойных слов (Bit String, Byte String, Word String, Double Word String) длиной до 4 Гбайт.

• Указатели:

• длинный указатель (48 бит) — 16-битный селектор (или сегмент) и 32-битное смещение;

• короткий указатель — 32-битное смещение;

• просто указатель (32 бит, единственный тип указателя для 8086 и 80286) • 16-битный селектор (или сегмент) и 16-битное смещение.

16-разрядные процессоры из приведенных типов данных не поддерживают учетверенные слова всех типов, битовые поля и строки, строки двойных слов, короткие и длинные указатели.

Числа в формате с плавающей точкой и упакованные 80-битные BCD-числа обрабатываются блоками FPU процессоров класса 486 и выше, а также сопроцессорами 8087/287/387. Упакованные 64-битные и 128-битные данные обрабатываются процессорами с ММХ и SSE. Форматы данных, обрабатываемых блоками FPU/MMX и ХММ, представлены на рис. 2.

• Действительные числа в формате с плавающей точкой:

• одинарной точности (Single Precision), 32 бит — 23 бит мантисса, 8 бит порядок;

• двойной точности (Double Precision), 64 бит — 52 бит мантисса, 11 бит порядок;

• повышенной точности (Extended Precision), 80 бит — 64 бит мантисса, 15 бит порядок.

• Двоично-десятичные 80-битные упакованные числа (18 десятичных разрядов и знак).

• Упакованные действительные числа одинарной точности в формате с плавающей точкой, обрабатываются блоком ХММ.

• Упакованные целые числа, знаковые и беззнаковые, обрабатываются блоком ММХ:

• упакованные байты (Packed byte) — восемь байт;

• упакованные слова (Packed word) — четыре слова;

• упакованные двойные слова (Packed doubleword) — два двойных слова;

• учетверенное слово (Quadword) — одно слово.

Для 16-разрядных процессоров, естественно, все форматы чисел для блоков ММХ и ХММ недоступны.

Рассмотрим более подробно блоки, упомянутые выше: блоки FPU, MMX, XMM, которые архитектуре процессоров IA-32 х86 держатся особняком. Они присутствуют не но всех процессорах и даже по схемотехнической реализации являются пристройками к центральному процессору с его набором обычных целочисленных регистров. Данные блоки предназначены для ускорения вычислений.

Математический сопроцессор (FPU) позволяет использовать несколько форматов чисел с плавающей точкой — FP-форматов. Операции с такими числами можно выполнять и программно средствами целочисленного процессора, но сопроцессор выполняет эти операции анпаратно во много раз быстрее. Блок ММХ дает ускорение целочисленных вычислений за счет одновременной обработки одной инструкцией целого пакета чисел (пар чисел). Блок ХММ комбинирует эти два приема — обрабатывает одной инструкцией пакет из четырех чисел в FP-формате. Исторически первым появился сопроцессор. Блок ММХ ради совместимости с операционными системами "спрятали" в то же оборудование, что и сопроцессор. Так появился комбинированный блок, называемый блоком FP/MMX, или FPU/ММХ. И толысо блок ХММ, используемый расширением SSE процессоров Pentium 3, стал полностью новым самостоятельным набором регистров.

Рис. 1. Типы данных, обрабатываемых целочисленным АЛУ

Рис. 2. Типы данных, обрабатываемых блоками FPU/MMX и ХММ

При отсутствии математического сопроцессора прикладная программа все-таки может использовать инструкции FPU, но для этого операционная система должна поддерживать эмуляцию сопроцессора. Эмулятор сопроцессора — это программа-обработчик прерывания от сопроцессора или исключения #NM, которая должна "выловить" код операции, сопроцессора, определить местонахождение данных и выполнить требуемые вычисления, используя целочисленную арифметику центрального процессора. Понятно, что эмуляция будет выполняться во много раз медленнее, чем те же действия, выполняемые настоящим сопроцессором. Тем не менее эмуляция позволяет все-таки пользоваться прикладными программами, требующими вычислений с плавающей точкой. Для этого в регистре CR0 должно быть установлено сочетание флагов ЕМ = 1, МР = 0. Для эмуляции в IBM PC обычно устанавливают значение NE = 0. Тогда каждая инструкция FPU автоматически будет вызывать эмулятор генерацией запроса прерывания (а не исключения #NM, как было бы при NE = 1).

Эмуляция для блоков ММХ и ХММ не предусматривается — эти блоки предназначены для ускорения вычислений в приложениях реального времени, и выполнять их с крайне низкой скоростью эмуляции было бы просто бессмысленно. Если установлен флаг эмуляции ЕМ = 1, то любая инструкция ММХ вызовет исключение #UD.

1. Математический сопроцессор x87

Математический сопроцессор предназначен для расширения вычислительных возможностей центрального процессора — выполнения арифметических операций, вычисления основных математических функции (тригонометрических, экспоненты, логарифма) и т. д. В разных поколениях процессоров он назывался по-разному — FPU (Floating Point Unit — блок чисел с плавающей точкой) или NPX (Numeric Processor eXtension — числовое расширение процессора).

Сопроцессор поддерживает семь типов данных: 16-, 32-, 64-битные целые числа; 32-, 64-, 80-битные числа с плавающей точкой и 18-разрядные числа в двоично-десятичном формате. Формат чисел с плавающей точкой соответствует стандартам IEEE 754 и 854. Применение сопроцессора повышает производительность вычислений в сотни раз. С программной точки зрения сопроцессор и процессор выглядят как единое целое. В современных (486+) процессорах FPU располагается на одном кристалле с центральным процессором. Для процессоров 386 и ниже сопроцессор был отдельной микросхемой, подключаемой к локальной нише основного процессора. В любом случае сопроцессор исполняет только свои специфические команды, а всю работу по декодированию инструкции и доставке данных осуществляет CPU. Сопроцессор может выполнять вычисления параллельно с центральным процессором, независимо от переключения задач в защищенном режиме. Как и основной процессор, сопроцессор может работать в реальном или защищенном режиме и переключать разрядность- 16 или 32. Переключение режимов влияет на формат отображения регистров сопроцессора в оперативной памяти, при этом формат используемых внутренних регистров не изменяется.

Форматы данных FPU

Сопроцессор оперирует данными в формате с плавающей точкой, который позволяет представлять существенно больше действительных чисел, чем целочисленное АЛУ центрального процессора. Арифметические операции (здесь под арифметическими понимаются операции, изменяющие значения операндов, а также операции сравнения) в FPU выполняются над 80-битными числами, преде га пленными во внутреннем формате расширенной точности (рис. 3). Формат позволяет представлять следующие категории чисел:

• нули (положительный и отрицательный) — оба значения эквивалентны;

• денормализованные конечные числа (положительные и отрицательные);

• нормализованные конечные числа (положительные и отрицательные);

• бесконечность (положительная и отрицательная).

Числа представляются в аффинном пространстве. Это означает, что меньше любого конечного числа, а больше любого конечного числа.

Рис. 3. Формат внутреннего представления чисел.

Бит Sign определяет знак числа: 0 — положительное, 1 — отрицательное число. Поле Exponent хранит смещенное значение двоичного порядка числа (biased exponent). Смещение позволяет все значения порядков допустимого диапазона чисел представлять положительным числом, при этом значению 000...000 соответствуют минимальные (по модулю) числа, значению 111...110 — максимальные допустимые числа, а значению 111...111 — бесконечно большие числа. Нуль может быть положительным или отрицательным, в зависимости от бита знака, при этом и мантисса, и порядок у него нулевые. Мантисса (Signficand) нормализованного числа, отличного от нуля, всегда имеет вид "1,ххх....ххх", то есть представляет величину, не меньшую единицы. У бесконечностей (тоже положительной и отрицательной) мантисса нулевая. Денормализованные числа имеют нулевой порядок (смещенное значение) и мантиссу вида "0,ххх...ххх" (отличную от нуля). Денормализованные числа — это слишком малые величины, которые представляются и обрабатываются с точностью меньшей, чем позволяет разрядность регистров сопроцессора.

Кроме вещественных чисел (конечных нормализованных и денормализованных, нулей и бесконечностей) регистры сопроцессора могут содержать не-числа NaN (Not a Number) четырех видов.

• -SNaN и +SNaN — порядок 111...111, мантисса 1,0ххх...ххх (ненулевая). Эти "сигнализирующие" не-числа (signaling NaN) вызывают исключения сопроцессора, если с ними пытаются выполнять арифметические действия.

• -QNaN и +QNaN — порядок 111...111, мантисса 1,1ххх...ххх (ненулевая). Эти "тихие" не-числа (quiet NaN) не вызывают исключений при арифметических операциях.

Внешние операнды могут быть представлены в одном из форматов, приведенных на рис. 2. Характеристики форматов чисел, поддерживаемых сопроцессором, приведены в таблице. При их загрузке в FPU и сохранении результатов преобразования форматов во внутренний и обратно выполняются автоматически. Во внешних представлениях вещественных чисел целая часть мантиссы всегда подразумевается равной единице. В расширенном формате целая часть задается явно (бит 63), она имеет нулевое значение только при представлении нулей и денормализованных чисел. Смещение порядка составляет 127 для одиночного, 1023 для двойного и 16 383 для расширенного вещественного форматов. Форматы вещественных чисел представляют только множество дискретных значений множества чисел, расположенных на непрерывной бесконечной числовой оси. Диапазон и плотность значений зависят от выбранного формата представления. Заметим, что не все десятичные дроби могут быть представлены точно в двоичном коде. Так, например, дробь 1/10 не имеет точного двоичного представления (аналогично тому, что 1/3 = 0,33333(3)).

Таблица. Форматы чисел, поддерживаемых сопроцессором

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | Длина, бит | Точность | Диапазон нормализованных значений |
| двоичная | десятичная | Двоичный | десятичный |
| Вещественные числа |
| Одиночные (single) | 32 | 24 | 7 |  |  |
| Двойные (double) | 64 | 53 | 15-16 |  |  |
| Рассширенные (extended) | 80 | 64 | 19 |  |  |
| Двоичные целые |
| Слова (word) | 16 | 15 | 4 |  |  |
| Короткие (short) | 32 | 31 | 9 |  |  |
| Длинные (long) | 64 | 63 | 18 |  |  |
| Упакованные двоично-десятичные |
| BCD | 80 | — | 18 | — |  |

Сопроцессор контролирует числа, участвующие в арифметических операциях. При загрузке денормализованного операнда в регистр FPU и попытке выполнения арифметических инструкций хотя бы с одним денормализованным операндом сопроцессор фиксирует условие исключения #D. Денормализованные числа могут появляться при выполнении вычислений, в этом случае сопроцессор фиксирует факт исчезновения значащих разрядов и генерирует исключение #U. При попытке выполнения арифметических операций с нечислами, а также с недопустимыми значениями операндов (например, извлечение квадратного корня из отрицательного числа) вырабатывается исключение #I. При переполнении вырабатывается исключение #O, при попытке деления на нуль ненулевого операнда вырабатывается исключение #Z.

Если результат вычисления невозможно представить точно в выбранном формате, сопроцессор выполняет округление результата в сторону соседнего допустимого значения. Правила округления программируются. Вместо автоматического выполнения округления сопроцессор может вырабатывать исключение #Р.

1. Технология MMX

Технология ММХ ориентирована на приложения мультимедиа, 2D/3D-графикy и коммуникации. Это расширение базовой архитектуры появилось только после выхода второго поколения процессоров Pentium. Основная идея ММХ заключается в одновременной обработке нескольких элементов данных за одну инструкцию — так называемая технология SIMD (Single Instruction — Multiple Data). Расширение ММХ использует новые типы упакованных 64-битных целочисленных данных:

• упакованные байты (Packed byte) — восемь байт;

• упакованные слова (Packed word) — четыре слова;

• упакованные двойные слова (Packed doubleword) — два двойных слова;

• учетверенное слово (Quadword) — одно слово.

Эти типы данных могут специальным образом обрабатываться в 64-битных регистрах ММХ0-ММХ7, представляющих собой младшие биты стека 80-битных регистров FPU. Каждая инструкция ММХ выполняет действие сразу над всем комплектом операндов (8, 4, 2 или 1), размещенных в адресуемых регистрах. Как и регистры FPU, эти регистры ММХ не могут использоваться для адресации памяти. Совпадение регистров ММХ и FPU накладывает ограничения на чередование кодов FPU и ММХ — забота об этом лежит на программисте приложений с ММХ.

Еще одна особенность технологии ММХ — поддержка арифметики с насыщением (saturating arithmetic). Ее отличие от обычной арифметики с циклическим переполнением (wraparound mode) заключается в том, что при возникновении переполнения в результате фиксируется максимально возможное значение для данного типа данных, а перенос игнорируется. В случае переполнения снизу в результате фиксируется минимально возможное значение. Граничные значения определяются типом (знаковый или беззнаковый) и разрядностью переменных. Такой режим вычислений удобен, например, для определения цветов.

В систему команд введено 57 дополнительных инструкций для одновременной обработки нескольких единиц данных. Одновременно обрабатываемое 64-битное слово может содержать как одну единицу обработки, так и 8 однобайтных, 4 двухбайтных или 2 четырехбайтных операнда. Новые инструкции включают следующие группы:

• арифметические (Arithmetic Instructions), куда входят сложение и вычитание в разных режимах, умножение и комбинация умножения и сложения;

• сравнение (Comparison Instructions) элементов данных на равенство или по величине;

• преобразование форматов (Conversion Instructions);

• логические инструкции (Logical Instructions) — И, И-НЕ, ИЛИ и исключающее ИЛИ, выполняемые над 64-битными операндами;

• сдвиги (Shift Instructions) — логические и арифметические;

• пересылки данных (Data Transfer Instructions) между регистрами ММХ и целочисленными регистрами или памятью;

• очистка ММХ (Empty ММХ State) — установка признаков пустых регистров в слове тегов.

Инструкции ММХ не влияют на флаги условий в слове состояния FPU.

Регистры ММХ в отличие от регистров FPU адресуются физически, а не относительно значения указателя стека ТОР. Более того, любая инструкция ММХ обнуляет поле ТОР регистра состояния FPU. В слове тегов свободному регистру соответствует комбинация 11, остальные комбинации указывают только на занятость регистра. После каждой операции ММХ биты тегов регистра назначения обнуляются. Неиспользуемые в ММХ биты [79:64] регистров FPU заполняются единицами, так что ошибочная обработка данных ММХ инструкцией FPU приведет к исключению.

Инструкции ММХ не порождают новых исключений. Исключения при выполнении инструкций ММХ могут возникать только в случае нарушения границ в обращениях к памяти (как при обмене данными, так и при выборке инструкции). Однако если предшествующая инструкция FPU породила условие исключения, то оно произойдет при выполнении инструкции ММХ. После его обработки инструкция ММХ может исполнена.

С инструкциями ММХ могут применяться префиксы замены сегмента и изменения разрядности адреса (влияют на инструкции, обращающиеся к памяти). Использование префиксов изменения разрядности операнда и повторов зарезервировано (может привести к непредсказуемым результатам). Префикс Lock вызывает исключение #UD.

Инструкции ММХ доступны из любого режима процессора. При переключении задач необходимо следить за корректностью сохранения контекста, как и при работе с FPU.

Любая инструкция ММХ вызывает обнуление полей тегов всех регистров FPU/ММХ, что для FPU означает наличие действительных данных во всех регистрах. Последующая инструкция для FPU над "неправильными" данными может привести к непредсказуемому результату, поскольку "входной контроль" данных осуществляется по состоянию тегов. Чтобы застраховаться от подобных неприятностей, после инструкций ММХ и перед инструкциями FPU в программный код вводят инструкцию EMMS, которая устанавливает в слове тегов значение FFFFh (все регистры пустые).

Различие в способе адресации регистров (относительная для FPU и явная прямая в ММХ), обнуление тегов инструкциями ММХ и некоторые другие нюансы не позволяют чередовать инструкции FPU и ММХ. Блок FPU/MMX может работать либо в одном, либо в другом режиме. Если, к примеру, в цепочку инструкций FPU нужно вклинить инструкции ММХ, после чего продолжить вычисления FPU, то перед первой инструкцией ММХ приходится сохранять контекст (состояние регистров) FPU в памяти, а после этих инструкций снова загружать контекст. На эти сохранения и загрузки расходуется процессорное время, в результате возможна полная потеря выигрыша от реализации технологии SIMD. Совпадение регистров ММХ и FPU оправдывают тем, что для сохранения контекста ММХ при переключении задач не требуется доработок в операционной системе — контекст ММХ сохраняется тем же способом, что и FPU, с которым умели работать издавна. Таким образом, операционным системам было все равно, какой процессор установлен — с ММХ или без. Но для того чтобы реализовать преимущества SIMD, приложения должны "уметь" ими пользоваться (и не проиграть на переключениях).

Частое чередование кодов FPU и ММХ может снизить производительность за счет необходимости сохранения и восстановления весьма объемного контекста FPU.

1. Расширение SSE и SSE2 — блок XMM

Процессоры Pentium 3 имеют так называемое потоковое расширение SSE (Streaming SIMD Extensions). В те времена, когда будущий Pentium III называли еще Kathmai, фирма Intel объявила о новых инструкциях KNI (Kathmai New Instruction), так что SSE — это синоним "староинтеловского" KNI. Новые процессоры имеют дополнительный независимый блок из восьми 128-битных регистров, названных ХММ0...ХММ7 (очевидно, eXtended MultiMedia), и регистр состояния/управления MXCSR. В каждый из регистров ХММ помещаются четыре 32-битных числа в формате с плавающей точкой одинарной точности. Блок позволяет выполнять векторные (они же пакетные) и скалярные инструкции. Векторные инструкции реализуют операции сразу над четырьмя комплектами операндов. Скалярные инструкции работают с одним комплектом операндов — младшим 32-битным словом. При выполнении инструкций с ХММ традиционное оборудование FPU/MMX не используется, что позволяет эффективно смешивать инструкции ММХ с инструкциями над операндами с плавающей точкой. Здесь блоки процессора меняются ролями — регистры ММХ, наложенные на регистры традиционного сопроцессора, используются для целочисленных потоковых вычислений, а вычисления с плавающей точкой (правда, только с одинарной точностью, но для мультимедийпых приложений ее хватает) возлагаются на новый блок ХММ. Кроме инструкций с новым блоком ХММ в расширение SSE входят и дополнительные целочисленные инструкции с регистрами ММХ, а также инструкции управления кэшированием. Новые инструкции с регистрами ММХ, как и их предшественники из "классического" ММХ, не допускают чередования с инструкциями FPU без переключения контекста FPU/MMX.

С инструкциями SSE могут использоваться префиксы замены сегмента и изменения разрядности адреса (влияют на инструкции, обращающиеся к памяти). Использование префиксов изменения разрядности операнда зарезервировано (может привести к непредсказуемым результатам). Префикс Lock вызывает исключение #UD. Из префиксов повтора можно использовать только безусловный (REP) и только для "потоковых" инструкций (с ХММ), Остальные применения префиксов повтора могут привести к непредсказуемым результатам.

В процессоре Pentium 4 набор инструкций получил очередное расширение — SSE2, в основном касающееся добавления новых типов 128-битных операндов для блока ХММ:

• упакованная пара вещественных чисел двойной точности;

• упакованные целые числа: 16 байт, 8 слов, 4 двойных слова или пара учетверенных (по 64 бита) слов.

В процессор введены новые функции целочисленной арифметики SIMD, 128-разрядные для регистров ХММ и такие же 64-разрядные для регистров ММХ; ряд старых инструкций ММХ распространили и на ХММ (в 128-битном варианте); добавлены инструкции преобразований для новых форматов данных, а также расширены возможности "перемешивания" данных в блоке ХММ. Кроме того, расширена поддержка управления кэшированием и порядком исполнения операций с памятью. Инструкции SSE2 предназначены для ЗD-графики, кодирования/декодирования видео, а также шифрования данных.

5. Команды обработки данных

Система команд 32-разрядных процессоров является существенно расширенной системой команд процессоров 8086/80286. Расширения касаются увеличения разрядности адресов и операндов, более гибкой системы адресации, появления принципиально новых типов данных (битовые строки и поля) и команд.

Команды (инструкции) содержат одно- или двухбайтный код инструкции, за которым может следовать несколько байт, определяющих режим исполнения команды, и операнды. Команды могут использовать до трех операндов (или ни одного). Операнды могут находиться в памяти, регистрах процессора или непосредственно в команде. Для 32-разрядных процессоров разрядность слова (word) по умолчанию может составлять 32, а не 16 бит. Это распространяется на многие инструкции, включая и строковые. В реальном режиме и режиме виртуального процессора 8086 по умолчанию используется 16-битная адресация и 16-битные операнды-слова. В защищенном режиме режим адресации и разрядность слов по умолчанию определяются дескриптором кодового сегмента. Перед любой инструкцией может быть указан префикс переключения разрядности адреса или слова. При адресации памяти использование сегментного регистра, предусмотренного командой, в ряде инструкций может подавляться префиксом изменения сегмента (Segment Override).

В системе команд насчитывается несколько сотен инструкций, поэтому в данной работе обзорно рассмотрены все команды обработки данных (блоков процессора АЛУ, FPU, MMX, и XMM), а далее более подробно описаны инструкции, появившиеся в процессорах Pentium 3 (блок XMM — SSE) и Pentium 4 (блок XMM — SSE2).

Инструкции пересылки данных (см. табл) позволяют передавать константы или переменные между регистрами и памятью, а также портами ввода-вывода в различных комбинациях, но в памяти может находиться не более одного операнда. В эту группу отнесены и инструкции преобразования форматов — расширений и перестановки байт. Операции со стеком выполняются словами с разрядностью, определяемой текущим режимом. При помещении в стек слова указатель стека SP уменьшается на число байт слова (2 или 4), при извлечении — увеличивается. "Классические" (8086) инструкции пересылки не влияют на содержимое регистра флагов. Инструкции пересылки по результатам сравнения (CMPXCHG) модифицируют флаг ZF. Новые инструкции условной пересылки (CMOVxx) позволяют сократить число ветвлений в программе.

Таблица. Инструкции пересылки данных

Инструкция Описание

BSWAP Перестановка байт из порядка младший-старший (L-H) в порядок старший-младший (H-L) (486+)

CBW/CWDE Преобразование байта AL в слово АХ (расширение знака AL в АН: АН заполняется битом AL.7) или слова АХ в двойное слово ЕАХ

CMOVA/CMOVNBE Пересылка, если выше "CF ИЛИ ZF)=0) (P6+)

CMOVAE/CMOVNB Пересылка, если не ниже (CF=0) (P6+)

CMOVB/CMOVNAE Пересылка, если ниже (CF=1) (P6+)

CMOVBE/CMOVNA Пересылка, если не выше ((CF ИЛИ ZF)=1) (P6+)

CMOVC Пересылка, если перенос (CF=1) (P6+)

CMOVE/CMOVZ Пересылка, если равно (ZF=1) (P6+)

CMOVG/CMOVNLE Пересылка, если больше (SF=(0F И ZF)) (P6+)

CMOVGE/CMOVNL Пересылка, если больше или равно (SF=0F) (P6+)

CMOVL/CMOVNGE Пересылка, если меньше (ZF0F) (P6+)

CMOVLE/CMOVNG Пересылка, если меньше или равно (SF0F или ZF=0) (P6+)

CMOVNC Пересылка, если нет переноса (CF=0) (P6+)

CMOVNE/CMOVNZ Пересылка, если не равно (ZF=0) (P6+)

CMOVNO Пересылка, если нет переполнения (0F=0) (P6+)

CMOVNP/CMOVPO Пересылка, если нет паритета (нечетность) (P6+)

CMOVNS Пересылка, если неотрицательно (SF=0) (P6+)

CMOVO Пересылка, если переполнение (0F=1) (P6+)

CMOVP/CMOVPE Пересылка, если паритет (четность) (Р6+)

CMOVS Пересылка, если отрицательно (SF=1)(P6+)

CMPXCHG r/in,r Обмен по результату сравнения байта, слова или двойного слова (486+)

CMPXCHG8B m64 Обмен по результату сравнения учетверенного слова (5+)

CWD/CDQ Преобразование слова АХ в двойное слово DX:AX (расширение знака, DX заполняется битом АХ. 15) или двойного слова ЕАХ в учетверенное EDX:EAX

IN Ввод из порта ввода-вывода в AL/(E)AX

MOV Пересылка(копирование)данных

MOVSX Копирование байта/слова со знаковым расширением до слова/ двойного слова(386+)

MOVZX Копирование байта/слова с нулевым расширением до слова/ двойного слова(386+)

OUT Вывод в порт из AL/(E)AX

POP Извлечение слова данных из стека в регистр или память, (E)SP инкрементируется

POPA(POPAll) Извлечение данных из стека в регистры Dl, SI, ВР, ВХ, DX, CX, AX (286+)

POPAD Извлечение данных из стека в регистры EDI, ESI, ЕВР, ЕВХ, EDX, ЕСХ, ЕАХ (386+)

PUSH Помещение слова из регистра или памяти в стек после декремента (E)SP

PUSHA (PUSH All) Помещение в стек регистров АХ, CX, DX, BX, SP (исходное значение), ВР, SI, Dl (286+)

PUSHAD Помещение в стек регистров ЕАХ, ЕСХ, EDX, ЕВХ, ESP (исходное значение), ЕВР, ESI, EDI (386+)

XCHG Обмен данными (взаимный) между регистрами или регистром и памятью

Инструкции ввода-вывода позволяют пересылать как одиночный бант или слово между портом и регистром процессора (инструкции IN и OUT), так и блок байт (слов) между портом и группой смежных ячеек памяти (инструкции INSB/INSW и OUTSB/OUTSW с префиксом повтора, см. ниже). Непосредственная адресация порта в команде обеспечивает доступ только к первым 256 адресам портов, косвенная (через регистр DX) — ко всему пространству ввода-вывода (64 Кбайт). Разрядность операнда и адрес должны согласовываться с физическими возможностями и особенностями поведения адресуемого устройства. При работе с памятью такие нюансы во внимание принимать обычно не приходится.

Инструкции двоичной арифметики выполняют все арифметические действия с байтами, словами и двойными словами, кодирующими знаковые или беззнаковые целые числа. Умножение и деление для 8086 возможны только с аккумулятором, результат для 16-битных операндов расширяется в регистре DX.

Для 286+ возможно двух- и трехадресное умножение с расширенном тилько в старший байт (два байта для 386+).

Таблица. Инструкции двоичной арифметики

Инструкция Описание

ADC Сложение двух операндов с учетом переноса от предыдущей операции

ADD Сложение двух операндов

СМР Сравнение (вычитание без сохранения результата — установка флагов)

DEC Декремент (вычитание 1, но не действует на флаг CF)

DIV Деление беззнаковое

IDIV Деление знаковое

IMUL Умножение знаковое

INC Инкремент (сложение с 1, но не действует на флаг CF)

MUL Беззнаковое умножение

NEG Изменение знака операнда

SBB Вычитание с заемом

SUB Вычитание

XADD Обмен содержимым и сложение (486+)

Инструкции десятичной арифметики являются дополнением к предыдущим. Они позволяют оперировать с неупакованными (биты [7:4] = 0, биты [3:0] содержат десятичную цифру 0-9) или упакованными (биты [7:4] содержат старшую, биты [3:0] — младшую десятичную цифру 0-9) двоичнодесятичными числами. Арифметические операции над этими числами требуют применения инструкций коррекции форматов.

Таблица. Инструкции десятичной арифметики

Инструкция Описание

ААА Десятичная коррекция после сложения двух неупакованных чисел

AAD Десятичная коррекция перед делением неупакованного двузначного числа

ААМ Десятичная коррекция после умножения двух неупакованных чисел

AAS Десятичная коррекция после вычитания двух неупакованных чисел

DAA Десятичная коррекция AL после сложения двух упакованных чисел

DAS Десятичная коррекция AL после вычитания двух упакованных чисел

Инструкции AAD и ААМ допускают обобщенный формат вызова, при котором коррекция выполняется но любому модулю (а не только по модулю 10).

Инструкции логических операций выполняют все функции булевой алгебры над байтами, словами или двойными словами.

Таблица. Инструкции логических операций

Инструкция ОписаниеAND Логическое ИNOT Инверсия (переключение всех бит)OR Логическое ИЛИXOR Исключающее ИЛИ

Сдвиги и вращения (циклические сдвиги) выполняются над регистром или операндом в памяти. Число позиций, на которое производится сдвиг, берется непосредственно из операнда или регистра CL по модулю 8 для однобайтного операнда и по модулю 16 или 32 для операнда-слова, в зависимости от разрядности данных (32 только для 386+). Биты, выталкиваемые при сдвигах, попадают во флаг CF. При сдвигах влево и простом сдвиге вправо освобождающиеся биты заполняются нулями (инструкции SAL и SHL — синонимы). При арифметическом сдвиге вправо старший бит (знак) сохраняет свое значение. При циклических сдвигах выталкиваемые биты попадают и во флаг CF, и в освобождающиеся позиции. В сдвигах могут участвовать и два операнда (инструкции SHLD и SHRD).

Таблица. Инструкции сдвигов

Инструкция ОписаниеRCL Циклический сдвиг влево через бит переноса

RCR Циклический сдвиг вправо через бит переноса

ROL Циклический сдвиг влево

ROR Циклический сдвиг вправо

SAL Сдвиг арифметический влево

SAR Сдвиг арифметический (с сохранением старшего бита) вправо

SHL Сдвиг влево

SHR Сдвиг вправо

SHLD Сдвиг влево и вставка данных в освободившиеся позиции (386+)

SHRD Сдвиг вправо и вставка данных в освободившиеся позиции (386+)

Инструкции обработки бит и байт позволяют проверять (копировать в CF) и устанавливать значение указанного операнда, а также искать установленный бит. Битовые операции выполняются над 16-или 32-битным словом памяти или регистром. Инструкции BSF, BSR и ВТ не изменяют значения слова; ВТС, BTR и BTS воздействуют на указанный бит слова. Номер интересующего бита берется из операнда по модулю 16 или 32, в зависимости от разрядности.

Операции с байтами обеспечивают условную установку значений 00h или 01h. Инструкция тестирования может выполняться над байтом, словом или двойным словом.

Таблица. Инструкции обработки бит и байт

Инструкция ОписаниеBSF Сканирование бит (поиск единичного) впередBSR Сканирование бит назадВТ Тестирование бита (загрузка в CF)

ВТС Тестирование и изменения значения бита

BTR Тестирование и сброс бита

BTS Тестирование и установка бита

SALC Условная (по CF) установка А1 в FFh или OOh (не документировано, код D6h)

SETA/ Установка байта в 01h, если выше ((CF ИЛИ ZF)=0), иначе в 00h

SETNBE

SETAE/ Установка байта в 01 h, если не ниже (CF=0), иначе в 00h

SETNB/

SETNC

SETB/ Установка байта в 01h, если ниже (CF=1), иначе в 00h

SETNAE/

SETC

SETBE/ Установка байта в 01h, если не выше (CF ИЛИ ZF)=1, иначе в 00h

SETNA

SETE/ Установка байта в 01h, если равно (ZF=1), иначе в 00h

SETZ

SETG/ Установка байта в 01 h, если больше (SF=(OP И ZF)), иначе в 00h

SETNLE

SETQE/ Установка байта в 01h, если больше или равно (SF=OF), иначе в 00h

SETNL

SETL/ Установка байта в 01h, если меньше (ZFOF), иначе в 00h

SETNGE 00h

SETLE/ Установка байта в 01h, если меньше или равно (SF0F или ZF=0),иначе в 00h

SETNG

SETNE/ Установка байта в 01h, если не равно (ZF=0), иначе в 00h

SETNZ

SETNO Установка байта в 01h, если нет переполнения (0F=0), иначе в 00h

SETNS Установка байта в 01 h, если неотрицательно (SF=0), иначе в 00h

SETO Установка байта в 01h, если переполнение (0F=1), иначе в 00h

SETPE/ Установка байта в 01h, если паритет (четность), иначе в 00h

SETP

SETPO/ Установка байта в 01 h, если нет паритета (нечетность), иначе в 00h

SETNP

SETS Установка байта в 01 h, если отрицательно (SF=1), иначе в 00h

SETC Установка байта в 01 h, если перенос (CF=1), иначе в 00h

SETNC Установка байта в 01 h, если нет переноса (CF=0), иначе в 00h

TEST Проверка бит (логическое И без записи результата — установка флагов)

Строковые операции выполняются с операндами в памяти, адресуемыми регистрами DS:SI (DS:ESI) для источника и ES:DI (ES:EDI) для приемника. Операции могут использоваться с префиксами условного или безусловного повтора. После каждой пересылки или сравнения индексные регистры (SI, DI или оба) участвующих операндов автоматически инкрементируются или декрементируются на количество байт, участвующих в операции (1,2 или 4). Направление модификации определяется флагом DF: DF = 0 -инкремент, DF = 1 — декремент. Строковые инструкции ввода-вывода с префиксами повтора позволяют достигать высоких скоростей обмена с портами при условии полной загрузки процессора.

Таблица. Инструкции строковых операций

Инструкция Описание

CMPSB, CMPSD, CMPSW Сравнение строк байт, слов или двойных слов с записью результата сравнения в регистр флагов

INSB, INSD, INSW Запись байта, слова или двойного слова, введенного из порта, в память(286+)

LODSB, LODSD, LODSW Копирование байта, слова или двойного слова из строки в AL/(E)AX

MOVSB, MOVSD, MOVSW Копирование байта, слова или двойного слова из одной строки в другую

OUTSB, OUTSD, OUTSW Вывод байта, считанного из памяти, в порт (286+)

SCASB, SCASD, SCASW Сканирование строки байт, слов или двойных слов — сравнение с AL/(E)AX и запись результата сравнения в регистр флагов

STOSB, STOSD, STOSW Запись байта, слова или двойного слова в строку из AL/(E)AX

REP Префикс повтора строковых операций до обнуления (Е)СХ, (Е)СХ декрементируется на каждом повторе

REPE/REPZ Префикс условного повтора строковых операций — выполнения REP при ZF=1

REPNE/ Префикс условного повтора строковых операций — выполнения

REPNZ REP при ZF=0

Инструкции математического сопроцессора (FPU) имеют свою специфику задания операндов. Переменная st(0) находится на вершине стека сопроцессора, st(i) смещена от вершины на i. Загрузка данных начинается с декремента указателя стека сопроцессора (поле TOP) — перемещения вершины. Если новая вершина не пустая (по полю TAG) или стек исчерпан, вызывается исключение с указанием причины.

После загрузки поле TAG устанавливается в соответствии с загруженным числом. При извлечении из стека производится инкремент ТОР, а в поле TAG старой вершины устанавливается признак пустой ячейки. Попытка использования пустого регистра в операциях или для сохранения результатов в памяти вызывает исключение. Инструкции с префиксом F предварительно проверяют флаг исключения ES (они называются ожидающими инструкциями), инструкции с префиксом FN флаг исключения не проверяют (неожидающие инструкции). Ряд инструкций не вызывает исключения в случае, если обнаруживаются операнды не-числа (NaN).

Таблица. Инструкции FPU

Инструкция Описание

Пересылки данных

FBLD Преобразование и помещение (push) числа в упакованном BCD-формате из памяти в стек

FBSTP Извлечение из стека и запись в память в упакованном BCD-формате (10 байт, 18 цифр)

FCMOVB Пересылка, если ниже (CF=1) (P6+)

FCMOVBE Пересылка, если не выше (CF ИЛИ ZF)=1 (P6+)

FCMOVE Пересылка, если равно (ZF=1) (P6+)

FCMOVNB Пересылка, если не ниже (CF=0) (P6+)

FCMOVNBE Пересылка, если выше ((CF ИЛИ ZF)=0) (P6+)

FCMOVNE Пересылка, если не равно (ZF=0) (P6+)

FCMOVNU Пересылка, если не NaN (PF=0) (P6+)

FCMOVU Пересылка, если NaN (unordered) (PF=0) (P6+)

FILD Загрузка (push) целого числа из памяти

FIST Запись в память в формате целого числа

FISTP Запись в память в формате целого числа с извлечением

FLD Загрузка (push) вещественного числа

FST Сохранение (копирование) числа в памяти (в вещественном формате) или в регистре стека

FSTP Запись числа в память (в вещественном формате) или в регистр стека с извлечением

FXCH Обмен значениями вершины стека и регистра

Загрузка константFLD1 Загрузка (push)+1,0FLDL2E Загрузка (push) log2(e)FLDL2T Загрузка (push) log2( 10)FLDLG2 Загрузка (push) lg(2)

FLDLN2 Загрузка (push) ln(2)

FLDPI Загрузка (push) pi

FLDZ Загрузка (push) + 0,0

Базовая арифметикаFABS Нахождение абсолютного значенияFADD Сложение вещественных чиселFADDP Сложение вещественных чисел с извлечением

FCHS Изменение знака

FDIV Деление вещественных чисел

FDIVP Деление вещественных чисел с извлечением

FDIVR Обратное деление вещественных чисел

FDIVRP Обратное деление вещественных чисел с извлечением

FIADD Сложение с целым числом

FIDIV Деление на целое число

FIDIVR Обратное деление целых чисел

FIMUL Умножение на целое число

FISUB Вычитание целого числа

FISUBR Вычитание из целого числа

FMUL Умножение вещественных чисел

FMULP Умножение вещественных чисел с извлечением

FPREM Нахождение частичного остатка

FPREM1 Нахождение частичного остатка в стандарте IEEE (387+)

FRNDINT Округление до ближайшего целого

FSCALE Масштабирование — умножение на округленную в сторону нуля степень числа 2

FSQRT Извлечение квадратного корня

FSUB Вычитание вещественного числа

FSUBP Вычитание вещественных чисел с извлечением

FSUBR Обратное вычитание числа

FSUBRP Обратное вычитание с извлечением

FXTRACT Выделение мантиссы и порядка числа

Сравнение данных

FCOM Сравнение вещественных чисел (установка флагов сопроцессора)

FCOMI Сравнение и соответствующая установка флагов в EFLAGS (ZF, PF, CF) (P6+)

FCOMIP Сравнение и соответствующая установка флагов в EFLAGS (ZF, PF, CF), с извлечением (P6+)

FCOMP Сравнение вещественных чисел с извлечением

FCOMPP Сравнение вещественных чисел с двойным извлечением

FICOM Сравнение с целочисленным операндом из памяти

FICOMP Сравнение с целочисленным операндом из памяти с извлечением

FTST Проверка на нуль

FUCOM Сравнение без генерации исключения в случае NaN (387+)

FUCOMI Сравнение без генерации исключения в случае NaN и соответствующая установка флагов в EFLAGS (ZF, PF, CF) (P6+)

FUCOMIP Сравнение без генерации исключения в случае NaN и соответствующая установка флагов в EFLAGS (ZF, PF, CF) с извлечением (P6+)

FUCOMP Сравнение без генерации исключения в случае NaN с извлечением (387+)

FUCOMPP Сравнение без генерации исключения в случае NaN с двойным извлечением (387+)

FXAM Анализ числа — установка кода условия в СО, С2, СЗ

Трансцендентные функцииР2ХМ1 Вычисление

FCOS Косинус (387+)PPATAN Арктангенс частного с извлечениемFPTAN Вычисление тангенса и загрузка (push) в стек +1,0

FSIN Вычисление синуса (387+)

FSINCOS Вычисление синуса и косинуса с помещением (push) в стек (387+)

FYL2X Вычисление Yxlog2(X)

FYL2XP1 Вычисление Yxlog2(X+1)

Управление сопроцессоромFCLEX Сброс флагов исключений с предварительной проверкой

ожидающих немаскированных исключений

FDECSTP Декремент указателя стека FPU

FFREE Освобождение регистра — пометка как свободного

FINCSTP Инкремент указателя стека FPU

FINIT Инициализация FPU с предварительной проверкой ожидающих исключений

FLDCW Загрузка управляющего слова (FPU CW) из памяти

FLDENV Загрузка состояния сопроцессора из памяти, сохраненного инструкциями FSTENV/FNSTENV

FNCLEX Сброс флагов исключений без проверки ожидающих

FNINIT Инициализация FPU без проверки ожидающих исключений

FNOP Пустая операция FPU

FNSAVE Сохранение состояния сопроцессора и стека регистров в памяти без проверки ожидающих исключений

FNSTCW Сохранение управляющего слова без проверки ожидающих исключений

FNSTENV Сохранение состояния сопроцессора (SR, CR, TAGW, FIP и FDP) в памяти без проверки ожидающих исключений

FNSTSW Запись слова состояния без проверки ожидающих исключений

FRSTOR Загрузка состояния сопроцессора и регистров из памяти

FSAVE Сохранение состояния сопроцессора и стека регистров в памяти с предварительной проверкой ожидающих исключений

FSTCW Сохранение управляющего слова с предварительной проверкой ожидающих исключений

FSTENV Сохранение состояния сопроцессора (SR, CR, TAGW, FIP и FDP) в памяти с предварительной проверкой ожидающих исключений

FSTSW Запись слова состояния для последующего переноса кода завершения в регистр флагов с предварительной проверкой ожидающих исключений

WAIT/FWAIT Синхронизация — останов CPU до завершения текущей операции FPU, проверка ожидающих исключений FPU

Инструкции ММХ появились в процессорах Pentium ММХ и с тех пор поддерживаются всеми более современными процессорами (Pentium Pro, появившийся раньше, эти инструкции не поддерживает). Они имеют сложную мнемонику, которая включает следующие элементы:

• префикс Р (Packed), указывающий на обработку упакованных форматов;

• мнемонику операции (например, ADD, CMP или XOR);

• суффикс, идентифицирующий тип насыщения: US (Unsigned Saturation) — насыщение беззнаковое, S (Signed saturation) — насыщение знаковое;

• суффикс, идентифицирующий тип данных: В — упакованные байты, W — упакованные слова, D — упакованные двойные слова, Q -учетверенное слово.

Инструкции, у которых типы входных и выходных данных различаются (например, преобразования), имеют два суффикса.

Для инструкций пересылки данных операнды источника и назначения могут находиться в памяти (m32 или m64), целочисленных регистрах (ir32) или регистрах ММХ (mm). Для остальных инструкций, кроме вышеперечисленных, операнд-источник может быть и непосредственным, а операнд назначения всегда является регистром ММХ. Для операндов, находящихся в памяти, применимы все существующие режимы адресации.

Таблица. Инструкции ММХ

Инструкция ОписаниеEMMS Очистка стека регистров — установка всех единиц в слове тегов

Пересылка данных

MOVD Пересылка данных в младшие 32 бита регистра ММХ (с заполнением старших бит нулями) или из младших 32 бит регистра ММХ

MOVQ Пересылка данных (64 бит) из/в регистр ММХ

Преобразование форматовPACKSSDW Упаковка со знаковым насыщением четырех двойных слов в четыре

слова

PACKSSWB Упаковка со знаковым насыщением восьми слов в восемь байт

PACKUSWB Упаковка с насыщением восьми знаковых слов в восемь беззнаковых байт

PUNPCKHBW Чередование в регистре назначения байт старшей половины операнда-источника с байтами старшей половины операнда назначения

PUNPCKHWD Чередование в регистре назначения слов старшей половины операнда-источника со словами старшей половины операнда назначения

PUNPCKHDQ Чередование в регистре назначения двойного слова старшей половины операнда-источника с двойным словом старшей половины операнда назначения

PUNPCKLBW Чередование в регистре назначения байт младшей половины операнда-источника с байтами младшей половины операнда назначения

PUNPCKLWD Чередование в регистре назначения слов младшей половины операнда-источника со словами младшей половины операнда назначения

PUNPCKLDQ Чередование в регистре назначения двойного слова младшей половины операнда-источника с двойным словом младшей половины операнда назначения

Упакованная арифметика

PADDB Сложение упакованных байт (слов или двойных слов) без насыщения

PADDW (с циклическим переполнением)

PADDD

PADDSB Сложение знаковых упакованных байт (слов) с насыщением

PADDSWPADDUSB Сложение упакованных беззнаковых байт (слов) с насыщением PADDUSW

PMADDWD Умножение четырех знаковых слов операнда-источника на четыре знаков слова операнда назначения. Два двойных слова результатов умножения младших слов суммируются и записываются в младшее двойное слово операнда назначения. Два двойных слова результатов умножения старших слов суммируются и записываются в старшее двойное слово операнда назначения

PMULHW Умножение упакованных знаковых слов с сохранением только старших 16 элементов результата

PMULLW Умножение упакованных знаковых или беззнаковых слов с сохранением только младших 16 бит элементов результата

PSUBB Вычитание упакованных байт (слов или двойных слов) без

PSUBW насыщения (с циклическим антипереполнением)

PSUBD

PSUBSB Вычитание упакованных знаковых байт (слов) с насыщением PSUBSW

PSUBUSB Вычитание упакованных беззнаковых байт (слов) с насыщением PSUBUSW

ЛогикаPAND Логическое ИPANDN Логическое И mm/m64 и инверсного значения mm

POR Логическое ИЛИ

PXOR Исключающее ИЛИ

СравнениеPCMPEQB Сравнение (на равенство) упакованных байт (слов, двойных

слов). Все биты элемента результата будут единичными (True)

PCMPEQD совпадении соответствующих элементов (байт, слов или двойных

PCMPEQW слов) операндов и нулевыми (False) при несовпадении

PCMPGTB Сравнение (по величине) упакованных знаковых байт (слов, двойных слов).

PCMPGTD, PCMPGTW Все биты элемента результата будут единичными (True), если соответствующий элемент операнда назначения больше элемента операнда-источника, и нулевыми (False) в противном случае

Сдвиги и вращения

PSLLD, PSLLQ, PSLLW Логический сдвиг влево упакованных слов (двойных, учетверенных) операнда назначения на количество бит, указанных в операнде-источнике, с заполнением младших бит нулями

PSRAD, PSRAW Арифметический сдвиг вправо упакованных двойных (учетверенных) знаковых слов операнда назначения на количество бит, указанных в операнде-источнике, с заполнением младших бит битами знаковых разрядов

PSRLD, PSRLQ, PSRLW Логический сдвиг вправо упакованных слов (двойных, учетверенных) операнда назначения на количество бит, указанных в операнде- источнике, с заполнением старших бит нулями

Инструкции SSE появились в процессорах Pentium 3. Они делятся на три основные группы: инструкции над числами в блоке ХММ, дополнительные целочисленные SIMD-инструкции (в блоке ММХ) и новые инструкции кэширования. Основное число новых инструкций предназначено для работы с блоком ХММ. Векторные инструкции выполняются сразу над четырьмя парами чисел. Скалярные инструкции выполняются только над числами, расположенными в младших 32 битах операндов. Операнд-источник для инструкций ХММ может быть как регистром ХММ, так и 128-битной ячейкой памяти. Для многих инструкций требуется, чтобы операнд в памяти был выровнен по границе параграфа. При обработке скалярными инструкциями операнда в памяти пересылка между памятью и регистрами ХММ производится для всего 128-битного слова, хотя используется только 32 бита.

Таблица. Инструкции расширения SSE

Инструкция Описание

Пересылка данных с участием регистров ХММMOVAPS Пересылка 128-битных данных между памятью и регистрами ХММ или

между регистрами ХММ. Данные в памяти должны быть выровнены по границе 16-байтного параграфа

MOVUPS Пересылка 128-битных данных между памятью и регистрами ХММ или между регистрами ХММ (без требования выравнивания)

MOVHPS Пересылка 64-битных данных между памятью и старшей половиной регистров ХММ или между регистрами ХММ (младшая половина ХММ не изменяется)

MOVHLPS Пересылка старшей половины источника в младшую половину назначения (старшая половина регистра назначения не меняется)

MOVLHPS Пересылка младшей половины источника в старшую половину назначения (младшая половина регистра назначения не меняется)

MOVLPS Пересылка 64-битных данных между памятью и младшей половиной регистров ХММ или между регистрами ХММ (старшая половина ХММ не изменяется)

MOVMSKPS Сборка старших бит упакованных операндов из регистра ХММ в регистр общего назначения (биты 31, 63, 95 и 127 регистра ХММ попадают в биты О, 1, 2 и 3 регистра-приемника, остальные биты приемника будут нулевыми)

MOVSS Пересылка скалярного операнда (младшие 32 бита) между памятью и регистрами ХММ или между регистрами ХММ

Арифметические инструкции над числами в FP-формате в регистрах ХММADDPS Векторное сложениеSUBPS Векторное вычитаниеADDSS Скалярное сложение

SUBSS Скалярное вычитание

MULPS Векторное умножение

MULSS Скалярное умножение

DIVPS Векторное деление

DIVSS Скалярное деление

SQRTPS Векторное извлечение квадратного корня

SQRTSS Скалярное извлечение квадратного корня

MAXPS Векторное нахождение максимума

MAXSS Скалярное нахождение максимума

MINPS Векторное нахождение минимума

MINSS Скалярное нахождение минимума

Сравнение

CMPPS Векторное сравнение (задается полный набор 12 условий, как в инструкциях условных переходов). В том элементе операнда назначения, для которого условие сравнения выполняется, устанавливаются все единицы (32 бита), где не выполняется — все нули

CMPSS Скалярное сравнение (12 условий), аналогично предыдущему, но только для младших 32 бит

COMISS Скалярное сравнение с установкой бит ZF, PF и CF регистра EFLAGS (биты 0F, SF и AF обнуляются)

UCOMISS Скалярное сравнение, но без генерации исключения в случае NaN (при этом ZF=PF=CF=1)

Инструкции преобразований

CVTPI2PS Преобразование двух знаковых целых из регистра ММХ или 64-битной ячейки памяти в два младших РР-числа в регистре ХММ (старшая пара не изменяется). При необходимости выполняется округление

CVTSI2SS Преобразование знакового целого из 32-битного регистра или 64-битной ячейки памяти в младшее упакованное FP-число в регистре ХММ (старшие три числа не изменяются). При необходимости выполняется округление

CVTPS2PI Преобразование двух младших FP-чисел из регистра ХММ или памяти в пару целых знаковых в регистре ММХ или 64-битной ячейки памяти. При необходимости выполняется округление; если результат не умещается, возвращается значение бесконечности (80000000h)

CVTTPS2PI Преобразование, аналогичное CVTPS2PI, но при невозможности точного преобразования выполняется усечение

CVTSS2SI Преобразование младшего FP-числа из регистра ХММ в целое знаковое в 32-битном регистре. При необходимости выполняется округление; если результат не умещается, возвращается значение бесконечности (80000000h)

CVTTSS2SI Преобразование, аналогичное CVTSS2SI, но при невозможности точного преобразования выполняется усечение

Логические инструкции в блоке ХММ

ANDPS Логическое И двух пакетов операндов

ANDNPS Логическое И-НЕ двух пакетов операндов

ORPS Логическое ИЛИ двух пакетов операндов

XORPS Исключающее ИЛИ двух пакетов операндов Перестановки операндов в ХММ

SHUFPS Перестановка слов в регистре ХММ под управлением 8-битного непосредственного операнда

UNPCKHPS Переупаковка старших половин с чередованием слов в результате

UNPCKLPS Переупаковка старших половин с чередованием слов в результате

Управление состояниемLDMXCSR Загрузка регистра MXCSR

STMXCSR Сохранение регистра MXCSR

FXSAVE Сохранение состояния блоков FP/MMX и ХММ

FXRSTOR Восстановление состояния блоков FP/MMX и ХММ

Дополнительные целочисленные SIMD-инструкции (выполняются с операндами в регистрах ММХ, входят и в расширенный набор 3DNow!)

PAVGB/PAVGW Нахождение среднего упакованных беззнаковых байт или слов

PEXTRW Извлечение 16-битного слова из регистра ММХ в младшую половину 32-битного регистра (старшая половина обнуляется). Номер слова определяется младшими битами непосредственного операнда

PINSRW Помещение младшей половины 32-битного регистра в выбранное слово регистра ММХ. Номер слова определяется младшими битами непосредственного операнда

PMAXUB, PMAXSW Нахождение максимума упакованных беззнаковых байт/знаковых слов

PMINUB, PMINSW Нахождение минимума упакованных беззнаковых байт/знаковых слов

PMOVMSKB Сборка старших бит упакованных байт в 8-битную маску, помещаемую в целочисленный регистр

PMULHUW Умножение беззнаковых слов с сохранением старших половин произведений

PSADBW Нахождение суммы модулей разности пар слов (результат — 16-битное число)

PSHUFW Перемешивание слов под управлением 8-битного непосредственного операнда

Управление кэшированием (входят и в расширенный набор 3DNow!)

MASKMOVQ Выборочная запись байт из регистра ММХ в память, минуя кэш MOVNTQ Запись из регистра ММХ в память, минуя кэш

MOVNTPS Запись из регистра ХММ в память, минуя кэш (адрес должен быть

выровнен по границе параграфа)

PREFETCHTO Загрузка 32 или более байт в кэш-память

PREFETCHT1

PREFETCHT2

PREFETCHNT

SFENCE Выгрузка результатов всех предыдущих инструкций в кэш-память

Новые инструкции управления кэшированием обеспечивают запись содержимого регистров ММХ и ХММ в память, минуя кэш, что позволяет избегать "загрязнения" кэш-памяти промежуточными данными. Появилась и возможность "закачивать" требуемые данные в кэш прежде использующих их инструкций.

По сравнению с расширением 3DNow! набор инструкций SSE шире, часть инструкций пересекается, но и в 3DNow! имеются уникальные инструкции, не реализованные в SSE.

Инструкции SSE2 появились в процессорах Pentium 4. Большая их часть предназначена для работы с числами с плавающей точкой двойной точности (64-битные операнды), расположенными в регистрах ХММ, векторными (упакованная пара 64-битных чисел) и скалярными (старшим или младшим числом). Они обеспечивают векторные и скалярные пересылки этих чисел, арифметические инструкции (сложение, вычитание, умножение, деление, извлечение корня, нахождение максимума и минимума), сравнение чисел, преобразования форматов, перестановки операндов, а также побитные логические функции. Появились и SIMD-инструкции обработки 32- и 64-битных целых чисел, расположенных в регистрах ХММ. Новые инструкции управления кэшированием позволяют миновать кэш при записи в память из регистров ХММ и общих регистров, упорядочивать последовательности загрузки данных из памяти и записи в память и выполнять некоторые другие действия.

Таблица. Инструкции SSE2

Инструкция Описание

Инструкции пересылки данных (чисел с плавающей точкой двойной точности между регистрами ХММ, а также регистрами ХММ и памятью)

MOVAPD Пересылка пары упакованных выровненных чисел

MOVUPD Пересылка пары упакованных невыровненных чисел

MOVHPD Пересылка старшего упакованного числа

MOVLPD Пересылка младшего упакованного числа

MOVMSKPD Извлечение знаковой маски из пары чисел

MOVSD Пересылка скалярного числа

Арифметические инструкции над операндами с плавающей точкой двойной точности в регистрах ХММ

ADDPD Векторное сложение

ADDSD Скалярное сложение

SUBPD Векторное вычитание

SUBSD Скалярное вычитание

MULPD Векторное умножение

MULSD Скалярное умножение

DIVPD Векторное деление

DIVSD Скалярное деление

SQRTPD Векторное извлечение квадратного корня

SQRTSD Скалярное извлечение квадратного корня

MAXPD Векторное нахождение максимума

MAXSD Скалярное нахождение максимума

MINPD Векторное нахождение минимума

MINSD Скалярное нахождение минимума

Логические инструкции над упакованными 64-битными операндами в регистрах ХММ (побитные функции)

ANDPD Логическое И

ANDNPD Логическое И-НЕ

ORPD Логическое ИЛИ

XORPD Исключающее ИЛИ

Инструкции сравнения упакованных (векторных) и скалярных операндов с плавающей точкой двойной точности в регистрах ХММ с помещением результата в операнд-приемник или регистр EFLAGS

CMPPD Сравнение векторное

CMPSD Сравнение скалярное

COMISD Упорядоченное сравнение скалярных чисел с помещением результата в биты регистра EFLAGS (если хоть один из операндов QNaN или SNaN, генерируется исключение #I и EFLAGS не модифицируется)

UCOMISD Неупорядоченное сравнение (то же, но исключение #I генерируется только в случае SNaN)

Инструкции перестановок и распаковки операндов с плавающей точкой двойной точности в регистрах ХММ

SHUFPD Перестановка элементов в упакованных операндах

UNPCKHPD Распаковка и чередование старших элементов (в приемнике собираются старшие части операндов)

UNPCKLPD Распаковка и чередование младших элементов (в приемнике собираются младшие части операндов)

Инструкции преобразований в формат и из формата упакованных и скалярных чисел с плавающей точкой двойной точности

CVTPD2PI Преобразование упакованных чисел с плавающей точкой в упакованные целые (двойные слова)

CVTTPD2PI Преобразование с усечением упакованных чисел с плавающей точкой двойной точности в упакованные целые (двойные слова)

CVTP12PD Преобразование упакованных целых (двойных слов) в упакованные числа с плавающей точкой двойной точности

CVTPD2DQ Преобразование упакованных чисел с плавающей точкой в упакованные целые (двойные слова)

CVTTPD2DQ Преобразование с усечением упакованных чисел с плавающей точкой двойной точности в упакованные целые (двойные слова)

CVTDQ2PD Преобразование упакованных 32-битных целых в упакованные числа с плавающей точкой двойной точности

CVTPS2PD Преобразование упакованных чисел с плавающей точкой одинарной точности в числа двойной точности

CVTPD2PS Преобразование упакованных чисел с плавающей точкой двойной точности в числа одинарной точности

CVTSS2SD Преобразование скалярного числа с плавающей точкой одинарной точности в число двойной точности

CVTSD2SS Преобразование скалярного числа с плавающей точкой двойной точности в число одинарной точности

CVTSD2SI Преобразование скалярного числа одинарной точности в 32-битное целое

CVTTSD2SI Преобразование с усечением скалярного числа двойной точности в 32-битное целое

CVTS12SD Преобразование 32-битного целого в число двойной точности

Инструкции преобразований с числами одинарной точности

CVTDQ2PS Преобразование упакованных 32-битных целых в упакованные числа с плавающей точкой одинарной точности

CVTPS2DQ Преобразование упакованных чисел одинарной точности в числа двойной точности

CVTTPS2DQ Преобразование с усечением упакованных чисел одинарной точности в числа двойной точности

Целочисленные 128-битные SIMD-инструкции

MOVDQA Пересылка выровненного 128-битного операнда

MOVDQU Пересылка невыровненного 128-битного операнда

MOVQ2DQ Пересылка 64-битного целого из ММХ в ХММ

MOVDQ2Q Пересылка 64-битного целого из ХММ в ММХ

PMULUDQ Умножение упакованных беззнаковых 32-битных целых

PADDQ Сложение упакованных 64-битных целых

PSUBQ Вычитание упакованных 64-битных целых

PSHUFLW Перестановка упакованных младших слов

PSHUFHW Перестановка упакованных старших слов

PSHUFD Перестановка упакованных двойных слов

PSLLDQ Логический сдвиг 64-битных чисел влево

PSRLDQ Логический сдвиг 64-битных чисел вправо

PUNPCKHQDQ Распаковка старших 64-битных чисел

PUNPCKLQDQ Распаковка младших 64-битных чисел Управление кэшированием

CLFLUSH Очистка и инвалидация строки кэша (всех уровней), связанной с указанным операндом в памяти

LFENCE Упорядочивание операций загрузки из памяти

MFENCE Упорядочивание операций загрузки и записи

PAUSE Улучшение выполнения цикла ожидания

MASKMOVDQU Выборочная запись байтов из ХММ в память, минуя кэш

MOVNTPD Запись пары упакованных чисел из ХММ в память, минуя кэш

MOVNTDQ Запись 128-битного числа из ХММ в память, минуя кэш

MOVNTI Запись двойного слова из регистра общего назначения в память, минуя кэш

Инструкции 3DNow!, появившиеся с процессорами AMD K6-2, поддерживаются всеми последующими процессорами AMD и некоторыми другими процессорами.

Процессоры Intel этот набор не поддерживают, хотя в SSE имеются инструкции, совпадающие с частью инструкций 3DNow!. В процессорах Athlon расширение 3DNow! получило дополнительные инструкции для сигнальных процессоров. Целочисленные инструкции ММХ и управления кэшированием совпадают с одноименными инструкциями SSE. В данной работе инструкции 3DNow! не рассмотрены, так как не относятся к инструкциям процессоров Pentium 3,4.

1. Команды блока XMM (SSE и SSE2)

Порядок описания команд в этом разделе следующий:

• в заголовок вынесена схема команды, поясняющая общий набор и назначение операндов;

• в следующей строке дается название команды, расшифровка ее мнемоники и назначение;

• далее следует синтаксис команды (сложный синтаксис приводится в виде диаграмм), при описании которого используются следующие обозначения:

• r8, r16, r32 — операнд в одном из регистров размером байт, слово или двойное слово;

• m8, m16, m32, m48, m64 — операнд в памяти размером байт, слово, двойное слово или 48 бит;

• i8, i16, i32 — непосредственный операнд размером байт, слово или двойное слово;

• машинный код для всех сочетаний операндов описываемой команды (при сложном синтаксисе машинный код включается в синтаксис);

• состояние флагов после выполнения команды;

• описание действия команды;

• описание флагов после выполнения команды, при этом приводятся сведения только о флагах, изменяемых командой, и используются следующие обозначения:

• 1 — флаг устанавливается (равен 1);

• 0 — флаг сбрасывается (равен 0);

• r — значение флага зависит от результата выполнения команды;

• ? — после выполнения команды флаг не определен;

• список исключений.

На многих диаграммах в целях компактности возможные сочетания операндов показаны в виде следующей конструкции:

Конструируя команду на основе подобной синтаксической диаграммы, нужно помнить о соответствии типов. Допустимы только следующие сочетания: "r8, m8", "r16, m16", "r32, m32", а сочетание, например, "r8, m16" недопустимо. Однако, есть единичные случаи, когда подобные сочетания возможны; тогда они оговариваются специальным образом.

Описание машинного кода приводится в двух вариантах.

• В двоичном виде. Это описание применяется для демонстрации особенностей внутренней структуры машинной команды. Байты машинного представления машинной команды отделяются двоеточием.

• В шеснадцатеричном виде. Каждый байт машинного представления команды представлен двумя шестнадцатеричными цифрами. Часто за одним (двумя и более) первым байтом следует обозначение: /цифра. Это означает, что поле reg в байте mod r/m используется как часть кода операции и цифра представляет содержимое этого поля.

Вместо цифры может стоять символ "r" — /r. Как уже не раз отмечалось, большинство команд процессора — двухоперандные. Один операнд располагается в регистре, местоположение другого операнда определяет байт ModR/M — это может быть либо регистр, либо ячейка памяти. Более того, если операнд — ячейка памяти, то содержимое байта ModR/M определяет номенклатуру компонентов машинного кода команды, которые должны использоваться для вычисления эффективного адреса.

При описании команд могут быть опущены некоторые из перечисленных пунктов. Например, отсутствие пункта "синтаксис" говорит о том, что он совпадает со схемой команды. Отсутствие пункта "исключения" означает, что при выполнении данной команды исключения не возникают. То же касается описания флагов.

Некоторые регистры программной модели процессора имеют внутреннюю структуру. Указание того, о каком поле такого регистра идет речь, показано следующим образом: имя\_регистра.имя\_поля.

* 1. Команды блока XMM (SSE – Pentium 3)

ADDPS приемник, источник

ADDPS (ADDition Packed Single-precision float-point) — сложение упакованных значений в формате ХММ.

Синтаксис: ADDPS rxmm1, rxmm2/m128

Машинный код: 00001111:01011000:mod rxmm1 r/m

Действие: алгоритм работы команды показан на рисунке ниже.

Исключения: NE: #O, #U, #I, #Р, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,11,12,13; #XM; RM: #GP: 13; #NM: 3; #UD: 17,18; #XM; VM: исключения реального режима; #PF(fault-code).

ADDSS приемник, источник

ADDSS (ADD Scalar Single-precision float-point) — скалярное сложение значений в формате ХММ.

Синтаксис: ADDSS rxmm1, rxmm2/m128

Машинный код: 11110011:00001111:01011000:mod rxmm1 r/m

Действие: алгоритм работы команды показан на рисунке ниже.

Исключения: NE: #O, #U,. #I, #P, #D; PM: #АС: 4; #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #АС(0); #PF(fault-code).

ANDNPS приемник, источник

ANDNPS (bit-wise logical AND Not for Packed Single-precision float-point) — поразрядное логическое И-НЕ над упакованными значениями в формате ХММ.

Синтаксис: ANDNPS rxmm1, rxmm2/m128

Машинный код: 00001111:01010101:mod rxmm1 r/m

Действие: инвертировать биты операнда приемник, над каждой парой битов операндов приемник (после инвертирования) и источник выполнить операцию логического И.

Исключения: PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,12,13; RM: #GP: 13; #NM: 3; #UD: 7; VM: исключения реального режима; #PF(fault-code); #UD: 16, 17.

ANDPS приемник, источник

ANDPS (bit-wise logical AND for Packed Single-precision float-point) — поразрядное логическое И над каждой парой бит операндов источник и приемник.

Синтаксис: ANDPS rxmm1, rxmm2/m128

Машинный код: 00001111:01010100: mod rxmm 1 r/m

Исключения: 1; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,12, 13; RM: #GP: 13; #NM: 3; #UD: 17, 19, 20; VM: исключения реального режима; #PF(fault-code).

CMPPS приемник, источник, условие

CMPPS (CoMPare Packed Single-precision float-point) — сравнение упакованных значений в формате ХММ

Синтаксис: CMPPS rxmm1, rxmm2/m128, i8

Машинный код: 00001111:11000010: mod rxmm1 r/m: i8

Действие: условие, в соответствии с которым производится сравнение каждой пары элементов операндов приемник и источник, задается явно в виде непосредственного операнда (см. ниже). В результате сравнения в приемнике формируются единичные (если условие выполнено) или нулевые элементы (если условие не выполнено).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Усло-вие  | Описание условия | Отношение  | Эмуляция | Код маски i8  | Результат, если операнд NaN | Исключение #I, если операнд qNAN/sNAN |
| Eq | Equal(равно) | xmm1==xmm2 | 000b | False | Нет |
| Lt | less-than(меньшечем) | xmm1<< xmm2 | 001 b | False | Да |
| Le | less-than -or-equal(меньшечем илиравно) | xmm1<<=xmm2 | 010b | False | Да |
| greater than (большечем) | xmm1>> xmm2 | Перестанов-ка с сохране­нием, lt |  | False | Да |
| greater-than-or-equal(большечем илиравно) | xmm1>>=xmm2 | Перестанов-ка с сохране­-нием, le  |  | False | Да |
| Unord | Unordered(одно изчиселQNAN) | xmm1 ?xmm2 | 011b | True | Нет |
| Neq | not-equal(не равно) | !(xmm1==xmm2) | 100b | True | Нет |
| Nit | not-less-than (неменьшечем) | !(xmm1 <<xmm2) | 101b | True | Да |
| NIe | not-less-than-or-equal (неменьше чемили равно) | !(xmm1<<=xmm2) | 110b | True | Да |
| not-greater-than(небольшечем) | !(xmm1>>xmm2)  | Перестанов-ка с сохране­нием, nlt |  | True | Да |
| not-greater-than-or-equal (не больше чемили равно) | !(xmm1>>= xmm2)  |  Перестанов-ка с сохране­нием, nle  |  | True | Да |
| Ord | Ordered(числанеQNAN) | !(xmm1?xmm2) | 111b | False | Нет |

Исключения: 1; NE: #I, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #VD: 17-20; #XM; VM: исключения реального режима; #PF(fault-code).

CMPSS приемник, источник, условие

CMPSS (CoMPare Scalar Single-precision float-point) — скалярное сравнение значений в формате ХММ.

Синтаксис: CMPSS rxmm1, rxmm2/m32, i8

Машинный код: 11110011:00001111:11000010:mod rxmm1 r/m: i8

Действие: для пары значений операндов приемник и источник выполняется сравнение, в результате которого формируются единичные (если условие выполнено) или нулевые элементы (если условие не выполнено). Значение источника может быть расположено в 32-битной ячейке памяти или в младшем двойном слове регистра ХММ. Значение приемника расположено в младшем двойном слове другого регистра ХММ.

Возможные значения условий приведены в описании команды CMPPS.

Исключения: NE: #I, #D; PM: #AC(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #ХМ; VM: исключения реального режима; #АС(0); #PF(fault-code).

COMISS приемник, источник

COMISS (COMpare ordered Scalar Single-precision float-point COMpare and Set EFLAGS) — скалярное упорядоченное сравнение значений в формате ХММ с установкой EFLAGS.

Синтаксис: COMISS rxmm1, rxmm2/m32

Машинный код: 00001111:00101111 :mod rxmm1 r/m

Действие: команда сравнивает пару значений операндов приемник и источник, в результате чего устанавливаются флаги в регистре EFLAGS, как показано ниже.

Значение источника может быть расположено в 32-битной ячейке памяти или младшем двойном слове регистра ХММ. Значение приемника расположено в младшем двойном слове другого регистра ХММ.

|  |  |
| --- | --- |
| Соотношение операндов | Значение флагов |
| Приемник>источник | 0F=SF=AF=ZF=PF=CF=0 |
| Приемник<источник | 0F=SF=AF=ZF=PF=0; CF=1 |
| Приемник=источник | 0F=SF=AF=PF=CF=0; ZF=1 |
| Приемник или источник=qNaN или sNaN | 0F=SF=AF=0; ZF=PF=CF=1 |

При возникновении незамаскированных исключений значение EFLAGS не изменяется. Исключения: NE: #I, #D; PM: #AC(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #ХМ; VM: исключения реального режима; #АС(0); #PF(fault-code).

CVTPI2PS приемник, источник

CVTPI2PS (Conversion Two Packed signed Int32 to Packed Single-precision float-point) — преобразование двух упакованных 32-битных целых в два упакованных вещественных значения.

Синтаксис: CVTP12PS rxmm1, rmmx2/m64

Машинный код: 00001111:00101010:mod rxmm1 r/m

Действие: алгоритм работы команды показан на рисунке ниже.

В случае, когда не удается выполнить точное преобразование, результат округляется в соответствии с полем MXCSR.RC.

Исключения: NE: #Р; РМ: #АС(0); #GP(0): 37; #MF; #NM: 3; #PF(fault-code);

#SS(0): 13; #UD: 10-13; #XM; RM: #AC; #GP: 13; #MF; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #АС(0); #PF(fault-code).

CVTPS2PI приемник, источник

CVTPS2PI (ConVersion Two Packed Single-precision float-point to Packed signed Int32) — преобразование двух вещественных целых в два упакованных 32-битных целых.

Синтаксис: CVTPS2PI rmmx1, rmmx2/m128

Машинный код: 00001111:00101101 :mod rmmx1 r/m

Действие: алгоритм работы команды показан на рисунке ниже.

Если преобразованный результат больше, чем максимально возможное целочисленное 32-битное значение, то возвращается значение 80000000h. В случае, когда не удается выполнить точное преобразование, значение округляется в соответствии с полем MXCSR.RC.

Исключения: NE: #I, #Р; РМ: #АС(0); #GP(0): 37; #MF; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #MF; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #АС(0); #PF(fault-code).

CVTSI2SS приемник, источник

CVTSI2SS (ConVerT Scalar signed Int32 to Scalar Single-precision float-point) -скалярное преобразование знакового 32-битного целого в вещественное значение.

Синтаксис: CVTS12SS rxmm, r32/m32

Машинный код: 11110011:00001111:00101010:mod rxmm r/m

Действие: алгоритм работы команды показан на рисунке ниже.

В случае когда не удается выполнить точное преобразование, значение округляется в соответствии с полем MXCSR.RC.

Исключения: NE: #Р; РМ: #АС(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #ХМ; VM: исключения реального режима; #АС(0); #PF(fault-code).

CVTSS2SI приемник, источник

CVTSS2SI (ConVerT Scalar Single-precision float-point to Scalar signed Int32) -скалярное преобразование вещественного целого в 32-битное знаковое целое.

Синтаксис: CVTSS2SI r32,rxmm/m128

Машинный код: 11110011:00001111:00101101 :mod r32 r/m

Действие: значение источника хранится в младшем двойном слове регистра ХММ или в 128-битной ячейке памяти. Приемник — один из 32-битных регистров.

Алгоритм работы команды показан на рисунке ниже.

Если преобразованный результат больше, чем максимально возможное целочисленное 32-битное значение, то возвращается значение 80000000h. В случае, когда не удается выполнить точное преобразование, значение округляется в соответствии с полем MXCSR.RC.

Исключения: NE: #I, #Р; РМ: #АС(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #АС(0); #PF(fault-code).

CVTTPS2PI приемник, источник

CVTTPS2PI (ConVerT Truncate two Packed Single-precision float-point to Packed signed Int32) — преобразование (путем отбрасывания дробной части) двух вещественных целых в два упакованных 32-битных целых значения.

Синтаксис: CVTTPS2PI rmmx,rxmm/m128

Машинный код: 00001111:00101100:mod rmmx r/m

Действие: алгоритм работы команды показан на рисунке ниже.

Если преобразованный результат больше, чем максимально возможное целочисленное 32-битное значение, то будет возвращено значение 80000000h.

Исключения: NE: ffl,#P; РМ: #АС(0); #GP(0): 37; #MF; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #MF; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #АС(0); #PF(fault-code).

CVTTSS2SI приемник, источник

CVTTSS2SI (ConVerT Truncate Scalar Single-precision float-point to Scalar signed Int32) — скалярное преобразование (путем отбрасывания дробной части) вещественного целого в знаковое целое.

Синтаксис: CVTTSS2SI r32,rxmm/m128

Машинный код: 11110011:00001111:00101100:mod r32 r/m

Действие: значение источника хранится в младшем двойном слове регистра ХММ или в 128-битной ячейке памяти. Приемник — один из 32-битных регистров. Алгоритм работы команды показан на рисунке ниже.

Если преобразованный результат больше, чем максимально возможное целочисленное 32-битное значение, то будет возвращено значение 80000000h.

Исключения: NE: #I,#Р; РМ: #АС(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #ХМ; VM: исключения реального режима; #АС(0); #PF(fault-code).

DIVPS приемник, источник

DIVPS (DIVide Packed Single-precision float-point) — деление упакованных значений в формате ХММ согласно следующей схеме.

Синтаксис: DIVPS rxmm1,rxmm2/m128

Машинный код: 00001111:01011110:mod rxmm r/m

Исключения: 1; NE: #O, #U, #I, #Z, #P, #D; РМ: #GP(0): 2,37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #UD: 17-20; #XM; VM: исключения реального режима; #PF(fault-code).

DIVSS приемник, источник

DIVSS (DIVide Scalar Single-precision float-point) — скалярное деление значений в формате ХММ согласно следующей схеме.

Синтаксис: DIVSS rxmm1,rxmm2/m32

Машинный код: 11110011:00001111:01011110:mod rxmm1 r/m

Исключения: 1; NE: #O, #U, #I, #Z, #P, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #АС(0); #PF(fault-code).

FXRSTOR источник

FXRSTOR (ReSTORe Fp and mmX state and streaming simd extension state) — восстановление без проверки наличия незамаскированных исключений с плавающей точкой состояния сопроцессора, целочисленного и потокового MMX-расширений из 512-байтной области памяти (см. рисунок ниже).

Синтаксис: FXRSTOR m512

Машинный код: 00001111:10101110:mod 001 m512

Исключения: 2; PM: #AC: 4; #GP(0): 38; #NM: 3; #SS(0): 13; #UD: 10; RM: #GP: 13; #NM: 3; #UD: 17; VM: исключения реального режима; #АС: 5; #PF(fault-code).

FXSAVE приемник

FXSAVE (SAVE Fp and mmX state and streaming simd extension state) — сохранение состояния сопроцессора, целочисленного и потокового MMX-расширений в 512-байтной области памяти (см. рисунок в описании команды FXRSTOR).

Синтаксис: FXSAVE m512

Машинный код:00001111:10101110:тос1000 m512

Исключения: 2; NE: #I, #Р; PM: #AC(0); #GP(0): 37; #NM: 3; #SS(0): 13; #UD: 10; RM: #GP: 13; #NM: 3; #UD: 17; VM: исключения реального режима; #АС(0); #PF(fault-code).

LDMXCSR источник

LDMXCSR (LoaD streaming simd extension control/status register MXCSR) — загрузка регистра состояния/управления mxcsr из 32-битной ячейки памяти.

Синтаксис: LDMXCSR m32

Машинный код: 00001111:10101110:mod 010 m32

Замечание: по умолчанию регистр MXCSR загружается значением 1f80h.

Исключения: 1; PM: #AC(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,12,13; RM: #GP: 13; #NM: 3; #UD: 17,19,20; VM: исключения реального режима; #АС(0); #PF(fault-code).

MAXPS приемник, источник

MAXPS (return MAXimum Packed Single-precision float-point) – возврат максимального из упакованных значений в формате ХММ.

Синтаксис: MAXPS rxmm1, rxmm2/m128

Машинный код: 00001111:01011111 :mod rxmm1 r/m

Действие: команда извлекает максимальные значения в каждой из четырех пар

вещественных чисел в коротком формате. При этом происходит сравнение значений

соответствующих элементов источника и приемника, по результатам которого

выполняются действия:

• если элемент приемника или элемент источника является сигнализирующим не-числом — sNAN, то в элемент приемника помещается значение источника;

• иначе, если элемент источника больше элемента приемника, то в элемент приемника помещается элемент источника.

В остальных случаях значения источника и приемника не изменяются.

Исключения: 1,3; NE: #I, ftD; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; ftXM; RM: #GP: 13; #NM: 3; #UD: 17-20; UXM; VM: исключения реального режима; #PF(fault-code).

MAXSS приемник, источник

MAXSS (return MAXimum Scalar Single-precision float-point) — скалярный возврат максимального значения в формате ХММ.

Синтаксис: MAXSS rxmm1,rxmm2/m32

Машинный код: 11110011:00001111:01011111:mod rxmm1 r/m

Действие: команда извлекает максимальное из двух вещественных чисел в коротком формате. При этом происходит сравнение значений младшей пары элементов источника и приемника, по результатам которого выполняются действия, аналогичные рассмотренным в описании команды MAXPS. Старшие три элемента источника и приемника не изменяются. Исключения: 1, 3; NE: #I, #D; PM: #AC(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #VD: 17-20; #XM; VM: исключения реального режима; #АС(0); #PF(fault-code).

MINPS приемник, источник

MINPS (return MINimum Packed Single-precision float-point) — возврат минимального упакованного значения в формате ХММ.

Синтаксис: MINPS rxmm1,rxmm2/m128

Машинный код: 00001111:01011101:mod rxmm1 r/m

Действие: команда извлекает минимальные значения в каждой из четырех пар вещественных чисел в коротком формате. При этом происходит сравнение значений соответствующих элементов источника и приемника, по результатам которого выполняются действия, аналогичные рассмотренным в описании команды MAXPS.

Исключения: 1; NE: #I, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #PF(fault-code).

MINSS приемник, источник

MINSS (return MINimum Scalar Single-precision float-point) — скалярный возврат минимального значения в формате ХММ.

Синтаксис: MINSS rxmm1,rxmm2/m32

Машинный код: 11110011:00001111:01011101:mod rxmm1 r/m

Действие: команда извлекает минимальное из двух вещественных чисел в коротком формате. При этом происходит сравнение значений младшей пары элементов источника и приемника, по результатам которого выполняются действия, аналогичные рассмотренным в описании команды MAXPS. Старшие три элемента источника и приемника не изменяются.

Исключения: NE: #I, #D; PM: #AC(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #ХМ; VM: исключения реального режима; #АС(0); #PF(fault-code).

MOVAPS приемник, источник

MOVAPS (MOVe Aligned four Packed Single-precision float-point) — перемещение выровненных 128 бит источника в соответствующие биты приемника.

Исключения: 1; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; RM: #GP: 13; #NM: 3; #UD: 17, 19, 20; VM: исключения реального режима;

#PF(fault-code).

MOVHLPS приемник, источник

MOVHLPS (MOVe High to Low Packed Single-precision float-point) — копирование содержимого старшей половины регистра ХММ (источника) в младшую половину другого регистра ХММ (приемника).

Синтаксис: MOVHLPS rxmm1, rxmm2

Машинный код: 00001111:00010010:11 rxmm1,rxmm2

Исключения: PM: #NM: 3; #UD: 10, 12, 13; RM: #NM: 3; #UD: 17, 19, 20; VM: исключения реального режима.

MOVHPS приемник, источник

MOVHPS (MOVe High Packed Single-precision float-point) — перемещение верхних упакованных значений в формате ХММ из источника в приемник.

Действие:

• если источник — 64-битный операнд в памяти, то команда MOVHPS перемещает его содержимое в старшую половину приемника, представляющего собой регистр ХММ;

• если источник — регистр ХММ, то команда MOVHPS перемещает содержимое его старшей половины в приемник, который представляет собой 64-битный операнд в памяти.

Исключения: РМ: #АС(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD:10,12,13; RM: #GP: 13; #NM: 3; #UD: 17,19,20; VM: исключения реального режима; #AC(0); #PF(fault-code).

MOVLHPS приемник, источник

MOVLHPS (MOVe Low to High Packed Single-precision float-point) — перемещение нижних упакованных значений в формате ХММ в верхние.

Синтаксис: MOVLHPS rxmm1, rxmm2

Машинный код: 00001111:00010110:11 rxmm1,rxmm2

Действие: команда копирует содержимое младшей половины регистра ХММ (источника) в старшую половину другого регистра ХММ (приемника). После операции изменяется только содержимое старшей половины приемника.

Исключения: РМ: #NM: 3; #UD: 10, 12, 13; RM: #NM: 3; #UD: 17, 19, 20; VM: исключения реального режима.

MOVLPS приемник, источник

MOVLPS (MOVe Unaligned Low Packed Single-precision float-point) — перемещение невыровненных нижних упакованных значений в формате ХММ.

Действие: команда копирует содержимое младшей половины регистра ХММ в 64-битную ячейку памяти или из нее:

• если источник — 64-битный операнд в памяти, то его содержимое перемещается в младшую половину приемника, представляющего собой регистр ХММ;

• если источник — регистр ХММ, то содержимое его младшей половины перемещается в приемник, который представляет собой 64-битный операнд в памяти.

Исключения: РМ: #АС(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD:10,12,13; RM: #GP: 13; #NM: 3; ttVD: 17,19,20; VM: исключения реального режима; #AC(0); ftPF(fault-code).

MOVMSKPS приемник, источник

MOVMSKPS (MOVe sign MaSK Packed Single-precision float-point to integer) -перемещение знаковой маски в целочисленный регистр.

Синтаксис: MOVMSKPS r32, rxmm

Машинный код: 00001111:01010000:11 r32 rxmm

Действие: команда формирует маску из знаковых разрядов четырех чисел с плавающей точкой в коротком формате, упакованных в регистр ХММ (источник). После операции содержимое всего 32-битного регистра (приемника) изменяется следующим образом: в его младшую тетраду заносится знаковая маска, остальные разряды регистра обнуляются.

Исключения: РМ: #MF; #NM: 3; #UD: 10,12,13; RM: #NM: 3; #UD: 17,19,20; VM:

исключения реального режима.

MOVNTPS приемник, источник

MOVNTPS (MOVe Non Temporal aligned four Packed Single float-point) — запись в память 128 бит из регистра ХММ, минуя кэш.

Синтаксис: MOVNTPS m128,rxmm

Машинный код: 00001111:00101011:mod rxmm r/m

Исключения: 1; РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,12, 13; RM: #GP: 13; #NM: 3; #UD: 17, 19, 20; VM: исключения реального режима; #PF(fault-code).

MOVSS приемник, источник

MOVSS (MOVe Scalar Single-precision float-point) — перемещение скалярных значений в формате ХММ.

Действие: команда копирует младшие 32 бита источника в младшие 32 бита приемника. Если используется операнд в памяти, то в команде указывается адрес, соответствующий адресу младшего байта данных в памяти. Если в качестве источника используется операнд в памяти, то эти 32 бита копируются в младшее двойное слово 128-битного приемника — регистра ХММ, остальные 96 бит этого регистра устанавливаются в 0.

Исключения: PM: #AC(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD:10,12,13; RM: #GP: 13; #NM: 3; #UD: 17,19,20; VM: исключения реального режима; #АС(0); #PF(fault-code).

MOVUPS приемник, источник

MOVUPS (MOVe Unaligned four PackedSingle-precision float-point) — перемещение невыровненных упакованных значений в формате ХММ.

Действие: переместить 128 бит источника в соответствующие биты приемника.

Исключения: PM: #АС(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10; RM: #GP: 13; #NM: 3; #UD: 17; VM: исключения реального режима; #АС(0)-#PF(fault-code).

MULPS приемник, источник

MULPS (MULtiply Packed Single-precision float-point) — умножение упакованных значений в формате ХММ.

Синтаксис: MULPS rxmmi, rxmm2/m128

Машинный код: 00001111:01011001:mod rxmm1 r/m

Действие: команда умножает четыре пары вещественных чисел в коротком формате. Схема работы команды MULPS показана на следующем рисунке.

Исключения: 1; NE: #O, #U, #I, #P, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10, 11; #XM; RM: #GP: 13; #NM: 3; #UD: 17, 18; #XM; VM: исключения реального режима; #PF(fault-code).

MULSS приемник, источник

MULSS (MULtiply Scalar Single-precision float-point) — умножение скалярных значений в формате ХММ.

Синтаксис: MULSS rxmm1,rxmm2/m32

Машинный код: 11110011:00001111:01011001:mod rxmm1 r/m

Действие: команда умножает вещественные значения в младших парах операндов в формате ХММ. Операнды источник и приемник находятся в регистре ХММ, кроме того, операнд источник может находиться в 32-битной ячейке памяти. Схема работы команды MULSS показана на следующем рисунке.

Исключения: 1; NE: #O, #U, #I, #P, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,11; #XM; RM: #GP: 13; #NM: 3; #UD: 17,18; #ХМ; VM: исключения реального режима; #АС(0); ftPF(fault-code).

ORPS приемник, источник

ORPS (bit-wise logical OR for Packed Single-precision float-point) — поразрядное логическое ИЛИ над каждой парой бит упакованных операндов источник и приемник в формате ХММ.

Синтаксис: ORPS rxmmi, rxmm2/m128

Машинный код: 00001111:01010110:mod rxmm1 r/m

Исключения: 1; РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10; RM: #GP: 13; #NM: 3; #UD: 17; VM: исключения реального режима; #PF(fault-code).

RCPPS приемник, источник

RCPPS (ReCiProcal Packed Single-precision float-point) — вычисление обратных упакованных значений в формате ХММ.

Синтаксис: RCPPS rxmm1, rxmm2/m128

Машинный код: 00001111:01010011:mod rxmm1 r/m

Действие: команда вычисляет обратные значения элементов источника по формуле 1/(элемент\_источника). Максимальная ошибка вычисления:. Схема работы команды RCPPS показана на следующем рисунке.

Исключения: 1; РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10; RM: #GP: 13; #NM: 3; #UD: 17; VM: исключения реального режима; #PF(fault-code).

RCPSS приемник, источник

RCPSS (ReCiProcal Scalar Single-precision float-point) — скалярное вычисление обратного упакованного значения в формате ХММ.

Синтаксис: RCPSS rxmm1, rxmm2/m32

Машинный код: 11110011:00001111:01010011:mod rxmm1 r/m

Действие: команда вычисляет обратное значение младшего элемента операнда источник по формуле 1/(элемент\_источника). Максимальная ошибка вычисления: .

Исключения: РМ: #АС: 5; #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10; RM: #GP: 13; #NM: 3; #UD: 17; VM: #AC: 5; исключения реального режима;

#PF(fault-code).

RSQRTPS приемник, источник

RSQRTPS (Reciprocal SQuare RooT Packed Single-precision float-point) — вычисление обратных значений квадратного корня упакованных значений в формате ХММ.

Синтаксис: RSQRTPS rxmm1, rxmm2/m128

Машинный код: 00001111:01010010:mod rxmm1 r/m

Действие: команда для каждого элемента элемент\_источника операнда источник выполняет следующие два действия: вычислить квадратный корень элемент\_ис-точника; вычислить обратную величину полученного значения корня по формуле:. Максимальная ошибка вычисления: .

Исключения: 1; РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,12,13; RM: #GP: 13; #NM: 3; #UD: 17, 19, 20; VM: исключения реального режима;

#PF(fault-code).

RSQRTSS приемник, источник

RSQRTSS (Reciprocal SQuare RooT Scalar Single-precision float-point) — скалярная аппроксимация обратных значений квадратного корня упакованных значений в формате ХММ.

Синтаксис: RSQRTSS rxmm1, rxmm2/m32

Машинный код: 11110011:00001111.01010010:mod rxmm r/m

Действие: команда для младшего элемента элемент\_источника операнда источник выполняет следующие два действия: вычисляет корень квадратный элемент\_источника; вычисляет обратную величину полученного значения корня по формуле: . Максимальная ошибка вычисления: . Старшие элементы операнда приемник не изменяются.

Исключения: РМ: #АС(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10; RM: #GP: 13; #NM: 3; #UD: 17; VM: исключения реального режима; #АС: 5;

#PF(fault-code).

SHUFPS приемник, источник, маска

SHUFPS (Shuffle Packed Single-precision float-point) — перераспределение упакованных значений в формате ХММ.

Синтаксис: SHUFPS rxmm1, rxmm2/m128,i8

Машинный код: 00001111:11000110:mod rxmm1 r/m: i8

Действие: команда перераспределяет любые два из четырех двойных слов приемника в два младших двойных слова того же приемника и любые два из четырех двойных слов источника в два старших двойных слова приемника. Если использовать один и тот же регистр ХММ в качестве источника и приемника, то можно выполнять любые перестановки в пределах одного регистра. Каждая пара бит маски определяет номер двойного слова источника или приемника, которое будет перемещено в приемник следующим образом:

• маска[1:0]

• 00 приемник[00...31] 🡨 источник[00...31]

• 01 приемник[00...31] 🡨 источник[32...63]

• 10 приемник[00...31] 🡨 источник[64...95]

• 11 приемник[00...31] 🡨 источник[96..127]

• маска[3:2]

• 00 приемник[32...63] 🡨 источник[00...31]

• 01 приемник[32...63] 🡨 источник[32...63]

• 10 приемник[32...63] 🡨 источник[64...95]

• 11 приемник[32...63] 🡨 источник[96...127]

• маска[5:4]

• 00 приемник[64...95] 🡨 источник[00...31]

• 01 приемник[64...95] 🡨 источник[32...63]

• 10 приемник[64...95] 🡨 источник[64...95]

• 11 приемник[64...95] 🡨 источник[96...127]

• маска[7:6]

• 00 приемник[96...127] 🡨 источник[00...31]

• 01 приемник[96...127] 🡨 источник[32...63]

• 10 приемник[96...127] 🡨 источник[64...95]

• 11 приемник[96...127] 🡨 источник[96...127]

Схематически работа команды SHUFPS показана на следующем рисунке.

Исключения: 1; РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,12,13; RM: #GP: 13; #NM: 3; #UD: 17, 19, 20; VM: исключения реального режима; #PF (fault-code).

SQRTPS приемник, источник

SQRTPS (SQuare RooT Packed Single-precision float-point) — корень квадратный упакованных значений в формате ХММ.

Синтаксис: SQRTPS rxmm1, rxmm2/m128

Машинный код: 00001111:01010001 :mod rxmm1 r/m

Действие: команда извлекает квадратный корень из четырех упакованных вещественных чисел в коротком формате.

Исключения: 1; NE: #I, #Р, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #PF(fault-code).

SQRTSS приемник, источник

SQRTSS (SQuare RooT Scalar Single-precision float-point) — скалярное извлечение квадратного корня.

Синтаксис: SQRTSS rxmm1, rxmm2/m32

Машинный код: 11110011:00001111:01010001:mod rxmm1 r/m

Действие: команда извлекает квадратный корень из младшего двойного слова операнда источник, который должен представлять собой упакованное вещественное число в формате ХММ.

Исключения: NE: #I, #P, #D; PM: #AC(0); #GP(Q): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #АС(0); #PF(fault-code).

STMXCSR приемник

STMXCSR (STore Streaming SIMD Extension Control/Status register MXCSR) -сохранение регистра управления/состояния MXCSR в 32-битной ячейке памяти.

Синтаксис: STMXCSR m32

Машинный код: 00001111:10101110:mod 011 m32

Исключения: PM: #AC(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,12,13; RM: #GP: 13; #NM: 3; #UD: 17,19,20; VM: исключения реального режима; #AC(0); #PF(fault-code).

SUBPS приемник, источник

SUBPS (SUBtract Packed Single-precision float-point) — вычитание упакованных значений в формате ХММ.

Синтаксис: SUBPS rxmm1, rxmm2/m128

Машинный код: 00001111:01011100:mod rxmm1 r/m

Исключения: 1; NE: #O, #U, #I, #P, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD:10-13; #XM; RM: #GP: 13; #NM: 3; ttVD: 17-20; #XM; VM:

исключения реального режима; #PF(fault-code).

SUBSS приемник, источник

SUBSS (SUBtract Scalar Single-precision float-point) — вычитание скалярных значений в формате ХММ.

Синтаксис: SUBSS rxmm1, rxmm2/m32

Машинный код: 11110011:00001111:01011100:mod rxmm r/m

Действие: команда вычитает значения младшей пары вещественных чисел в формате ХММ.

Исключения: NE: #O, #U, #I, #P, #D; PM: #AC(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #АС(0); #PF(fault-code).

UCOMISS приемник, источник

UCOMISS (Unordered Scalar Single-fp COMpare and set eflags) — неупорядоченное скалярное сравнение значений в формате ХММ с установкой флагов в EFLAGS.

Синтаксис: UCOMISS rxmm1, rxmm2/m32

Машинный код: 00001111:00101110:mod rxmm1 r/m

Действие: сравнение пары вещественных элементов в коротком формате, расположенных в младшем двойном слове операндов в формате ХММ. В результате выполнения команды формируются значения флагов ZF, PF и CF, а флаги 0F, SF и AF устанавливаются в 0 (см. ниже). В процессе работы команда распознает специальные значения qNaN и sNaN. При возникновении незамаскированных исключений расширения ХММ регистр EFLAGS не изменяется.

|  |  |
| --- | --- |
| Соотношение операндов | Значение флагов |
| Приемник>источник | 0F=SF=AF=ZF=PF=CF=0 |
| Приемник<источник | 0F=SF=AF=ZF=PF=0; CF=1 |
| Приемник=источник | 0F=SF=AF=PF=CF=0; ZF=1 |
| Приемник или источник=qNaN или sNaN | 0F=SF=AF=0; ZF=PF=CF=1 |

Исключения: NE: #I, #D; PM: #AC(0); #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-13; #XM; RM: #GP: 13; #NM: 3; #UD: 17-20; #XM; VM: исключения реального режима; #АС(0); #PF(fault-code).

UNPCKHPS приемник, источник

 UNPCKHPS (Unpack High Packed Single-precision float-point data) — чередование верхних упакованных значений в формате ХММ.

Синтаксис: UNPCKHPS rxmm1, rxmm2/m128

Машинный код: 00001111:00010101:mod rxmm1 r/m

Действие: перемещение путем чередования двух старших двойных слова приемника и источника. Младшие двойные слова приемника и источника игнорируются. Схема выполнения команды UNPCKHPS показана на следующем рисунке.

Исключения: 1; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,12,13; RM: #GP: 13; #NM: 3; #UD: 17, 19, 20; VM: исключения реального режима;

#PF(fault-code).

UNPCKLPS приемник, источник

UNPCKLPS (Unpack Low Packed Single-precision float-point data) — чередование нижних упакованных значений в формате ХММ.

Синтаксис: UNPCKLPS rxmm1, rxmm2/m128

Машинный код: 00001111:00010100:mod rxmm1 r/m

Действие: перемещение путем чередования двух младших двойных слова приемника и источника. Старшие двойные слова приемника и источника игнорируются. Схема работы команды UNPCKLPS показана на следующем рисунке.

Исключения: 1; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,12, 13; RM: #GP: 13; #NM: 3; #UD: 17, 19, 20; VM: исключения реального режима;#PF(fault-code).

XORPS приемник, источник

XORPS (bit-wise logical XOR for Packed Single-precision float-point) — поразрядное логическое ИСКЛЮЧАЮЩЕЕ ИЛИ над упакованными значениями в формате ХММ.

Синтаксис: XORPS rxmm1, rxmm2/m128

Машинный код: 00001111:01010111 :mod rxmm1 r/m

Исключения: PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,12,13; RM: #GP: 13; #NM: 3; #UD: 17, 19, 20; VM: исключения реального режима;

#PF(fault-code).

6.2. Команды блока XMM (SSE2 – Pentium 4)

ADDPD приемник, источник

ADDPD (ADD Packed Double-precision floating-point values) — сложение упакованных значений с плавающей точкой двойной точности.

Синтаксис: ADDPD rxmm1, rxmm2/m128

Машинный код: 66 0F 58 /r

Действие: сложить пары упакованных значении с плавающей точкой двойной точности источника и приемника (аналогично команде ADDPS) и сохранить результат сложения в соответствующих упакованных значениях с плавающей точкой двойной точности приемника.

Исключения: SIMD (NE): #O, #U, #I, #Р, #D; PM: #GP(0): 37, 42; #NM: 3; #ХМ; #PF(fault-code); #SS(0): 13; #UD: 10-12,15; RM: #GP: 13,16; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #PF(fault-code).

ADDSD приемник, источник

ADDSD (ADD Scalar Double-precision floating-point values) — сложение скалярных упакованных значений с плавающей точкой двойной точности.

Синтаксис: ADDSD rxmm1, rxmm2/m64

Машинный код: F2 0F 58 /r

Действие: сложить младшие упакованные значения с плавающей точкой двойной точности источника и приемника (аналогично команде ADDSS) и сохранить результат сложения в младшем упакованном значении с плавающей точкой двойной точности приемника.

Исключения: SIMD (NE): #O, #U, #I, #Р, #D; PM: #GP(0): 37; #PF(fault-code); #SS(0): 13; #NM: 3; #XM; #UD: 10, 11,12, 15; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17, 18, 19, 22; #XM; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

ANDPD приемник, источник

ANDPD (bitwise logical AND 0F Packed Double-precision floating-point values) — поразрядное логическое И над упакованными значениями с плавающей точкой двойной точности.

Синтаксис: ANDPD xmm1, xmm2/m128

Машинный код: 66 0F 54 /r

Действие: выполнить поразрядное логические И над двумя упакованными значениями с плавающей точкой двойной точности по схеме: приемник[127-0]🡨 приемник[127-0] пoбитнoe\_AND источник[127-0].

Исключения: PM: #GP(0): 37, 42; #SS(0): 13; #NM: 3; #XM; #PF(fault-code);

#UD: 10-12,15; RM: #GP: 13,16; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #PF(fault-code).

ANDNPD приемник, источник

ANDNPD (bitwise logical AND NOT 0F Packed Double-precision floating-point values) — поразрядное логическое И-НЕ над упакованными значениями с плавающей точкой двойной точности. Синтаксис: ANDNPD xmm1, xmm2/m128

Машинный код: 66 0F 55 /r

Действие: выполнить операцию поразрядного логического И-НЕ над парами упакованных значений с плавающей точкой двойной точности в приемнике и источнике по схеме: приемник[127-0]🡨((NОТприемник[127-0]) пoбитнoe\_AND источник[127-0]).

Исключения: PM: #GP(0): 37, 42; #SS(0): 13; #NM: 3; #PF(fault-code); #UD: 10, 11,12, 15; #XM; RM: #GP: 13, 16; #NM: 3; #UD: 17, 18,19, 22; #XM; VM: исключения реального режима; #PF(fault-code).

CLFLUSH адрес\_байта

CLFLUSH (FLUSH Cache Line) — сброс на диск строки кэша, содержащей адрес\_байта.

Синтаксис: CLFLUSH m8

Машинный код: 0F AE /7

Действие: объявить недействительной строку кэша, которая содержит линейный адрес адрес\_байта на всех уровнях иерархии кэшей данных и команд процессора. Если на одном из уровней иерархии кэшей строка "грязная" (противоречит содержимому памяти), то перед объявлением ее недействительной она записывается в память.

Возможность использования команды CLFLUSH на данном процессоре необходимо выяснить с помощью CPUID. Выровненный размер строки кэша, на который воздействует CLFLUSH, также определяется командой CPUID.

Исключения: PM: #GP(0): 37; #PF(fault-code); #SS(0): 13; #UD: 16; RM: #GP: 13; #UD: 23; VM: исключения реального режима; #PF(fault-code).

CMPPD приемник, источник, условие

CMPPD (CoMPare Packed Double-precision floating-point values description) — сравнение упакованных значений с плавающей точкой двойной точности.

Синтаксис: CMPPD xmm1, xmm2/m128, imm8

Машинный код: 66 0F C2 /r i8

Действие: сравнить упакованные значения с плавающей точкой двойной точности в приемнике и источнике. Результат сравнения для каждой пары упакованных чисел представляется в виде маски: единичная маска ffffffffffffffffh — значения чисел равны, нулевая маска 0000000000000000h — значения не равны. Условие сравнения задается непосредственным операндом условие, первые 3 бита которого определяют тип сравнения. Остальные биты зарезервированы. Соответствие значений операнда условие условию сравнения следующее: 0 (приемник = источник), 1 (приемник < источник), 2 (приемник <= источник), 3 (приемник и (или) источник — NAN или в неопределенном формате), 4 (приемник источник), 5 ((приемник <источник)), 6 ( (приемник <= источник)), 7 (упакованные значения приемника и источника — правильные значения с плавающей точкой двойной точности). Для проверки остальных условий необходимо вначале поменять содержимое приемника и источника, а затем использовать команду CMPPD со следующими значениями операнда условие: 1 (приемник > источник), 2 (приемник >= источник), 5 ( (приемник > источник)), 6 ( (приемник >= источник)).

Исключения: SIMD (NE): #I (если операнд SNaN или QNaN), #D; PM: #GP(0):

37, 42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; RM: #GP: 13, 16; #NM: 3; #UD: 17, 19, 22; #XM; VM: исключения реального режима; #PF(fault-code).

CMPSD приемник, источник, условие

CMPSD (CoMPare Scalar Double-precision floating-point values description) — сравнение скалярных значений с плавающей точкой двойной точности.

Синтаксис: CMPSD xmm1, xmm2/m64, imm8

Машинный код: F2 0F C2 /r i8

Действие: сравнить упакованные значения с плавающей точкой двойной точности в разрядах [63-0] приемника и источника. Формирование проверяемого условия и результата выполнения команды аналогичны соответствующим атрибутам команды CMPPD.

Исключения: SIMD (NE): #I (если операнд SNaN или QNaN), #D; PM: #GP(0): 37; #SS(0): 13; #PF(fault-code); #NM: 3; #XM; #UD: 10-12, 15; #AC(0)\_cpl3; RM: #GP: 13, 16; #NM: 3; #XM; #UD: 17-19, 22; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

COMISD приемник, источник, условие

COMISD (COMpare Scalar ordered Double-precision floating-point values and set EFLAGS) — сравнение упорядоченных скалярных значений с плавающей точкой двойной точности и установка регистра EFLAGS.

Синтаксис: COMISD xmm1, xmm2/m64

Машинный код: 66 0F 2F /r

Действие: сравнить упорядоченные скалярные значения с плавающей точкой двойной точности в разрядах [63-0] приемника и источника. По результату сравнения установить флаги ZF, PF и CF в регистре EFLAGS: приемник > источник (ZF = О, PF = О, CF = 0), приемник > источник (ZF = О, PF = О, CF = 1), приемник = источник (ZF = 1, PF = О, CF = 0), приемник и(или) источник NAN или в неопределенном формате (ZF = 1, PF = 1, CF = 1). Флаги 0F, SF и AF устанавливаются в 0. В случае генерации немаскированного исключения с плавающей точкой регистр EFLAGS не модифицируется.

Исключения: SIMD (NE): #I (если операнд SNaN или QNaN), #D; PM: #GP(0): 37; #SS(0): 13; #PF(fault-code); #NM: 3; #XM; #UD: 10-12, 15; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #XM; #UD: 17-19, 22; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

CVTDQ2PD приемник, источник

CVTDQ2PD (ConVerT packed Doubleword Integers to Packed Double-precision floating-point values) — преобразование двух упакованных 32-битных целых в два упакованных значения с плавающей точкой двойной точности.

Синтаксис: CVTDQ2PD rxmm1, rxmm2/m64

Машинный код: F3 0F E6

Действие: алгоритм работы команды показан на рисунке ниже.

Исключения: PM: #GP(0): 37; #SS(0): 13; #PF(fault-code); #NM: 3; #XM; #UD:

10, 11, 12, 15; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #АС: 5; #PF(fault-code); #AC(0)\_u.

CVTDQ2PS приемник, источник

CVTDQ2PS (ConVerT Packed Doubleword integers to Packed Single-precision floatingpoint values) — преобразование четырех упакованных 32-битных целых со знаком в четыре упакованных значения с плавающей точкой одинарной точности.

Синтаксис: CVTDQ2PS rxmm1, rxmm2/m128

Машинный код: 0F 5B /r

Действие: алгоритм работы команды показан на рисунке ниже.

В случае когда не удается выполнить точное преобразование, значение округляется в соответствии с полем MXCSR.RC.

Исключения: SIMD (NE): #P; PM: #GP(0): 37, 42; #SS(0): 13; #PF(fault-code); #NM: 3; #XM; #UD: 10-12, 15; RM: #GP: 13, 16; #NM: 3; #XM; #UD: 17-19, 22; VM: исключения реального режима; #АС: 5; #PF(fault-code).

CVTPD2DQ приемник, источник

CVTPD2DQ, (ConVerT Packed Double-Precision Floating-Point Values to Packed Doubleword integers) — преобразование двух упакованных значений с плавающей точкой двойной точности в два упакованных 32-битных целых.

Синтаксис: CVTPD2DQ rxmm1, rxmm2/m128

Машинный код: F2 0F E6

Действие: алгоритм работы команды показан на рисунке ниже.

В случае, когда не удается выполнить точное преобразование, значение округляется в соответствии с полем MXCSR.RC. Если преобразованный результат больше чем максимально возможное целочисленное 32-битное значение, то возвращается значение 80000000h.

Исключения: SIMD (NE): #I, #Р; РМ: #GP(0): 37, 42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; RM: #GP: 13, 16; #NM: 3; #UD: 17, 18, 19, 22; #XM; VM: исключения реального режима; #PF(fault-code).

CVTPD2PI приемник, источник

CVTPD2PI (ConVerT Packed Double-precision floating-point values to Packed doubleword Integers) — преобразование двух упакованных значений с плавающей точкой двойной точности в два упакованных 32-битных целых.

Синтаксис: CVTPD2PI rmmx, rxmm/m128

Машинный код: 66 0F 2D /r

Действие: алгоритм работы команды показан на рисунке ниже.

В случае, когда не удается выполнить точное преобразование, значение округляется в соответствии с полем MXCSR.RC. Если преобразованный результат больше чем максимально возможное целочисленное 32-битное значение, то возвращается значение 80000000h.

Исключения: SIMD (NE): #I, #Р; РМ: #GP(0): 37,42; #SS(0): 13; #PF(fault-code); #MF; #NM: 3; #XM; #UD: 10, 11, 12, 15; RM: #GP: 13, 16; #NM: 3; #MF; #XM;

#UD: 17-19, 22; VM: исключения реального режима; #PF(fault-code).

CVTPD2PS приемник, источник

CVTPD2PS (CoVerT Packed Double-precision floating-point values to Packed Single-precision floating-point values) — преобразование двух упакованных значений с плавающей точкой двойной точности в два упакованных значения с плавающей точкой одинарной точности.

Синтаксис: CVTPD2PS rxmm1, rxmm2/m128

Машинный код: 66 0F 5A /r

Действие: алгоритм работы команды показан на рисунке ниже.

В случае, когда не удается выполнить точное преобразование, значение округляется в соответствии с полем MXCSR.RC.

Исключения: SIMD (NE): #O, #U, #I, #Р, #D; PM: #GP(0): 37, 42; #SS(0): 13;

#PF(fault-code); #NM: 3; #XM; #UD: 10-12, 15; RM: #GP: 13, 16; #NM: 3; #ХМ; #UD: 17-19, 22; VM: исключения реального режима; #PF(fault-code).

CVTP12PD приемник, источник

CVTPI2PD (ConVerT Packed doubleword Integers to Packed Double-precision floating-point values) — преобразование двух упакованных 32-битных целых в два упакованных значения с плавающей точкой двойной точности.

Синтаксис: CVTP12PD rxmm, rmmx/m64

Машинный код: 66 0F 2A /r

Действие: алгоритм работы команды показан на рисунке ниже.

Исключения: PM: #GP(0): 37; #SS(0): 13; #PF(fault-code); #NM: 3; #MF; #XM;

#UD: 10-12,15; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #MF; #XM; #UD: 17-19, 22; VM: исключения реального режима; #АС; #PF(fault-code); #AC(0)\_u.

CVTPS2DQ приемник, источник

CVTPS2DQ (ConVerT Packed Single-precision floating-point values to packed Doubleword integers) — преобразование четырех упакованных значений с плавающей точкой одинарной точности в четыре упакованных 32-битных целых со знаком.

Синтаксис: CVTPS2DQ rxmm1, rxmm2/m128

Машинный код: 66 0F 5В /r

Действие: алгоритм работы команды показан на рисунке ниже.

В случае когда не удается выполнить точное преобразование, значение округляется в соответствии с полем MXCSR.RC. Если преобразованный результат больше чем максимально возможное целочисленное 32-битное значение, то возвращается значение 80000000h.

Исключения: SIMD (NE): #I, #P; PM: #GP(0): 37,42; #SS(0): 13; #PF(fault-code); #MF; #NM: 3; #XM; #UD: 10, 11, 12, 15; RM: #GP: 13, 16; #NM: 3; #MF; #XM;

#UD: 17-19, 22; VM: исключения реального режима; #PF(fault-code).

CVTPS2PD приемник, источник

CVTPS2PD (CoVerT Packed Single-precision floating-point values to Packed Double-precision floating-point values) — преобразование двух упакованных значений с плавающей точкой одинарной точности в два упакованных значения с плавающей точкой двойной точности.

Синтаксис: CVTPS2PD rxmm1, rxmm2/m64

Машинный код: 0F 5A /r

Действие: алгоритм работы команды показан на рисунке ниже.

Исключения: РМ: #GP(0): 37; #SS(0): 13; #PF(fault-code); #NM: 3; #XM; #UD:

10-12,15; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #XM; #UD: 17-19,22; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

CVTSD2SI приемник, источник

CVTSD2SI (ConVerT Scalar Double-precision floating-point value to Doubleword Integer) — преобразование скалярного значения с плавающей точкой двойной точности в 32-битное целое.

Синтаксис: CVTSD2SI r32, rxmm/m64

Машинный код: F2 0F 2D /r

Действие: алгоритм работы команды показан па рисунке ниже.

В случае, когда не удается выполнить точное преобразование, значение округляется в соответствии с полем MXCSR.RC. Если преобразованный результат больше чем максимально возможное целочисленное 32-битное значение, то возвращается значение 80000000h.

Исключения: SIMD (NE): #I, #Р; РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #ХМ; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

CVTSD2SS приемник, источник

CVTSD2SS (ConVerT Scalar Double-precision floating-point value to Scalar Single-precision floating-point value) — преобразование скалярного значения с плавающей точкой двойной точности в скалярное значение с плавающей точкой одинарной точности.

Синтаксис: CVTSD2SS rxmm1, rxmm2/m64

Машинный код: F2 0F 5A /r

Действие: алгоритм работы команды показан на рисунке ниже.

В случае, когда не удается выполнить точное преобразование, значение округляется в соответствии с полем MXCSR.RC.

Исключения: SIMD (NE): #O, #U, #I, #Р, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #ХМ; #АС(0)\_ср13; RM: #GP: 13; #NM: 3; #XM; #UD: 17-19, 22; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

CVTS12SD приемник, источник

CVTSI2SD (ConVerT Signed doubleword Integer to Scalar Double-precision floatingpoint value) — преобразование 32-битного целого значения со знаком в упакованное значение с плавающей точкой двойной точности.

Синтаксис: CVTS12SD rxmm, r/m32

Машинный код: F2 0F 2A /r

Действие: алгоритм работы команды показан на рисунке ниже.

Исключения: РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #AC(0)\_u; #PF(fault-code).

CVTSS2SD приемник, источник

CVTSS2SD (ConVerT Scalar Single-Precision floating-point value to Scalar Double-precision floating-point value) — преобразование скалярного значения с плавающей точкой одинарной точности в скалярное значение с плавающей точкой двойной точности.

Синтаксис: CVTSS2SD rxmm1, rxmm2/m32

Машинный код: F3 0F 5A /r

Действие: алгоритм работы команды показан на рисунке ниже.

Исключения: РМ: #GP(0): 37; ftNM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #ХМ; #АС(0)\_ср13; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #AC(0)\_u; #PF(fault-code).

CVTTPD2PI приемник, источник

CVTTPD2PI (ConVerT with Truncation Packed Double-precision floating-point values to Packed doubleword Integers) — преобразование (путем отбрасывания дробной части) двух упакованных значений с плавающей точкой двойной точности в два упакованных 32-битных целых значения.

Синтаксис: CVTTPD2PI rmmx, rxmm/m128

Машинный код: 66 0F 2C /r

Действие: алгоритм работы команды показан на рисунке ниже.

Если преобразованный результат больше чем максимально возможное целочисленное 32-битное значение, то будет возвращено значение 80000000h.

Исключения: SIMD (NE): #I,#Р; РМ: #GP(0): 37, 42; #MF; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; RM: #GP: 13, 16; #MF; #NM: 3; #UD: 17-19, 22; #ХМ; VM: исключения реального режима; #PF(fault-code).

CVTTPD2DQ приемник, источник

CVTTPD2DQ (ConVerT with Truncation Packed Double-precision floating-point values to packed Doubleword integers) — преобразование усечением двух упакованных значений с плавающей точкой двойной точности в два упакованных 32-битных целых.

Синтаксис: CVTTPD2DQ rxmm1, rxmm2/m128

Машинный код: 66 0F E6

Действие: алгоритм работы команды показан на рисунке ниже.

В случае, когда не удается выполнить точное преобразование, значение округляется в сторону нуля. Если преобразованный результат больше чем максимально возможнее целочисленное 32-битное значение, то возвращается значение 80000000h.

Исключения: SIMD (NE): #I, #Р; РМ: #GP(0): 37, 42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12,15; #XM; RM: #GP: 13,16; #NM: 3; #UD: 17-19, 22; #ХМ; VM: исключения реального режима; #PF(fault-code).

CVTTPS2DQ приемник, источник

CVTTPS2DQ (ConVerT with Truncation Packed Single-precision floating-point values to packed Doubleword integers) — преобразование (путем отбрасывания дробной части) четырех упакованных значений с плавающей точкой одинарной точности в четыре упакованных 32-битных целых со знаком.

Синтаксис: CVTTPS2DQ rxmm1, rxmm2/m128

Машинный код: F3 0F 5B /r

Действие: алгоритм работы команды показан на рисунке ниже.

В случае, когда не удается выполнить точное преобразование, значение округляется в сторону нуля. Если преобразованный результат больше чем максимально возможнее целочисленное 32-бнтное значение, то будет возвращено значение 80000000h.

Исключения: SIMD (NE): #I, #Р; РМ: #GP(0): 37, 42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; RM: #GP: 13, 16; #NM: 3; #UD: 17-19, 22; #ХМ; VM1; исключения реального режима; #PF(fault-code).

CVTTSD2SI приемник, источник

CVTTSD2SI (ConVerT with Truncation Scalar Double-precision floating-point value to Signed doubleword Integer) — преобразование (путем отбрасывания дробной части) скалярного значения с плавающей точкой двойной точности в 32-битное целое.

Синтаксис: CVTTSD2SI r32, rxmm/m64

Машинный код: F2 0F 2C /r

Действие: алгоритм работы команды показан на рисунке ниже.

В случае, когда не удается выполнить точное преобразование, значение округляется в сторону нуля. Если преобразованный результат больше чем максимально возможное целочисленное 32-битное значение, то будет возвращено значение 80000000h.

Исключения: SIMD (NE): #I,#Р; РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #ХМ; VM: исключения реального режима; #AC(0)\_u; #PF(fault-code).

DIVPD приемник, источник

DIVPD (DIVide Packed Double-precision floating-point values) — деление упакованных значений с плавающей точкой двойной точности.

Синтаксис: DIVPD xmm1, xmm2/m128

Машинный код: 66 0F 5E /r

Действие: разделить пары упакованных значений с плавающей точкой двойнор точности источника и приемника по схеме: приемник[63-0]🡨приемник[63-0]/источник[63-0]; приемник[127-64]🡨приемник[127-64]/источник[127-64].

Исключения: SIMD (NE): #O, #U, #I, #Z, #P, #D; PM: #GP(0): 37,42; #SS(0): 13; #PF(fault-code); #NM: 3; #UD: 10-12, 15; #XM; RM: #GP: 13, 16; #NM: 3; #VD: 17-19, 22; #XM; VM: исключения реального режима; #PF(fault-code).

DIVSD приемник, источник

DIVSD (DIVide Scalar Double-Precision Floating-Point Values) — деление скалярных упакованных значений с плавающей точкой двойной точности.

Синтаксис: DIVSD rxmm1, rxmm2/m64

Машинный код: F2 0F 5E /r

Действие: разделить младшие упакованные значения с плавающей точкой двойной точности источника и приемника по схеме: приемник[63-0]🡨приемник[63-0]/ источник[63-0]; приемник[127-64] — не изменяется.

Исключения: SIMD (NE): #O, #U, #Z, #I, #P, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #ХМ; VM: исключения реального режима; #PF(fault-code), AC(0)\_u.

LFENCE адрес\_байта

LFENCE (Load FENCE) — упорядочить операции загрузки.

Синтаксис: LFENCE

Машинный код: 0F AE /5

Действие: выполнить упорядочивание исполнения команд загрузки из памяти, которые были инициированы перед этой командой LFENCE. Эта операция гарантирует, что каждая команда загрузки, за которой следует в программе команда LFENCE, глобально видима перед любой другой командой загрузки, за которой следует команда LFENCE. Команда LFENCE упорядочивается относительно команд загрузки, других команд LFENCE, MFENCE и любых команд упорядочивания (сериализации, типа команды CPUID). Она не упорядочивается относительно команд сохранения в памяти или команды SFENCE.

Исключения: отсутствуют.

MASKMOVDQU источник, маска

MASKMOVDQU (Store Selected Bytes 0F Double Quadword) — выборочная запись байт из источника в память с использованием байтовой маски в приемнике.

Синтаксис: 66 0F F7 /r

Машинный код: MASKMOVDQU rxmm1, rxmm2

Действие: сохранить выбранные байты операнда источник в 128-разрядную ячейку памяти. Операнд маска определяет байты источника, которые сохраняются в памяти. Местоположение первого байта ячейки памяти приемника, в которую сохраняются байты, определяются парой DS:DI/EDI. Старший значащий бит каждого байта операнда маска определяет, будет ли сохранен в приемнике соответствующий байт источника: 0 — байт не сохраняется; 1 — байт сохраняется. Команда MASKMOVEDQU генерирует указание процессору не использовать кэш. Это указание реализуется посредством метода кэширования WC (Write Combining — память с комбинированной записью). При этом операции упорядочивания, осуществляемые командами SFENCE или MFENCE, необходимо использовать совместно с командами MASKMOVEDQU. Для многопроцессорной конфигурации это особенно важно, так как различным процессорам могут требоваться различные типы памяти для чтения/записи ячейки приемника.

Исключения: PM: #GP(0): 37 (в том числе при нулевой маске); #NM: 3; #PF(fault-code); #SS(0): 13 (в том числе при нулевой маске); #UD: 10, 12, 15; RM: #GP: 13 (в том числе при нулевой маске); #NM: 3; #UD: 17,19, 22; VM: исключения реального режима; #PF(fault-code).

MAXPD приемник, источник

MAXPD (return MAXimum Packed Double-precision floating-point values) — возврат максимальных упакованных значений с плавающей точкой двойной точности.

Синтаксис: MAXPD rxmm1, rxmm2/m128

Машинный код: 66 0F 5F /r

Действие: сравнить упакованные значения с плавающей точкой двойной точности в источнике и приемнике и заместить максимальными из них соответствующие упакованные значения в приемнике. Если значение в источнике — SNAN (не QNAN), то оно помещается в приемник. Если только одно значение в приемнике или источнике — не число NaN (SNaN или QNAN), то в приемник помещается содержимое источника, которое может быть либо NAN, либо правильным значением числа с плавающей точкой.

Исключения: SIMD (NE): #I (в том числе, если источник = QNaN), #D; PM: #GP(0): 37, 42; #SS(0): 1;#NM: 3; #PF(fault-code); #UD: 10-12, 15; #XM; RM: #GP: 13, 16; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #PF(fault-code).

MAXSD приемник, источник

MAXSD (return MAXimum Scalar Double-precision floating-point value) — возврат максимального скалярного значения с плавающей точкой двойной точности.

Синтаксис: MAXSD rxmm1, rxmm2/m64

Машинный код: F2 0F 5F /r

Действие: сравнить значения с плавающей точкой двойной точности в разрядах [63-0] источника и приемника и заместить максимальным из них значение в разрядах [63-0] приемника. Если значение в источнике — SNAN (не QNAN), то оно помещается в приемник. Если только одно значение в приемнике или источнике — не число NaN (SNaN или QNAN), то в приемник помещается содержимое источника, которое может быть либо NAN, либо правильным значением числа с плавающей точкой. Значение в разрядах [127-64] приемника не изменяется.

Исключения: SIMD (NE): #I (в том числе, если источник = QNaN), #D; PM: #GP(0): 37; #SS(0): 1; #NM: 3; #PF(fault-code); #UD: 10-12, 15; #ХМ; #АС(0)\_срl3; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

MFENCE

MFENCE (Memory FENCE) — упорядочить операции загрузки и сохранения. Синтаксис: MFENCE

Машинный код: 0F AE /6

Действие: выполнить упорядочивание команд загрузки из памяти и сохранения в памяти, которые были инициированы перед этой командой MFENCE. Эта операция гарантирует, что каждая команда загрузки и сохранения, за которой следует в программе команда MFENCE, глобально видима перед любой другой командой загрузки и сохранения, за которой следует команда MFENCE. Команда MFENCE упорядочивается относительно команд загрузки и сохранения, других команд LFENCE, MFENCE, SFENCE и любых команд упорядочивания (сериализации, типа команды CPUID).

Исключения: отсутствуют.

MINPD приемник, источник

MINPD (return MINimum Packed Double-precision floating-point values) — возврат минимальных упакованных значений с плавающей точкой двойной точности.

Синтаксис: MINPD xmm1, xmm2/m128

Машинный код: 66 0F 5D /r

Действие: сравнить упакованные значения с плавающей точкой двойной точности в источнике и приемнике и заместить минимальными из них соответствующие упакованные значения в приемнике. Если значение в источнике — SNAN (не QNAN), то оно помещается в приемник. Если только одно значение в приемнике или источнике — не число NaN (SNaN или QNAN), то в приемник помещается содержимое источника, которое может быть либо NAN, либо правильным значением числа с плавающей точкой.

Исключения: SIMD (NE): #I (в том числе, если источник == QNaN), #D; PM: #GP(0): 37, 42; #SS(0): 1; #NM: 3; #PF(fault-code); #UD: 10-12, 15; #XM; RM: #GP: 13, 16; #NM: 3; #UD: 17-19, 22; #ХМ; VM: исключения реального режима; #PF(fault-code).

MINSD приемник, источник

MINSD (return MINimum Scalar Double-precision floating-point value) — возврат минимального скалярного значения с плавающей точкой двойной точности.

Синтаксис: MINSD xmm1, xmm2/m64

Машинный код: F2 0F 5D /r

Действие: сравнить значения с плавающей точкой двойной точности в разрядах [63-0] источника и приемника и заместить минимальным из них значение в разрядах [63-0] приемника. Если значение в источнике — SNAN (не QNAN), то оно помещается в приемник. Если только одно значение в приемнике или источнике — не число NaN (SNaN или QNAN), то в приемник помещается содержимое источника, которое может быть либо NAN, либо правильным значением числа с плавающей точкой. Значение в разрядах [127-64] приемника не изменяется.

Исключения: SIMD (NE): #I (в том числе, если источник = QNaN), #D; PM: #GP(0): 37; #SS(0): 1; #NM: 3; #PF(fault-code); #UD: 10-12, 15; #ХМ; #АС(0)\_ср13; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

MOVAPD приемник, источник

MOVAPD (MOVe Aligned Packed Double-precision floating-point values) — перемещение упакованных выровненных значений с плавающей точкой двойной точности.

Синтаксис и машинный код:

0F 28 /r MOVAPS xmm1, xmm2/m128

0F 29 /r MOVAPS xmm2/m128, xmm1

Действие: переместить два двойных учетверенных слова (содержащих два упакованных значения с плавающей точкой двойной точности) из источника в приемник. Операнд в памяти должен быть выровнен на 16-байтовой границе.

Исключения: PM: #GP(0): 37,42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; RM: #GP: 13,16; #NM: 3; #UD: 17-19,22; #XM; VM: исключения реального режима; #PF(fault-code).

MOVD приемник, источник

MOVD (Move Double word) — перемещение двойного слова между ХММ-регист-ром и 32-разрядным регистром (ячейкой памяти).

Синтаксис и машинный код:

66 0F 6Е /r MOVD rxmm, r/m32

66 0F 7E /r MOVD r/m32, rxmm

Действие:

• Если приемник является ХММ-регистром, поместить в биты 0-31 приемника значение источника, поместить в биты 32-127 приемника нулевое значение.

• Если приемник является 32-разрядной ячейкой памяти или регистром общего назначения, то поместить в приемник значение бит 0-31 источника (ХММ-регистра).

Флаги: не изменяются.

Исключения: РМ: #GP(0): 1, 2; #MF; #PF(fault-code); #SS(0): 1; #UD: 10, 12, 15; #NM 3; #AC(0)\_cpl3; RM: #GP: 13; #MF; #NM: 3; #UD: 17,19,22; VM: #PF(fault-code); #AC(0)\_u.

MOVDQA приемник, источник

MOVDQA (MOVe Aligned Double Quadword) — перемещение выровненных 128 бит из источника в приемник.

Синтаксис и машинный код:

66 0F 6F /r MOVDQA rxmm1, rxmm2/m128

66 0F 7F /r MOVDQA rxmm2/m128, rxmml

Действие: переместить содержимое источника в приемник. Операнд в памяти должен быть выровнен на 16-байтовой границе.

Исключения: РМ: #GP(0): 2,42; #NM: 3; #SS(0): 1; RM: #GP: 13,16; #NM: 3; #UD: 17, 19, 22; #PF(fault-code); VM: исключения реального режима; #PF(fault-code).

MOVDQU приемник, источник

MOVDQU (MOVe Unaligned Double Quadword description) — перемещение невыровненных 128 бит из источника в приемник.

Синтаксис и машинный код:

F3 0F 6F /r MOVDQU xmm1, xmm2/m128

F3 0F 7F /r MOVDQU xmm2/m128, xmm1

Действие: переместить содержимое источника в приемник.

Исключения: РМ: #GP(0): 2; #NM: 3; #PF(fault-code); #SS(0): 1; #UD: 10,12,15; RM: #GP: 13; #NM: 3; #UD: 17,19,22; VM: исключения реального режима; #PF(fault-code).

MOVDQ2Q приемник, источник

MOVDQ2Q (MOVe Quadword from XMM to MMX register description) — перемещение младшего учетверенного слова ХММ-регистра в ММХ-регистр.

Синтаксис: MOVDQ2Q mm, xmm

Машинный код: F2 0F D6

Действие: переместить содержимое источника в приемник по схеме: приемник 🡨 источник[63-0].

Исключения: РМ: #NM: 3; #UD: 10, 12, 15; #MF; RM: исключения защищенного

режима; VM: исключения защищенного режима.

MOVHPD приемник, источник

MOVHPD (MOVe High Packed Double-precision floating-point value) — перемещение старшего упакованного значения с плавающей точкой двойной точности.

Синтаксис и машинный код:

66 0F 16 /r MOVHPD rxmm, m64

66 0F 17 /r MOVHPD m64, rxmm

Действие: переместить учетверенное слово (содержащее упакованное значение с плавающей точкой двойной точности) из источника в приемник. Источник и приемник могут быть либо ХММ-регистром, либо 64-разрядной ячейкой памяти (но не одновременно). Для регистрового операнда перемещению подвергается старшее учетверенное слово (разряды [64-127]). Младшее учетверенное слово ХММ-регистра (разряды [0-63]) не изменяется.

Исключения: РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10, 13, 19; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 12,13,17; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

MOVLPD приемник, источник

MOVLPD (MOVe Low Packed Double-precision floating-point value) — перемещение младшего упакованного значения с плавающей точкой двойной точности.

Синтаксис и машинный код:

66 0F 12 /r MOVLPD rxmm, m64

66 0F 13 /r MOVLPD m64, rxmm

Действие: переместить учетверенное слово (содержащее упакованное значение с плавающей точкой двойной точности) из источника в приемник. Источник и приемник могут быть либо ХММ-регистром, либо 64-разрядной ячейкой памяти (но не одновременно). Для регистрового операнда перемещению подвергается младшее учетверенное слово (разряды [0-63]). Старшее учетверенное слово ХММ-регистра (разряды [64-127]) не изменяется.

Исключения: РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10, 12, 13; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17,19,20; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

MOVMSKPD приемник, источник

MOVMSKPD (extract Packed Double-precision floating-point sign MaSK) — извлечение 2-битной знаковой маски упакованных значений с плавающей точкой двойной точности.

Синтаксис: MOVMSKPD r32, rxmm

Машинный код: 66 0F 50 /r

Действие: извлечь знаковые разряды из упакованных значении с плавающей точкой двойной точности операнда источник (ХММ-регистр) и сохранить полученную знаковую маску в двух младших битах операнда приемник (32-битный общий регистр).

Исключения: РМ: #NM: 3; #ХМ; #UD: 10-12,15; RM: исключения защищенного режима; VM: исключения защищенного режима.

MOVNTDQ приемник, источник

MOVNTDQ (store Double Quadword using Non-Temporal hint description) — сохранение двойного учетверенного слова из ХММ-регистра в память без использования кэша.

Синтаксис: 66 0F E7 /r

Машинный код: MOVNTDQ m128, rxmm

Исключения: РМ: #GP(0): 37, 42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10, 12,15; RM: #GP: 13,16; #NM: 3; #UD: 17,19,22; VM: исключения реального режима; #PF(fault-code).

MOVNTI приемник, источник

MOVNTI (store doubleword using Non-Temporal hint description) — сохранение двойного слова из 32-разрядного регистра общего назначения в память без использования кэша.

Синтаксис: MOVNTI m32, r32

Машинный код: 0F C3 /r

Исключения: РМ: #GP(0): 37; #PF(fault-code); #SS(0): 13; #UD: 15; RM: #GP: 13, 16; #NM: 3; #UD: 22; VM: исключения реального режима; #PF(fault-code).

MOVNTPD приемник, источник

MOVNTPD (store Packed Double-Precision floating-point values using Non-Temporal hint) — сохранение упакованных значений с плавающей точкой двойной точности из ХММ-регистра в память без использования кэша.

Синтаксис: MOVNTPD m128, rxmm

Машинный код: 66 0F 2В /r

Исключения: РМ: #GP(0): 37, 42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10, 12,15; RM: #GP: 13,16; #NM: 3; #UD: 17,19,22; VM: исключения реального режима; #PF(rault-code).

MOVQ приемник, источник

MOVQ (Move Quadword) — переместить учетверенное слово.

Синтаксис и машинный код:

F3 0F 7E MOVQ rxmm1, rxmm2/m64

66 0F D6 MOVQ rxmm2/m64, rxmm1

Действие:

• Если приемник и источник являются ХММ-регистрами, то изменить содержимое приемника следующим образом: приемник[63-0]🡨источник[63-0]; разряды приемник[127-64] не изменяются.

• Если приемник — 64-разрядная ячейка памяти, то изменить содержимое приемника следующим образом: приемник[63-0]🡨источник[63-0].

• Если источник — 64-разрядная ячейка памяти, то изменить содержимое приемника следующим образом: приемник[63-0]🡨источник[63-0], приемник[63-0]🡨0000000000000000h.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 1, 2; #SS(0): 1; #UD: 10,12,15; #NM: 3; #MF; #PF(fault-code); #AC(0)\_cpl3; RM: #GP: 13; #MF; #NM: 3; #UD: 17,19, 22; VM: исключения В реального режима; #PF(fault-code); #AC(0)\_u.

MOVQ2DQ приемник, источник

MOVQ2DQ (MOVe Quadword from MMX to XMM register description) — перемещение учетверенного слова из ММХ-регистра в младшее учетверенное слово ХММ-регистра.

Синтаксис: MOVQ2DQ rxmm, rmmx

Машинный код: F3 0F D6

Действие: переместить содержимое источника в приемник по схеме: приемник[63-0] 🡨 источник; приемник[127-64] 🡨 00000000000000000h.

Исключения: РМ: #NM: 3; #UD: 10, 12, 15; #MF; RM: исключения защищенного режима; VM: исключения защищенного режима.

MOVSD приемник, источник

MOVSD (MOVe Scalar Double-precision floating-point value) — перемещение скалярного значения с плавающей точкой двойной точности.

Синтаксис и машинный код:

F2 0F 10 /r MOVSD rxmm1, rxmm2/m64

F2 0F 11 /r MOVSD rxmm2/m64, rxmm1

Действие: переместить скалярное значение с плавающей точкой двойной точности из разрядов [0-63] источника в разряды [0-63] приемника. Если операнды — ХММ-регистры, то разряды [64-127] приемника не изменяются. Если источник -ячейка памяти, то разряды [64-127] приемника обнуляются.

Исключения: РМ: #GP(0): 37; #SS(0): 13; #PF(fault-code); #NM: 3; #XM; #UD:

10-12,15; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17-19,22; #ХМ; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

MOVUPD приемник, источник

MOVUPD (MOVe Unaligned Packed Double-precision floating-point values) — перемещение невыровненных упакованных значений с плавающей точкой двойной точности.

Синтаксис и машинный код:

66 0F 10 /r MOVUPD xmm1, xmm2/m128

66 0F 11 /r MOVUPD xmm2/m128, xmm1

Действие: переместить два двойных учетверенных слова (содержащих два упакованных значения с плавающей точкой двойной точности) из источника в приемник. Не требуется выравнивания операнда в памяти на 16-байтовой границе.

Исключения: РМ: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #PF(fault-code).

MULPD приемник, источник

MULPD (MULtiply Packed Double-precision floating-point values) — умножение упакованных значений с плавающей точкой двойной точности.

Синтаксис: MULPD rxmm1, rxmm2/m128

Машинный код: 66 0F 59 /r

Действие: умножить пары упакованных значений с плавающей точкой двойной точности источника и приемника по схеме: приемник[63-0]🡨приемник[63-0]источник[63-0]; приемник[127-64]🡨приемник[127-64]источник[127-64].

Исключения: SIMD (NE): #O, #U, #I, #Р, #D; PM: #GP(0): 37, 42; #SS(0): 13; #NM: 3; #PF(fault-code); #UD: 10-12, 15; #XM; RM: #GP: 13, 16; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #PF(fault-code).

MULSD приемник, источник

MULSD (MULtiply Scalar Double-precision floating-point values) — умножение скалярных упакованных значений с плавающей точкой двойной точности.

Синтаксис: MULSD rxmm1, rxmm2/m64

Машинный код: F2 0F 59 /r

Действие: умножить младшие упакованные значения с плавающей точкой двойной точности источника и приемника по схеме: приемник[63-0]🡨приемник[63-0] источник[63-0]; приемник[127-64] — не изменяется.

Исключения: SIMD (NE): #O, #U, #I, #Р, #D; PM: #GP(0): 37; #SS(0): 13; #NM: 3; #PF(fault-code); #UD: 10-12, 15; #XM; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #ХМ; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

ORPD приемник, источник

ORPD (bitwise logical OR 0F Double-precision floating-point values) — поразрядное логическое ИЛИ над упакованными значениями с плавающей точкой двойной точности.

Синтаксис: ORPD xmm1, xmm2/m128

Машинный код: 66 0F 56 /r

Действие: выполнить операцию поразрядного логического ИЛИ над парами упакованных значений с плавающей точкой двойной точности в приемнике и источнике по схеме: приемник[127-0]🡨приемник[127-0]) побитное\_ОR источник[127-0].

Исключения: PM: #GP(0): 37,42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; RM: #GP: 13,16; #NM: 3; #UD: 17-19,22; #XM; VM: исключения реального режима; #PF(fault-code).

PACKSSWB/PACKSSDW приемник, источник

PACKSSWB (Pack with Signed Saturation Words to Bytes) — упаковка со знаковым насыщением слов в байты. PACKSSDW (Pack with Signed Saturation Double Words to Words) — упаковка со знаковым насыщением двойных слов в слова.

Синтаксис и машинный код:

66 0F 63 /r PACKSSWB rxmm1, rxmm2/m128

66 0F 6В /r PACKSSDW rxmm1, rxmm2/m128

Действие: команды преобразуют шестнадцать/восемь элементов размером в слово/двойное слово из источника и приемника в шестнадцать/восемь элементов в операнде приемник размером в байт/слово (см. рисунок ниже). Если значение элемента источника превышает допустимое значения элемента приемника, то в элементе приемника формируется предельный результат в соответствии с принципом знакового насыщения:

• PACKSSWB — 07fh для положительных чисел и 080h для отрицательных;

• PACKSSDW — 07fffh для положительных чисел и 08000h для отрицательных.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PACKUSWB приемник, источник

PACKUSWB (PACK with Unsigned Saturation Description) — упаковка с беззнаковым насыщением слов в байты.

Синтаксис: PACKUSWB rxmm1, rxmm2/m128

Машинный код: 66 0F 67 /r

Действие: команда преобразует шестнадцать элементов из источника и приемника размером в слово в шестнадцать элементов в приемнике размером в байт (см. рисунок ниже). Если пересылаемое значение больше допустимого для поля приемника, то в нем формируется предельный результат в соответствии с принципом беззнакового насыщения, что соответствует значениям 0Ffh для положительных чисел и 00h для отрицательных.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17, 19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PADDB/PADDW/PADDD приемник, источник

PADDB (Packed ADDition Bytes) — сложение упакованных байт. PADDW (Packed ADDition Words) — сложение упакованных слов. PADDD (Packed ADDition Double words) — сложение упакованных двойных слов.

Синтаксис и машинный код:

66 0F FC /r PADDB rxmm1, rxmm2/m128

66 0F FD /r PADDW rxmm1, rxmm2/m128

66 0F FE /r PADDD rxmm1, rxmm2/m128

Действие: команда в зависимости от кода операции складывает соответствующие элементы операндов источника и приемника размером байт/слово/двойное слово. При возникновении переполнения результат формируется в соответствии с принципом циклического переполнения и помещается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PADDQ приемник, источник

PADDQ (ADD Packed Quadword integers description) — сложение учетверенных слов.

Синтаксис и машинный код:

0F D4 /r PADDQ rmmx1 ,rmmx2/m64

66 0F D4 /r PADDQ rxmm1,rxmm2/m128

Действие: сложить 64-битные целые значения в источнике и приемнике. Исходя из типа источника, возможны две схемы умножения:

• источник — ММХ-регистр или ячейка памяти m64: приемник[63-0]🡨прием-ммк[63-0] + источник[63-0], приемник — ММХ-регистр;

• источник — ХММ-регистр или ячейка памяти: приемник[63-0]🡨приемник[63-0] + источник[63-0]; приемник[127-64.]🡨приемник[127-64] + источник[127-64].

В результате выполнения команды PADDQ регистр EFLAGS не отражает факта возникновения ситуации переполнения или переноса. Когда результат умножения слишком большой, чтобы быть представленным в 64-битном элементе приемника, то он "заворачивается" (перенос игнорируется). Для обнаружения подобных ситуаций программное обеспечение должно использовать другие методы.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2,42; #SS(0): 1; #UD: 10,12,15; #NM: 3; #PF(fault-code); RM: #GP: 13, 16; #UD: 17, 19, 22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PADDSB/PADDSW приемник, источник

PADDSB (Packed ADDition signed Bytes with Saturation) — сложение упакованных бант со знаковым насыщением. PADDSW (Packed ADDition signed Words with Saturation) — сложение упакованных слов со знаковым насыщением.

Синтаксис и машинный код:

66 0F EC /r PADDSB rxmm1, rxmm2/m128

66 0F ED /r PADDSW rxmm1, rxmm2/m128

Действие: команда в зависимости от кода операции складывает соответствующие элементы операндов источника и приемника размером байт/слово с учетом знака. При возникновении переполнения результат формируется в соответствии с принципом знакового насыщения:

• PADDSB — 07fh для положительных чисел и 080h для отрицательных;

• PADDSW — 07fffh для положительных чисел и 08000h для отрицательных.

Результат помещается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19, 22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PADDUSB/PADDUSW приемник, источник

PADDUSB (Packed ADDition unsigned Bytes with Unsigned Saturation) — сложение упакованных байт с беззнаковым насыщением. PADDUSW (Packed ADDition unsigned Words with Unsigned Saturation) — сложение упакованных слов с беззнаковым насыщением.

Синтаксис и машинный код:

66 0F DC /r PADDUSB rxmm1, rxmm2/m128

66 0F DD /r PADDUSW rxmm1, rxmm2/m128

Действие: команда в зависимости от кода операции складывает без учета знака соответствующие элементы операндов источника и приемника размером байт/слово. При возникновении переполнения результат формируется в приемнике в соответствии с принципом беззнакового насыщения:

• PADDUSB – 0ffh и 00h для результатов сложения соответственно больших или меньших максимально/минимально представимых значений в беззнаковом байте;

• PADDUSW – 0ffffh и 0000h для результатов сложения соответственно больших или меньших максимально/минимально представимых значений в беззнаковом слове.

Результат помещается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2,42; #SS(0): 1; #UD: 10,12,15; #NM: 3; #PF(fault-code); RM: #GP: 13, 16; #UD: 17, 19, 22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PAND приемник, источник

PAND (Packed logical AND) — упакованное логическое И.

Синтаксис: PAND rxmm1, rxmm2/m128

Машинный код: 66 0F DB /r

Действие: команда выполняет побитовую операцию логическое И над всеми битами операндов источника и приемника. Результат помещается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PANDN приемник, источник

PANDN (Packed logical AND Not) — упакованное логическое И-НЕ. Синтаксис: PANDN rxmm1, rxmm2/m128

Машинный код: 66 0F DF /r

Действие: команда выполняет побитовую операцию логическое И-НЕ над всеми битами операндов источника и приемника. Результат помещается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PAUSE

PAUSE (Spin Loop Hint) — улучшить выполнение цикла ожидания-занятости. Синтаксис: PAUSE

Машинный код: F3 90

Действие: улучшить выполнение цикла ожидания-занятости (spin-wait loops). При выполнении подобных циклов процессор Pentium 4 испытывает проблему при завершении цикла, обнаруживая возможное нарушение доступа к памяти. Команда PAUSE подсказывает процессору, что данная кодовая последовательность — цикл ожидания-занятости. Процессор использует эту подсказку, чтобы игнорировать возможную ситуацию нарушения доступа к памяти в большинстве случаев. Это улучшает работу процессора вплоть до значительного снижения его энергопотребления. По этой причине рекомендуется включать команду PAUSE во все циклы ожидания-занятости.

Исключения: отсутствуют.

PAVGB/PAVGW приемник, источник

PAVGB/PAVGW (Packed Average) — упакованное среднее.

Синтаксис и машинный код:

66 0F E0 /r PAVGB rxmm1, rxmm2/m128

66 0F E3 /r PAVGW rxmm1, rxmm2/m128

Действие: выполнить параллельное сложение байт/слов источника и приемника и сдвинуть результат сложения на один разряд вправо (деление на 2).

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2,42; #SS(0): 1; #UD: 10,12,15; #NM: 3; #PF(fault-code); RM: #GP: 13, 16; #UD: 17, 19, 22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PCMPEQB/PCMPEQW/PCMPEQD приемник, источник

PCMPEQB (Packed CoMPare for Equal Byte) — сравнение на равенство упакованных байт. PCMPEQW (Packed CoMPare for Equal Word) — сравнение на равенство упакованных слов. PCMPEQD (Packed CoMPare for Equal Double word) — сравнение на равенство упакованных двойных слов.

Синтаксис и машинный код:

66 0F 74 /r PCMPEQB rxmm1, rxmm2/m128

66 0F 75 /r PCMPEQW rxmm1, rxmm2/m128

66 0F 76 /r PCMPEQD rxmm1, rxmm2/m128

Действие: команды сравнивают на равенство элементы источника и приемника и

формируют элементы результата по следующему принципу:

• если элемент источника равен соответствующему элементу приемника, то элемент результата в зависимости от применяемой команды устанавливается равным одному из следующих значений: 0ffh, 0ffffh, 0ffffffffh;

• если элемент источника не равен соответствующему элементу приемника, то элемент результата в зависимости от применяемой команды устанавливается равным одному из следующих значений: 00h, 0000h, 00000000h.

Результат помещается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PCMPGTB/PCMPGTW/PCMPGTD приемник, источник

PCMPGTB (Packed CoMPare for Greater Than Byte) — сравнение по условию "больше чем" упакованных байт. PCMPGTW (Packed CoMPare for Greater Than Word) — сравнение по условию "больше чем" упакованных слов. PCMPGTD (Packed CoMPare for Greater Than Double word) — сравнение но условию "больше чем" упакованных двойных слов.

Синтаксис и машинный код:

66 0F 64 /r PCMPGTB rxmm1, rxmm2/m128

66 0F 65 /r PCMPGTW rxmm1, rxmm2/m128

66 0F 66 /r PCMPGTD rxmm1, rxmm2/m128

Действие: команда производит сравнение по условию "больше чем" элементов операндов источника и приемника и формирует элементы результата по следующему принципу:

• если элемент приемника больше соответствующего элемента источника, то элемент результата в зависимости от применяемой команды устанавливается равным одному из следующих значений: 0ffh, 0ffffh, 0ffffffffh;

• если элемент приемника не больше соответствующего элемента источника, то элемент результата в зависимости от применяемой команды устанавливается равным одному из следующих значений: 00h, 0000h, 00000000h.

Результат помещается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PEXTRW приемник, источник, маска

PEXTRW (Extract Word) — извлечение 16-битного слова из ХММ-рстистра по маске.

Синтаксис: PEXTRW r32, rxmm, imm8

Машинный код: 66 0F C5 /r i8

Действие: команда выделяет четыре младших бита непосредственного операнда маска. Их значение определяет номер слова в операнде источник (ХММ-регистр). Данное слово перемещается в младшие 16 бит операнда приемник, представляющего собой 32-разрядный регистр общего назначения. Старшие 16 бит этого регистра обнуляются.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PINSRW приемник, источник, маска

PINSRW (Insert Word) — вставка 16-битного слова в регистр ММХ.

Синтаксис: PINSRW rxmm, r32/m16, imm8

Машинный код: 66 0F C4 /r i8

Действие: команда выделяет четыре младших бита непосредственного операнда маска. Их значение определяет номер слова в операнде приемник, который представляет собой ХММ-регистр. В это слово будут перемещены младшие 16 бит операнда источник, который представляет собой 32-разрядный регистр общего назначения или 16-битную ячейку памяти.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PMADDWD приемник, источник

PMADDWD (Packed Multiply and ADD Word to Double word) — упакованное знаковое умножение знаковых слов операндов источник и приемник с последующим сложением промежуточных результатов в формате двойного слова.

Синтаксис: PMADDWD rxmm1, rxmm2/m128

Машинный код: 66 0F F5 /r

Действие: работа команды аналогична команде блока MMX PMADDWD, за исключением того, что вместо ММХ-регистров используются ХММ-регистры и разрядность операндов в памяти повышается до 128 бит.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PMAXSW приемник, источник

PMAXSW (MAXimum 0F Packed Signed Word integers) — возврат максимальных упакованных знаковых слов.

Синтаксис: PMAXSW rxmmi, rxmm2/m128

Машинный код: 66 0F ЕЕ /r

Действие: команда определяет наибольшее слово для каждой пары упакованных слов источника и приемника с учетом знака и заменяет им соответствующие слова приемника.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PMAXUB приемник, источник

PMAXUB (MAXimum 0F Packed Unsigned Byte integers) — возврат максимальных упакованных беззнаковых байт.

Синтаксис: PMAXUB rxmmi, rxmm2/m128

Машинный код: 66 0F DE /r

Действие: для каждой пары байтовых элементов источника и приемника определить наибольший без учета знака и заменить им соответствующим элемент приемника.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19, 22; #NM: 3; VM: исключения реального режима; ftPF(fault-code).

PMINSW приемник, источник

PMINSW (MINimum of Packed Signed Word integers) — возврат минимальных упакованных знаковых слов.

Синтаксис: PMINSW rxmmi, rxmm2/m128

Машинный код: 66 0F EA /r

Действие: для каждой пары элементов (размером 16 бит) источника и приемника команда определяет наименьший с учетом знака и заменяет им соответствующий элемент приемника.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15: #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PMINUB приемник, источник

PMINUB (MINimum of Packed Unsigned Byte integers) — возврат минимальных упакованных беззнаковых бант.

Синтаксис: PMINUB rxmmi, rxmm2/m128

Машинный код: 66 0F DA /r

Действие: для каждой пары байтовых элементов источника и приемника команда определяет наименьший без учета знака и заменяет им соответствующий элемент

приемника.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(lault-code); RM: #GP: 13,16; #UD: 17,19, 22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PMOVMSKB приемник, источник

PMOVMSKB (MOVe Byte MaSK) — перемещение байтовой маски в целочисленный регистр.

Синтаксис: PMOVMSKB r32, rxmm

Машинный код: 66 0F D7 /r

Действие: команда извлекает и копирует значения старшего бита каждого из упакованных байт ХММ-регистра в младшие 16 бит 32-битного целочисленного регистра общего назначения. Остальные разряды целочисленного регистра обнуляются.

Флаги: не изменяются.

Исключения: РМ: #UD: 10, 12,15; #NM: 3; RM: исключения защищенного режима; VM: исключения защищенного режима.

PMULHUW приемник, источник

PMULHUW (MULtiply Packed Unsigned integers and store High result) — умножение упакованных беззнаковых слов с возвратом старших слов результата.

Синтаксис: PMULHUW rxmm1, rxmm2/m128

Машинный код: 66 0F E4 /r

Действие: команда производит умножение упакованных слов источника и приемника без учета знака и формирует элементы результата в соответствии с приведенной ниже схемой. Как видно из нее, в результате умножения слов операндов источник и приемник получаются промежуточные результаты размером 32 бита.

Далее старшее слово (16 бит) из каждого промежуточного результата умножения исходных элементов помещается в 16-битный элемент окончательного результата. Результат помещается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PMULHW приемник, источник

PMULHW (MULtiply Packed signed integers and store High result) — упакованное знаковое умножение слов с возвратом старшего слова результата.

Синтаксис: PMULHW rxmm1, rxmm2/m128

Машинный код: 66 0F E5 /r

Действие: команда производит умножение упакованных слов источника и приемника с учетом знака и формирует элементы результата в соответствии со схемой, приведенной при описании команды PMULHUW.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PMULLW приемник, источник

PMULLW (MULtiply Packed signed integers and store Low result) — упакованное знаковое умножение слов с возвратом младшего слова результата.

Синтаксис: PMULLW xmm 1, xmm2/m 128

Машинный код: 66 0F D5 /r

Действие: команда производит умножение с учетом знака упакованных слов источника и приемника и формирует элементы результата в соответствии с приведенной ниже схемой.

Как видно из этой схемы, в результате умножения слов источника и приемника получаются промежуточные результаты размером 32 бита. Далее младшее слово (16 бит) из каждого 32-битного элемента промежуточного результата умножения исходных элементов помещается в 16-битный элемент результата (операнд приемник). Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13, 16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PMULUDQ приемник, источник

PMULUDQ, (MULtiply Packed Unsigned Doubleword integers description) — умножение 32-битных целых значений без учета знака и сохранение результата в ХММ-регистре.

Синтаксис и машинный код:

0F F4 /r PMULUDQ rmmx1, rmmx2/m64

66 0F F4 /r PMULUDQ rxmm1, rxmm2/m128

Действие: умножить 32-битные целые значения со знаком в источнике и приемнике. Исходя из типа источника, возможны две схемы умножения:

• источник — ММХ-регистр или ячейка памяти m64: приемник[63-0] 🡨приемник[?А-0] источник[31-0];

• источник — ХММ-регистр или ячейка памяти: приемник[63-0}🡨приемник[31-0] источник[31-0]; приемник[127-64]🡨приемник[95-64]источник[95-64].

Когда результат умножения слишком большой, чтобы быть представленным в приемник, то он "заворачивается" (перенос игнорируется).

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2,42; #SS(0): 1; #UD: 10,12,15; #NM: 3; #PF(fault-code); RM: #GP: 13, 16; #UD: 17, 19, 22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

POR приемник, источник

POR (bitwise logical OR) — упакованное логическое ИЛИ.

Синтаксис: POR rxmm1, rxmm2/m128

Машинный код: 66 0F ЕВ /r

Действие: команда производит побитовую операцию логическое ИЛИ над всеми битами операндов источника и приемника. Результат помешается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19, 22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PSADBW приемник, источник

PSADBW (Compute Sum 0F Absolute Differences) — суммарная разница значений нар беззнаковых упакованных байт.

Синтаксис: PSADBW rxmm1, rxmm2/m128

Машинный код: 66 0F F6 /r

Действие: для каждой пары байт двух упакованных учетверенных слов операндов источник и приемник вычислить модуль разности, после чего сложить полученные модули. Результат записать в младшее слово каждого из двух упакованных учетверенных слов приемника, старшие три слова в упакованных учетверенных словах приемника обнулить. Принцип работы программы поясняет схема ниже.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PSHUFD приемник, источник, маска

PSHUFD (SHUFfle Packed Doublewords) — копирование двойных слов из ХММ-операнда источник в ХММ-операнд приемник.

Синтаксис: PSHUFD xmm1, xmm2/m128, imm8

Машинный код: 66 0F 70 /r i8

Действие: на основе значения пар бит маски копировать двойные слова из источника в приемник. Каждая пара бит маски определяет номер слова источника для перемещения.в приемник следующим образом:

• маска[1:0]:

• 00 приемник[0...31] 🡨 источник[0...31],

• 01 приемник[0...31] 🡨 источник[32...63],

• 10 npueMHUK[0...31] 🡨 источник[64...95];

• 11 приемник[0...31] 🡨 источник[96..127];

• маска[3:2]:

• 00 приемник[32...63] 🡨 источник[0...31];

• 01 приемник[32...63] 🡨 источник[32...63];

• 10 приемник[32...63] 🡨 источник[64...95];

• 11 приемник[32...63] 🡨 источник[96..127];

• маска[5:4]:

• 00 приемник[64...95] 🡨 источник[0...31];

• 01 приемник[64...95] 🡨 источник[32...63],

• 10 приемиик[64...95] 🡨 источник[64...95];

• 11 приемник[64...95] 🡨 источник[96..127];

• маска[7:6]:

• 00 приемник[96..127] 🡨 источник[0...31];

• 01 приемник[96..127] 🡨 источник[32...63];

• 10 приемник[96..127] 🡨 источник[64...95];

• 11 приемник[96..127] 🡨 источник[96...127].

Если использовать один и тот же ХММ-регистр в качестве источника и приемника, то можно выполнять любые перестановки двойных слов в пределах одного ХММ-регистра, в том числе и инициализацию значением одного двойного слова других двойных слов. Работу команды PSHUFD поясняет следующая схема:

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #NM: 3; #UD: 17,19, 22; VM: исключения реального режима; #PF(fault-code).

PSHUFHW приемник, источник, маска

PSHUFHW (SHUFfle Packed High Words) — копирование слов из старшего учетверенного упакованного слова ХММ-операнда источник в старшее учетверенное упакованное слово ХММ-операнда приемник.

Синтаксис: PSHUFHW xmm1, xmm2/m128, imm8

Машинный код: F3 0F 70 /r i8

Действие: на основе значения пар бит маски копировать слова из старшего упакованного учетверенного слова источника в старшее учетверенное упакованное слово приемника. Каждая пара бит маски определяет номер слова источника для перемещения в приемник следующим образом:

• маска[1:0]:

• 00 приемник[64...79] 🡨 источник[64...79];

• 01 приемник[64...79] 🡨 источник[80...95];

• 10 приемник[64...79] 🡨 источник[96..111];

• 11 приемник[64...79] 🡨 источник[112...127];

• маска[3:2]:

• 00 приемник[80...95] 🡨 источник[64...79];

• 01 приемник[80...95] 🡨 источник[80...95];

• 10 приемник[80...95] 🡨 источник[96..111];

• 11 приемник[80...95] 🡨 источник[112..127];

• маска[5:4]:

• 00 приемник [96...111] 🡨 источгшк[64...79];

• 01 приемник [96...111] 🡨 источник[80...95];

• 10 приемник [96...111] 🡨 источпик[96...111];

• 11 приемник [96...111] 🡨 источник[112..127];

• маска[7:6]:

• 00 приемник[112...127] 🡨 источник[64...79];

• 01 приемник[112...127] 🡨 источник[80...95];

• 10 приемник[112...127] 🡨 источник[96..111];

• 11 приемник[112...127} 🡨 источник[112..127].

Если использовать один и тот же ХММ-регистр в качестве источника и приемника, то можно выполнять любые перестановки слов в пределах старшего учетверенного слова одного ХММ-регистра, в том числе и инициализацию значением одного слова других слов. Работу команды PSHUFHW поясняет следующая схема.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-codc); RM: #GP: 13,16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fanlt-code).

PSHUFLW приемник, источник, маска

PSHUFLW (SHUFfle Packed Low Words) — копирование слов из младшего учетверенного упакованного слова ХММ-онераида источник в младшее учетверенное упакованное слово ХММ-онеранда приемник.

Синтаксис: PSHUFLW rxmm1, rxmm2/m128, imm8

Машинный код: F2 0F 70 /r i8

Действие: на основе значения пар бит маски копировать слова из младшего учетверенного слова источника в младшее учетверенное слово приемника. Каждая пара бит маски определяет номер слова источника для перемещения в приемник следующим образом:

• маска[1:0]:

• 00 приемник[00..15] 🡨 источник[00..15];

• 01 приемник[00..15] 🡨 источник[16..31];

• 10 приемник[00..15] 🡨 источник[32..47];

• 11 приемник[00..15] 🡨 источник[48..63];

• маска[3:2]:

• 00 приемник[16...31] 🡨 источник[00..15];

• 01 приемник[16...31] 🡨 источник[16...31];

• 10 приемник[16...31] 🡨 источгшк[32..47];

• 11 приемник[16...31] 🡨 источник[48...63];

• маска[5:4]:

• 00 приемник[32..47] 🡨 источник[00..15];

• 01 приемник[32..47] 🡨 источник[16..31];

• 10 приемник[32..47] 🡨 источник[32..47];

• 11 приемник[32..47] 🡨 источник[48..63];

• маска[7:6]:

• 00 приемник[47...63] 🡨 источник[00..15];

• 01 приемник[47...63] 🡨 источник[16..31];

• 10 приемник[47...63] 🡨 истспник[32..47];

• 11 приемник[47...63] 🡨 источник[48..63].

Если использовать один и тот же ХММ-регистр в качестве источника и приемника, то можно выполнять любые перестановки слов в пределах младшего учетверенного слова одного ХММ-регистра, в том числе и инициализацию значением одного слова других слов. Работу команды PSHUFLW поясняет схема, показанная далее.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13, 16; #UD: 17,19,22; #NM: 3; VM: исключения реального режима; #PF(fault-code).

PSLLDQ приемник, количество сдвигов

PSLLDQ (Shift Double Quadword Left Logical) — логический сдвиг влево приемника на число байт количество \_сдвигов.

Синтаксис: PSLLDQ xmm1, imm8

Машинный код: 66 0F 73 /7 i8

Действие: сдвиг влево приемника на число байт, указанных непосредственным операндом количество \_сдвигов. Освобождаемые слева младшие байты обнуляются. Если значение, указанное операндом количество \_сдвигов, больше чем 15, операнд приемник обнуляется.

Флаги: не изменяются.

Исключения: #UD: 10,12,15; #NM: 3; RM: исключения защищенного режима; VM: исключения защищенного режима.

PSLLW/PSLLD/PSLLQ приемник, количество\_сдвигов

PSLLW/PSLLD/PSLLQ (SHIFt packed data Left Logical) — сдвиг влево логический приемника на число бит количество\_сдвигов.

Синтаксис и машинный код:

66 0F F1 /r PSLLW rxmrn-l, rxmm2/m128

66 0F 71 /6 ib PSLLW rxmm1, imm8

66 0F F2 /r PSLLD rxmm1, rxmm2/m128

66 0F 72 /6 ib PSLLD rxmm1, imm8

66 0F F3 /r PSLLQ rxmm1, rxmm2/m128

66 0F 73 /6 ib PSLLQ rxmm1, imm8

Действие: сдвиг упакованных элементов приемника (слов, двойных слов, учетверенных слов) влево на число бит, указанных операндом количество \_сдвигов. Освобождаемые слева биты замещаются нулевыми. Если значение, указанное операндом количество \_сдвигов, больше чем 15 (для слов), 31 (для двойных слов) или 63 (для учетверенных слов), то значение операнда приемник устанавливается равным 0. Операнд количество \_сдвигов может быть либо ХММ-регистром (128-разрядной ячейкой памяти), либо непосредственным 8-разрядным операндом. Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #NM: 3; #UD: 17,19,22; VM: исключения реального режима; #PF(fault-code).

PSRAW/PSRAD приемник, количество\_сдвигов

PSRAW/PSRAD (SHIFt Packed data Right Arithmetic) — сдвиг вправо арифметический приемника на число бит количество\_сдвигов.

Синтаксис и машинный код:

66 0F Е1 /r PSRAWxmmI, xmm2/m128

66 0F 71/4 i8 PSRAW xmm1, imm8

66 0F E2 /r PSRAD xmm1, xmm2/m128

66 0F 72 /4 i8 PSRAD xmm1, imm8

Действие: сдвиг упакованных элементов приемника (слов, двойных слов) вправо на число бит, указанных операндом количество\_сдвигов. Освобождаемые справа биты заполняются значением знакового разряда элемента данных. Если значение, указанное операндом количество\_сдвигов, больше чем 15 (для слов) или 31 (для двойных слов), то каждый элемент данных приемника заполняется начальным значением знакового разряда элемента. Операнд количество \_сдвигов может быть либо ХММ-регистром (128-разрядной ячейкой памяти), либо непосредственным 8-разрядным операндом.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #NM: 3; #UD: 17,19,22; VM: исключения реального режима; #PF(fault-code).

PSRLDQ приемник, количество\_сдвигов

PSRLDQ (Shift Double Quadword Right Logical) — сдвиг вправо приемника на число байт количество\_сдвигов.

Синтаксис: PSRLDQ xmm1, imm8

Машинный код: 66 0F 73 /З i8

Действие: сдвиг вправо приемника на число байт, указанных непосредственным операндом количество\_сдвигов. Освобождаемые справа младшие байты обнуляются. Если значение, указанное операндом количество\_сдвигов, больше чем 15, операнд приемник обнуляется.

Флаги: не изменяются.

Исключения: #UD: 10, 12, 15; #NM: 3; RM: исключения защищенного режима; VM: исключения защищенного режима.

PSRLW/PSRLD/PSRLQ приемник, количество\_сдвигов

PSRLW/PSRLD/PSRLQ (Shift Packed Data Right Logical) — сдвиг вправо логический приемника на число бит количество \_сдвигов.

Синтаксис и машинный код:

66 0F D1 /r PSRLW rxmm1, rxmm2/m128

66 0F 71 /2 i8 PSRLW rxmm1, imm8

66 0FD2/r PSRLD rxmm1, rxmm2/m128

66 0F 72 /2 i8 PSRLD rxmm1, imm8

66 0F D3/r PSRLQ rxmm1, rxmm2/m128

66 0F 73/2 i8 PSRLQ rxmm1, imm8

Действие: сдвиг упакованных элементов приемника (слов, двойных слов, учетверенных слов) вправо на число бит, указанных операндом количество \_сдвшов. Освобождаемые справа биты замещаются нулевыми. Если значение, указанное операндом количество\_сдвиюв, больше чем 15 (для слов), 31 (для двойных слов) или 63 (для учетверенных слов), то значение операнда приемник устанавливается равным 0. Операнд количество\_сдвшов может быть либо ХММ-регистром (128-разрядной ячейкой памяти), либо непосредственным 8-разрядным операндом. Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #NM: 3; #UD: 17,19, 22; VM: исключения реального режима; #PF(lault-code).

PSUBB/PSUBW/PSUBD приемник, источник

PSUBB (Packed Subtraction Bytes) — вычитание упакованных байт. PSUBW (Packed Subtraction Words) — вычитание упакованных слов. PSUBD (Packed Subtraction Double words) — вычитание упакованных двойных слов.

Синтаксис и машинный код:

66 0F F8 /r PSUBB rxmm1, rxmm2/m128

66 0F F9 /r PSUBW rxmm1, rxmm2/m128

66 0F FA /r PSUBD rxmm1, rxmm2/m128

Действие: команда вычитает из элементов источника элементы приемника размером байт/слова/двойное слово в зависимости от кода операции. При переполнении результат формируется в соответствии с принципом циклического переполнения. Результат помещается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(iau!t-code); RM: #GP: 13,16; ^NM: 3; #UD: 17,19,22; VM: исключения реального режима; #PF(fault-code).

PSUBQ приемник, источник

PSUBQ (SUBtract Packed Qyadword integers description) — вычитание учетверенных слов.

Синтаксис и машинный код:

0F FB /r PSUBQ rmmx1, rmmx2/m64

66 0F FB /r PSUBQ rxmm1, rxmm2/m128

Действие: вычесть 64-битные целые значения в источнике и приемнике. Исходя из типа источника, возможны две схемы умножения:

• источник — ММХ-регистр или ячейка памяти m64: приемник[63-0] 🡨приемник[63-0] — источник[63-0], приемник — ММХ-регистр;

• источник — ХММ-регистр или ячейка памяти: приемник[63-0]🡨приемник[63-0] — источник[63-0}; приемник[127-64]🡨приемник[127-6А] — нсточник[127-64].

В результате выполнения команды PSUBQ регистр EFLAGS не отражает факта возникновения ситуации переполнения или переноса. Когда результат умножения слишком большой, чтобы быть представленным в 64-битном элементе приемника, то он "заворачивается" (перенос игнорируется). Для обнаружения подобных ситуаций программное обеспечение должно использовать другие методы.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #NM: 3; #UD: 17,19,22; VM: исключения реального режима; #PF(fault-code).

PSUBSB/PSUBSW приемник, источник

PSUBSB (Packed Subtraction with signed Saturation Bytes) — вычитание упакованных байт со знаковым насыщением. PSUBSW (Packed Subtraction with signed Saturation Words) — вычитание упакованных слов со знаковым насыщением.

Синтаксис и машинный код:

66 0F E8 /r PSUBSB rxmm1, rxmm2/m128

66 0F E9 /r PSUBSW rxmm1, rxmm2/m128

Действие: вычесть элементы источника и приемника размером байт/слово в зависимости от кода операции. Вычитание элементов производится с учетом их знака. При возникновении переполнения результат формируется в соответствии с принципом знакового насыщения:

• PSUBSB — 07fh для положительных чисел и 080h для отрицательных;

• PSUBSW — 07fffh для положительных чисел и 08000h для отрицательных.

Результат помещается в операнд приемник. Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #NM: 3; #UD: 17,19,22; VM: исключения реального режима; #PF(fault-code).

PSUBUSB/PSUBUSW приемник, источник

PSUBUSB (Packed Subtraction with Unsigned Saturation Bytes) — вычитание упакованных байт с беззнаковым насыщением. PSUBUSW (Packed Subtraction with Unsigned Saturation Words) — вычитание упакованных слов с беззнаковым насыщением.

Синтаксис и машинный код:

66 0F D8 /r PSUBUSB xmm1, xmm2/m128

66 0F D9 /r PSUBUSW xmm1, xmm2/m128

Действие: вычесть без учета знака элементы операндов источника и приемника размером байт/слово в зависимости от кода операции. При возникновении переполнения результат формируется в соответствии с принципом беззнакового насыщения:

• PSUBUSB — 00h для результатов вычитания меньших нуля;

• PSUBUSW – 0000h для результатов вычитания меньших нуля.

Результат помещается в операнд приемник.

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #NM: 3; #UD: 17,19,22; VM: исключения реального режима; #PF(fault-code).

PUNPCKHBW/PUNPCKHWD/PUNPCKHDQ/ PUNPCKHQDQ приемник, источник

PUNPCKHBW/PUNPCKHWD/PUNPCKHDQ/PUNPCKHQDQ (UNPaCK High Data) — распаковка старших упакованных байт (слов, двойных слов, учетверенных слов) в слова (двойные слова, учетверенные слова, двойное учетверенное слово).

Синтаксис и машинный код:

66 0F 68 /r PUNPCKHBW rxmm1, rxmm2/m128

66 0F 69 /r PUNPCKHWD rxmm1, rxmm2/m128

66 0F 6A /r PUNPCKHDQ rxmm1, rxmm2/m128

66 0F 6D /r PUNPCKHQDQ rxmm1, rxmm2/m128

Действие: команды PUNPCKHBW, PUNPCKHWD, PUNPCKHDQ и PUNPCKHQDQ производят размещение с чередованием элементов из операндов источник и приемник согласно следующей схеме:

PUNPCKHBW:

приемник[7–0]🡨приемник[71–64]; приемник[71–64]🡨приемник[103–96];

приемник[15–8]🡨источник[71–64]; приемник[79–72]🡨источник[103–96];

приемник[23–16]🡨приемник[79–72]; приемник[87–80]🡨приемник[111–104];

приемник[31–24]🡨источник[79–72]; приемник[95–88]🡨 источник[111–104];

приемник[39–32]🡨приемник[87–80]; приемник[103–96]🡨приемник[119–112];

приемник[47–40]🡨источник[87–80]; приемник[111–104]🡨источник[119–112];

приемник[55–48]🡨приемник[95–88], приемник[119–112]🡨приемник[127–120];

приемник[63–56]🡨источник[95–88]; приемник[127–120]🡨источник[127–120];

PUNPCKHWD:

приемник[15–0]🡨приемник[79–64]; приемник[79–64]🡨приемник[111–96];

приемник[31–16]🡨источник[79–64]; приемник[95–80]🡨источник[111–96];

приемник[47–32]🡨приемник[95–80]; приемник[111–96]🡨привмник[127–112];

приемник[63–А8]🡨источник[95–80]; приемник[127–112]🡨источник[127–112];

PUNPCKHDQ:

приемник[31–0]🡨приемник[95–64]; приемник[95–64]🡨приемник[127–96];

приемник[63–32]🡨источник[95–64]; приемник[127–96]🡨источник[127–96];

PUNPCKHQDQ:

приемник[63–0]🡨приемник[127–64]; приемник[127–64]🡨исоточник[127–64];

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2,42; #MF; #NM: 3; #PF(fault-code); #SS(0): 1; #UD: 10; #AC(0)\_cpl3; RM: #GP: 13, 16; #MF; #NM: 3; #UD: 17; VM: исключения реального режима; #PF(fault-code).

PUNPCKLBW/PUNPCKLWD/PUNPCKLDQ/ PUNPCKLQDQ приемник, источник

PUNPCKLBW/PUNPCKLWD/PUNPCKLDQ/PUNPCKLQDQ (UNPaCK Low Data) — распаковка младших упакованных байт (слов, двойных слов, учетверенных слов) в слова (двойные слова, учетверенные слова, двойное учетверенное слово).

Синтаксис и машинный код:

66 0F 60 /r PUNPCKLBW rxmm1, rxmm2/m128

66 0F 61 /r PUNPCKLWD rxmm1, rxmm2/m128

66 0F 62 /r PUNPCKLDQ rxmm1, rxmm2/m128

66 0F 6C /r PUNPCKLQDQ rxmm1, rxmm2/m128

Действие: команды PUNPCKLBW, PUNPCKLWD, PUNPCKLDQ и PUNPCKLQDQ производят размещение с чередованием элементов из операндов источник и приемник согласно следующей схеме:

PUNPCKLBW:

приемник[7–0]🡨приемник[7–0]; приемник[71–64]🡨приемник[39–32];

приемник[15–8]🡨источник[7–0]; приемник[79–72]🡨источник[39–32];

приемник[23–16]🡨приемник[15–8]; приемник[87–80]🡨приемник[47–40];

приемник[31–24]🡨источник[15–8]; приемник[95–88]🡨 источник[47–40];

приемник[39–32]🡨приемник[23–16]; приемник[103–96]🡨приемник[55–48];

приемник[47–40]🡨источник[23–16]; приемник[111–104]🡨источник[55–48];

приемник[55–48]🡨приемник[31–24], приемник[119–112]🡨приемник[63–56];

приемник[63–56]🡨источник[31–24]; приемник[127–120]🡨источник[63–56];

PUNPCKLWD:

приемник[15–0]🡨приемник[15–0]; приемник[79–64]🡨приемник[47–32];

приемник[31–16]🡨источник[15–0]; приемник[95–80]🡨источник[47–32];

приемник[47–32]🡨приемник[31–16]; приемник[111–96]🡨привмник[63–48];

приемник[63–А8]🡨источник[31–16]; приемник[127–112]🡨источник[63–48];

PUNPCKLDQ:

приемник[31–0]🡨приемник[31–0]; приемник[95–64]🡨приемник[63–32];

приемник[63–32]🡨источник[31–0]; приемник[127–96]🡨источник[63–32];

PUNPCKLQDQ:

приемник[63–0]🡨приемник[63–0]; приемник[127–64]🡨исоточник[63–0];

Флаги: не изменяются.

Исключения: РМ: #GP(0): 2,42; #MF; #NM: 3; #PF(fault-code); #SS(0): 1; #UD: 10;

#AC(0)\_cpl3; RM: #GP: 13, 16; #MF; #NM: 3; #UD: 17; VM: исключения реального режима; #PF(fault-cocle); #AC(0)\_u.

PXOR приемник, источник

PXOR (Packed logical Exclusive OR) — упакованное логическое исключающее ИЛИ.

Синтаксис: PXOR xmm1, xmm2/m128

Машинный код: 66 0F EF /r

Действие: команда производит побитовую операцию логическое исключающее ИЛИ над всеми битами операндов источник и приемник. Результат помещается в операнд приемник. Флаги: не изменяются.

Исключения: РМ: #GP(0): 2, 42; #SS(0): 1; #UD: 10, 12, 15; #NM: 3; #PF(fault-code); RM: #GP: 13,16; #NM: 3; #UD: 17,19,22; VM: исключения реального режима; #PF(fault-code).

SHUFPD приемник, источник, маска

SHUFPD (Shuffle Packed Double-Precision Floating-Point Values Description) — перестановка упакованных значений с плавающей точкой двойной точности.

Синтаксис: SHUFPD xmm1, xmm2/m128, imm8

Машинный код: 66 0F С6 /r i8

Действие: переместить упакованные значения с плавающей точкой двойной точности из приемника и источника в приемник в соответствии со значением непосредственного операнда маска. Биты маски определяют номера упакованных значений с плавающей точкой двойной точности в источнике или приемнике, которые будут перемещены в приемник следующим образом:

• маска.0 = 0: приемник[63–0] 🡨 приемник[63–0];

• маска.0 = 1: приемник[63–0] 🡨 приемник[127–64];

• маска.1 = 0: приемник[127–64] 🡨 источник[63–0];

• маска.1 = 1: приемник[127–64] 🡨 источник[127–64].

Для перестановки в пределах одного регистра можно использовать один и тот же регистр ХММ в качестве источника и приемника.

Исключения: РМ: #GP(0): 37,42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; RM: #GP: 13,16; #NM: 3; #UD: 17-19,22; #XM; VM: исключения реального режима; #PF(fault-code).

SQRTPD приемник, источник

SQRTPD (compute SQuare RooTs of Packed Double-precision floating-point values) — вычисление квадратного корня упакованных значений с плавающей точкой двойной точности.

Синтаксис: SQRTPD rxmm1, rxmm2/m128

Машинный код: 66 0F 51 /r

Действие: вычислить значения квадратных корней упакованных значений с плавающей точкой двойной точности источника по следующей схеме: приемник[63-0]🡨 SQRT(источник[63-0]); приемник[127-64]🡨SQRT(источник[127-64]).

Исключения: SIMD (NE): #O, #U, #I, #Р, #D; PM: #GP(0): 37, 42; #SS(0): 13; #PF(fault-code); #NM: 3; #XM; #UD: 10-12, 15; RM: #GP: 13, 16; #NM: 3; #ХМ; #UD: 17-19, 22; VM: исключения реального режима; #PF(fault-code).

SQRTSD приемник, источник

SQRTSD (compute SQuare RooT of Scalar Double-precision floating-point value) — вычисление квадратного корня скалярного упакованного значения с плавающей точкой двойной точности.

Синтаксис: SQRTSD rxmm1, rxmm2/m64

Машинный код: F2 0F 51 /r

Действие: вычислить значение квадратного корня младшего упакованного значения с плавающей точкой двойной точности источника по схеме: приемник[63-0] 🡨SQRT(источник[63-0]); приемник[127-64] — не изменяется.

Исключения: SIMD (NE): #I, #Р, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12,15; #XM; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #XM; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

SUBPD приемник, источник

SUBPD (SUBtract Packed Double-precision floating-point values) — вычитание упакованных значений с плавающей точкой двойной точности.

Синтаксис: SUBPD rxmm1, rxmm2/m128

Машинный код: 66 0F 5C /r

Действие: вычесть пары упакованных значений с плавающей точкой двойной точности источника и приемника по схеме: приемник[63-0]🡨приемник[63-0] — источник[63-0]; приемник[127-64]🡨приемник[127-64] — источник[127-6А].

Исключения: SIMD (NE): #O, #U, #I, #Р, #D; PM: #GP(0): 37,42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,11,12,15; #XM; RM: #GP: 13,16; #NM: 3; #UD: 17,18, 19, 22; #XM; VM: исключения реального режима; #PF(fault-code).

SUBSD приемник, источник

SUBSD (SUBtract Scalar Double-precision floating-point values) — вычитание скалярных упакованных значений с плавающей точкой двойной точности.

Синтаксис: SUBSD rxmm1, rxmm2/m64

Машинный код: F2 0F 5C /r

Действие: вычесть младшие упакованные значения с плавающей точкой двойной точности источника и приемника по схеме: приемник[63-0]🡨приемник[63-0] — источник[63-0]; приемник[127-63] — не изменяется.

Исключения: SIMD (NE): #O, #U,. #I, #Р, #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #ХМ; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

UCOMISD приемник, источник, условие

UCOMISD (Unordered COMpare Scalar Double-precision floating-point values and set EFLAGS) — сравнение неупорядоченных скалярных значений с плавающей точкой двойной точности и установка регистра EFLAGS.

Синтаксис: UCOMISD xmm1, xmm2/m64

Машинный код: 66 0F 2Е /r

Действие: сравнить неупорядоченные скалярные значения с плавающей точкой двойной точности в разрядах [63-0] приемника и источника. По результату сравнения установить флаги ZF, PF и CF в регистре EFLAGS (см. описание команды COMISD). Отличие команды COMISD от команды UCOMISD состоит в генерации исключения недействительной операции с плавающей точкой (#I): COMISD генерирует его, когда приемник и(или) источник — QNAN или SNAN; команда UCOMISD генерирует #I только в случае, если один из исходных операндов — SNAN. В случае генерации немаскированного исключения с плавающей точкой регистр EFLAGS не модифицируется.

Исключения: SIMD (NE): #I (если операнд — SNaN), #D; PM: #GP(0): 37; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10,11,12,15; #XM; #AC(0)\_cpl3; RM: #GP: 13; #NM: 3; #UD: 17-19, 22; #ХМ; VM: исключения реального режима; #PF(fault-code); #AC(0)\_u.

UNPCKHPD приемник, источник

UNPCKHPD (UNPaCK and interleave High Packed Double-precision floating-point values) — разделение и чередование старших упакованных значений с плавающей точкой двойной точности.

Синтаксис: UNPCKHPD xmm1, xmm2/m128

Машинный код: 66 0F 15 /r

Действие: разделить старшие упакованные значения с плавающей точкой двойной точности в источнике и приемнике и поместить их с чередованием в приемник по схеме: приемник[63-0] 🡨 приемник[127-64]; приемник[127-64] 🡨 источник[127-64].

Исключения: PM: #GP(0): 37, 42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10, 12,15; RM: #GP: 13,16; #NM: 3; #UD: 17-19,22; #XM; VM: исключения реального режима; #PF(fault-code).

UNPCKLPD приемник, источник

UNPCKLPD (UNPaCK and interleave Low Packed Double-precision floating-point values) — разделение и чередование младших упакованных значений с плавающей точкой двойной точности.

Синтаксис: UNPCKLPD xmm1, xmm2/m128

Машинный код: 66 0F 14/r

Действие: разделить младшие упакованные значения с плавающей точкой двойной точности в источнике и приемнике и поместить их с чередованием в приемник по схеме: приемник[63-0] 🡨 приемник[63-0]; приемник[127-64] 🡨 источник[63-0].

Исключения: PM: #GP(0): 37,42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; RM: #GP: 13,16; #NM: 3; #UD: 17-19,22; #XM; VM: исключения реального режима; #PF(fault-code).

XORPD приемник, источник

XORPD (bitwise logical XOR for Double-precision floating-point values) — поразрядное логическое исключающее ИЛИ над упакованными значениями с плавающей точкой двойной точности.

Синтаксис: XORPD xmm1, xmm2/m128

Машинный код: 66 0F 57 /r

Действие: выполнить операцию поразрядного логического исключающего ИЛИ над парами упакованных значений с плавающей точкой двойной точности в приемнике и источнике по схеме: приемник[127-0] 🡨 приемник[127-0]) побитное\_\_ХОR источник[127-0].

Исключения: РМ: #GP(0): 37,42; #NM: 3; #PF(fault-code); #SS(0): 13; #UD: 10-12, 15; #XM; RM: #GP: 13,16; #NM: 3; #UD: 17-19,22; #XM; VM: исключения реального режима; #PF(fault-code).

**Литература**

1. Гук М., Юров В. Процессоры Pentium 4, Athlon и Duron. — СПб.: Питер, 2001.— 512 с.: ил.

 2. Зубков С.В., Assembler для DOS, Windows и Unix.— М.: ДМК, 1999.— 640 с., ил.

 3. Ровдо А.А., Микропроцкссоры от 8086 до Pentium III Xeon и AMD-K6-3.— М.: ДМК, 2000.— 592 с.: ил.