**Команды XMM (SSE)**

Команды SSE делятся на 4 категории:

* SIMD-команды для данных одинарной точности с плавающей запятой (SPFP-команды);
* Дополнительные SIMD-команды для целочисленных данных;
* Команды управления кэшированиукем;
* Команды сохранения и восстановления компонент состояния процессора.

Одна SIMD-команда с плавающей запятой может обрабатывать одновременно четыре 32-разрядных числа одинарной точности с плавающей запятой (называемых SPFP-элементами данных).

SIMD-команды для работы с SPFP-данными используют восемь новых 128-разрядных регистров - XMM-регистров. Команды обращаются к XMM-регистрам по именам: XMM0, XMM1, :, XMM7.

Каждое 32-ра fld зрядное число с плавающей запятой имеет 1 знаковый бит, 8 битов порядка и 23 бита мантиссы, что соответствует стандарту IEEE-754 на формат представления чисел одинарной точности с плавающей запятой.

SIMD-команды поддерживают два типа операций над упакованными данными с плавающей запятой - параллельные и скалярные.

Параллельные операции, как правило, действуют одновременно на все четыре 32-разрядных элемента данных в каждом из 128-разрядных операндов. В именах команд, выполняющих параллельные операции, присутствует суффикс PS.

Скалярные операции действуют на младшие (занимающие разряды 0-31) элементы данных двух операндов. Остальные три элемента данных в выходном операнде не изменяются (исключение составляет команда скалярного копирования MOVSS). В имени команд, выполняющих скалярные операции, присутствует суффикс SS.

SSE-команды имеют следующий синтаксис: **instruction [dest, src]**

Здесь instruction - имя команды, dest обозначает выходной операнд, src - входной операнд.

Доступны следующие SSE-команды (обозначения: xmm - XMM-регистр; mm - MMX-регистр; m - память; imm - 8-битный непосредственный операнд; ir32 - целочисленный регистр):

|  |  |
| --- | --- |
| **ADDPS xmm, xmm/m** | команда попарно складывает упакованные элементы данных и записывает суммы в соответствующие элементы XMM-регистра. |
| **SUBPS xmm, xmm/m** | команда вычитает элементы входного операнда из элементов первого регистра и записывает полученные разности в соответствующие элементы первого регистра. |
| **MULPS xmm, xmm/m** | команда попарно перемножает упакованные элементы. Произведения записываются в соответствующие элементы XMM-регистра. |
| **DIVPS xmm, xmm/m** | команда делит элементы первого операнда на соответствующие элементы входного операнда . Результаты деления записываются в XMM-регистр на место делимых. |
| **ADDSS xmm, xmm/m** | команда складывает младшие элементы данных и записывает результат в младший элемент XMM-регистра. Остальные элементы выходного операнда не меняются. |
| **SUBSS xmm, xmm/m** | команда вычитает младший элемент входного операнда из соответствующего элемента выходного операнда и записывает разность в младший элемент выходного операнда. Остальные элементы выходного операнда не меняются. |
| **MULSS xmm, xmm/m** | команда перемножает младшие элементы данных и записывает результат в младший элемент XMM-регистра. Остальные элементы выходного операнда не меняются. |
| **DIVSS xmm, xmm/m** | команда делит младший элемент выходного операнда на соответствующий элемент входного операнда и записывает результат в младший элемент выходного операнда. Остальные элементы выходного операнда не меняются. |
| **SQRTPS xmm, xmm/m** | команда вычисляет квадратный корень для каждого из четырех чисел во входном операнде и записывает результаты в выходной операнд. |
| **SQRTSS xmm, xmm/m** | команда вычисляет квадратный корень из младшего элемента входного операнда и записывает результат в младший элемент в выходной операнд. Остальные элементы выходного операнда не меняются. |
| **RCPPS xmm, xmm/m** | команда определяет приближенное обратное значение для каждого из четырех чисел входного операнда и записывает результаты в XMM-регистр. |
| **RSQRTPS xmm, xmm/m** | команда вычисляет приближенное обратное значение для квадратного корня из каждого из четырех чисел входного операнда и записывает результаты в XMM-регистр. |
| **RCPSS xmm, xmm/m** | команда определяет приближенное обратное значение для числа, находящегося в младшем элементе входного операнда и записывает результат в младший элемент выходного операнда. Остальные элементы выходного операнда не меняются. |
| **RSQRTSS xmm, xmm/m** | команда вычисляет приближенное обратное значение для квадратного корня из числа , находящегося в младшем элементе входного операнда и записывает результат в младший элемент выходного операнда. Остальные элементы выходного операнда не меняются. |
| **MAXPS xmm, xmm/m** | команда попарно сравнивает элементы данных и записывает большее значение из каждой пары в соответствующий элемент выходного операнда. |
| **MINPS xmm, xmm/m** | команда попарно сравнивает элементы данных и записывает меньшие значения из каждой пары в соответствующие элементы выходного операнда. |
| **MAXSS xmm, xmm/m** | команда сравнивает младшие элементы данных и записывает большее из значений в младший элемент выходного операнда. Остальные элементы выходного операнда не меняются. |
| **MINSS xmm, xmm/m** | команда сравнивает младшие элементы данных и записывает меньшее из значений в младший элемент выходного операнда. Остальные элементы выходного операнда не меняются. |
| **SHUFPS xmm, xmm/m, imm** | команда с непосредственным операндом выбирает из первого операнда два элемента с 2-битными номерами из непосредственного операнда (непосредственный операнд делится справа налево на 4 2-битовых числа) и записывает их в младшие элементы выходного операнда. Из второго операнда выбираются два элемента со следующими 2-битными номерами и записываются в старшие элементы выходного операнда. Перестановка всех элементов происходит одновременно. |
| **UNPCKHPS xmm, xmm/m** | команда копирует третьи элементы входного и выходного операндов в соседние младшие элементы выходного операнда, а четвертые элементы входного и выходного операндов - соответственно, в старшие элементы выходного операнда. Распаковка элементов выполняется одновременно. |
| **UNPCKLPS xmm, xmm/m** | команда копирует первые элементы входного и выходного операндов в соседние младшие элементы выходного операнда, а вторые элементы входного и выходного операндов - соответственно, в старшие элементы этого операнда. Распаковка элементов выполняется одновременно. |

**MOVAPS xmm/m, xmm/m** - команда копирует четыре FP-элемента одним из следующих способов:

* из памяти в XMM-регистр
* из XMM-регистра в память
* из одного XMM-регистра в другой

Линейный адрес соответствует адресу младшего байта группы данных в памяти. Обращение в память должно быть по адресу, кратному 16 байтам; в противном случае генерируется исключение.

**MOVUPS xmm/m, xmm/m** - команда копирует четыре FP-элемента одним из следующих способов:

* из памяти в XMM-регистр
* из XMM-регистра в память
* из одного XMM-регистра в другой

Линейный адрес соответствует адресу младшего байта группы данных в памяти. Команда применяется для обращения к невыровненным данным в памяти.

**MOVHPS xmm/m, xmm/m** - команда копирует два FP-элемента одним из следующих способов:

* из памяти в два старшие элемента XMM-регистра
* из двух старших элементов XMM-регистра в память

При копировании данных из памяти в XMM-регистр, два младших элемента этого регистра не изменяются. Команда работает с адресом младшего байта группы данных в памяти.

**MOVLPS xmm/m, xmm/m** - команда копирует два FP-элемента одним из следующих способов:

* из памяти в два младшие элемента XMM-регистра
* из двух младших элементов XMM-регистра в память

При копировании данных из памяти в XMM-регистр, два старших элемента этого регистра не изменяются. Линейный адрес соответствует адресу младшего байта группы данных в памяти.

**MOVSS xmm/m, xmm/m** - команда копирует один 32-рарядный FP-элемент одним из следующих способов:

* из памяти в младший элемент XMM-регистра
* из младшего элемента XMM-регистра в память
* из младшего элемента одного XMM-регистра в младший элемент другого XMM-регистра

При копировании 32 битов данных из памяти в XMM-регистр, все 96 оставшихся старших битов этого регистра обнуляются. Команда работает с адресом младшего байта группы данных в памяти.

При копировании 32 битов данных из памяти в XMM-регистр, все 96 оставшихся старших битов этого регистра обнуляются. Команда работает с адресом младшего байта группы данных в памяти.

|  |  |
| --- | --- |
| **MOVLHPS xmm, xmm** | команда копирует 64 младших разряда входного регистра в 64 старших разряда выходного регистра, не меняя содержимое 64 младших разрядов выходного регистра. |
| **MOVHLPS xmm, xmm** | команда копирует 64 старших разряда входного регистра в 64 младших разряда выходного регистра, не меняя содержимое 64 старших разрядов выходного регистра. |
| **MOVMSKPS ir32, xmm** | команда копирует содержимое старших (знаковых) разрядов каждого из четырех FP-элементов, находящихся во входном XMM-регистре, в младшие разряды выходного целочисленного регистра, и формирует таким образом 4-битовую маску. Старшие 28 разрядов целочисленного регистра обнуляются. |
| **CMPEQPS xmm, xmm/m; CMPLTPS xmm, xmm/m; CMPLEPS xmm, xmm/m; CMPUNORDPS xmm, xmm/m; CMPNEQPS xmm, xmm/m; CMPNLTPS xmm, xmm/m; CMPNLEPS xmm, xmm/m; CMPORDPS xmm, xmm/m** | команды попарно сравнивают числа в соответствующих элементах операндов, проверяя выполнение условий равно, меньше, меньше или равно, несравнимы, не равно, не меньше, не { меньше или равно}, сравнимы соответственно. Элементы выходного операнда заполняются масками из единиц или нулей в зависимости от результата. |
| **CMPEQSS xmm, xmm/m; CMPLTSS xmm, xmm/m; CMPLESS xmm, xmm/m; CMPUNORDSS xmm, xmm/m; CMPNEQSS xmm, xmm/m; CMPNLTSS xmm, xmm/m; CMPNLESS xmm, xmm/m; CMPORDSS xmm, xmm/m** | команды сравнивают числа в младших элементах операндов, проверяя выполнение условий равно, меньше, меньше или равно, несравнимы, не равно, не меньше, не { меньше или равно}, сравнимы соответственно. Младший элемент выходного операнда заполняется маской из единиц или нулей в зависимости от результата. Содержимое трех старших элементов выходного операнда сохраняется. |
| **ANDPS xmm, xmm/m** | команда вычисляет поразрядное логическое И своих 128-битных входного и выходного операндов. Каждый бит результата полагается равным 1, если оба соответствующих бита операндов равны 1, и равным 0 в противном случае. Результат записывается в выходной операнд. |
| **ANDNPS xmm, xmm/m** | команда сначала инвертирует все разряды выходного операнда (логическое НЕ), а затем вычисляет поразрядное логическое И входного и инвертированного выходного операндов. Каждый бит результата полагается равным 1, если для входного операнда соответствующий бит равен 1, а для выходного - равен 0. В противном случае присваивается 0. Результат записывается в выходной операнд. |
| **ORPS xmm, xmm/m** | команда вычисляет поразрядное логическое ИЛИ своих 128-битных входного и выходного операндов. Каждый бит результата полагается равным 0, если оба соответствующих бита операндов равны 0, и равным 1 в противном случае. Результат записывается в выходной операнд. |
| **XORPS xmm, xmm/m** | команда вычисляет поразрядное логическое исключающее ИЛИ своих 128-битных входного и выходного операндов. Каждый бит результата полагается равным 1, если соответствующие биты операндов содержат различные значения. В случае одинаковых значений присваивается 0. Результат записывается в выходной операнд. |
| **COMISS xmm, xmm/m** | команда выполняет сравнение двух младших элементов операндов и устанавливает для них одно из соотношений: "меньше", "равны", "больше", "несравнимы". По результатам сравнения устанавливаются следующие значения флагов состояния ZF, PF и CF: |

|  |  |  |  |
| --- | --- | --- | --- |
| **Результат** | **Значения в EFLAGS** | | |
|  | ZF | PF | CF |
| Несравнимы | 1 | 1 | 1 |
| Меньше | 0 | 0 | 1 |
| Больше | 0 | 0 | 0 |
| Равны | 1 | 0 | 0 |

Если хотя бы один из сравниваемых элементов операндов содержит значение QNaN или SNaN, флаги устанавливаются в "несравнимы" и генерируется исключение. Согласно стандарту IEEE, величина QNaN определяется как любое нечисловое значение (NaN), у которого старший бит мантиссы равен 1. Величина SNaN определяется как любое нечисловое значение (NaN), у которого старший бит мантиссы равен 0, а хотя бы один из оставшихся битов мантиссы содержит 1. Если все биты мантиссы равны 0, то величина считается равной бесконечности. SNaN обычно используется для активизации обработчика исключений.

|  |  |
| --- | --- |
| **UCOMISS xmm, xmm/m** | команда выполняет те же действия, что и команда COMISS xmm, xmm/m. Разница заключается в обработке нечисловых операндов. Для операндов типа SNaN команда генерирует исключение, а для операндов типа QNaN - нет. Флаги устанавливаются в значение "несравнимы" в каждом из случаев. |
| **CVTPS2PI mm, xmm/m** | команда преобразует FP-числа из двух младших элементов входного операнда в целые 32-разрядные числа со знаком, записываемые в два элемента выходного MMX-регистра. При необходимости результаты округляются в соответствии со значением поля RC в регистре MXCSR. |
| **CVTSS2SI ir32, xmm/m** | команда преобразует одно FP-число, содержащееся в младшем элементе входного операнда, в целое 32-разрядное число со знаком, записываемое в выходной целочисленный регистр. При необходимости результат округляется в соответствии со значением поля RC в регистре MXCSR. |
| **CVTTPS2PI mm, xmm/m** | команда преобразует FP-числа из двух младших элементов входного операнда в целые 32-разрядные числа со знаком, записываемые в два элемента выходного MMX-регистра. Когда необходимо округление, дробная часть результата отбрасывается, без учета значения поля RC в регистре MXCSR. Используйте эту команду для преобразований, где при округлении требуется только отбрасывание дробной части. |
| **CVTTSS2SI ir32, xmm/m** | команда преобразует одно FP-число, содержащееся в младшем элементе входного операнда, в целое 32-разрядное число со знаком, записываемое в выходной целочисленный регистр. Когда необходимо округление, дробная часть результата отбрасывается, без учета значения поля RC в регистре MXCSR. Используйте эту команду для преобразований, где требуется только отбрасывание дробной части при округлении. |
| **CVTPI2PS xmm, mm/m** | команда преобразует два 32-разрядных целых числа со знаком в два FP-числа, записываемые в два младших элемента выходного XMM-регистра. Два старших элемента выходного регистра не изменяются. |
| **CVTSI2SS xmm, ir32/m** | команда преобразует 32-разрядное целое число со знаком в FP-число, записываемое в младший элемент выходного XMM-регистра. Три старших элемента выходного регистра не изменяются. |
| **PEXTRV ir32, mm, imm** | команда копирует 16-разрядное слово, на которое указывают два младших разряда непосредственного операнда, из MMX-регистра в целочисленный регистр. Старшие 16 разрядов целочисленного регистра заполняются нулями. |
| **PINSRW mm, ir32/m, imm** | команда копирует 16-разрядное слово из входного операнда в слово MMX-регистра, на которое указывают два младших разряда непосредственного операнда. |
| **PMINSW mm, mm/m** | команда попарно сравнивает четыре слова со знаком в MMX-регистре с четырьмя словами со знаком во входном операнде. Минимальные значения записываются в выходной операнд. |
| **PMAXSW mm, mm/m** | команда попарно сравнивает четыре слова со знаком в MMX-регистре с четырьмя словами со знаком во входном операнде. Максимальные значения записываются в выходной операнд. |
| **PMINUB mm, mm/m** | команда попарно сравнивает восемь байтов без знака в MMX-регистре с восемью байтами без знака во входном операнде. Минимальные значения записываются в выходной операнд. |
| **PMAXUB mm, mm/m** | команда попарно сравнивает восемь байтов без знака в MMX-регистре с восемью байтами без знака во входном операнде. Максимальные значения записываются в выходной операнд. |
| **PMOVMSKB ir32, mm** | команда копирует старшие (знаковые) биты всех восьми упакованных байтов входного операнда - MMX-регистра и формирует 8-разрядную маску в младших разрядах 32-разрядного целочисленного регистра. Все старшие 24 разряда выходного целочисленного регистра обнуляются. |
| **PMULHUW mm, mm/m** | команда записывает в выходной операнд старшие 16 разрядов попарных промежуточных произведений 16-разрядных слов без знака входного и выходного операндов. |
| **PSHUFW mm, mm/m, imm** | команда выбирает четыре 16-разрядных слова (не обязательно различных) из входного операнда и записывает их в определенном порядке в выходной операнд. Порядок записи слов задается 2-разрядными полями 8-разрядного непосредственного операнда. |
| **MOVNTPS m, xmm** | команда записывает 128 бит SPFP-данных непосредственно в память и не кэширует данные при кэш-промахе (cache miss). При кэш-попадании (cache hit) данные в кэше обновляются, а прямой записи в память не происходит. |
| **MOVNTQ m, mm** | команда записывает 64 бита целочисленных данных непосредственно в память и не кэширует данные при кэш-промахе. При кэш-попадании данные в кэше обновляются, а прямой записи в память не происходит. |
| **MASKMOVQ mm, mm** | команда выборочно записывает байты из MMX-регистра непосредственно в память. Байты выбираются в соответствии с 8-разрядной маской, состоящей из старших битов в байтах второго операнда - MMX-регистра. Единица в некотором разряде маски означает запись соответствующего байта в память, нуль - отсутствие записи. Адрес памяти, по которому производится запись, указывается в регистре EDI. |
| **hrefETCHT0 m** | команда записывает кэшируемые данные из памяти в кэш всех уровней. |
| **hrefETCHT1 m ; hrefETCHT2 m** | команды в архитектуре процессора PentiumR III записывают кэшируемые данные из памяти в кэш L2. |
| **hrefETCHTA m** | команда записывает кэшируемые данные из памяти в кэш для данных со слабой локализацией. В архитектуре процессора PentiumR III данные обходят L2-кэш и переносятся в L1-кэш, используемый для этих целей. |
| **SFENCE** | команда применяется для строгого упорядочения последовательности обращений в память и синхронизации ее с содержимым кэш-памяти, если такие обращения являются слабо упорядоченными. При использовании данной команды в основную память копируются все данные предшествующих команд записи, хранящиеся в буфере записи и кэше. Только после этого будут выполняться следующие команды записи. Команда временно блокирует выполнение лишь последующих команд сохранения данных, не влияя при этом на работу других команд. |
| **STMXCSR m** | команда записывает в 32-разрядное слово памяти содержимое регистра MXCSR. |
|  |  |
| **IDMXCSR m** | команда загружает из памяти 32-разрядное слово состояния и управления в регистр MXCSR. |
| **FXSAVE m** | команда сохраняет в области памяти по указываемому адресу состояние регистров данных с плавающей запятой, MMX-регистров, а также новых регистров процессора PentiumR III. Размер области памяти, в которой хранится информация о состоянии процессора, составляет 512 байт. |
| **FXSTOR m** | команда загружает предварительно сохраненную информацию о состоянии регистров данных с плавающей запятой, MMX-регистров, а также новых регистров процессора PentiumR III из 512-байтной области памяти в соответствующие регистры. Начальный адрес области памяти при загрузке должен быть выровнен на 16 байт. |