КОНСТРУИРОВАНИЕ ПРОГРАММ

Лекция № 17

Арифметические расширения команд процессора. Технология SSE

+375 17 293 8039 (505a-5)

+375 17 320 7402 (ОИПИ НАНБ)

prep@lsi.bas-net.by

ftp://student:2ok*uK2@Rwox@lsi.bas-net.by/

Кафедра ЭВМ, 2022

2022.05.11

Оглавление

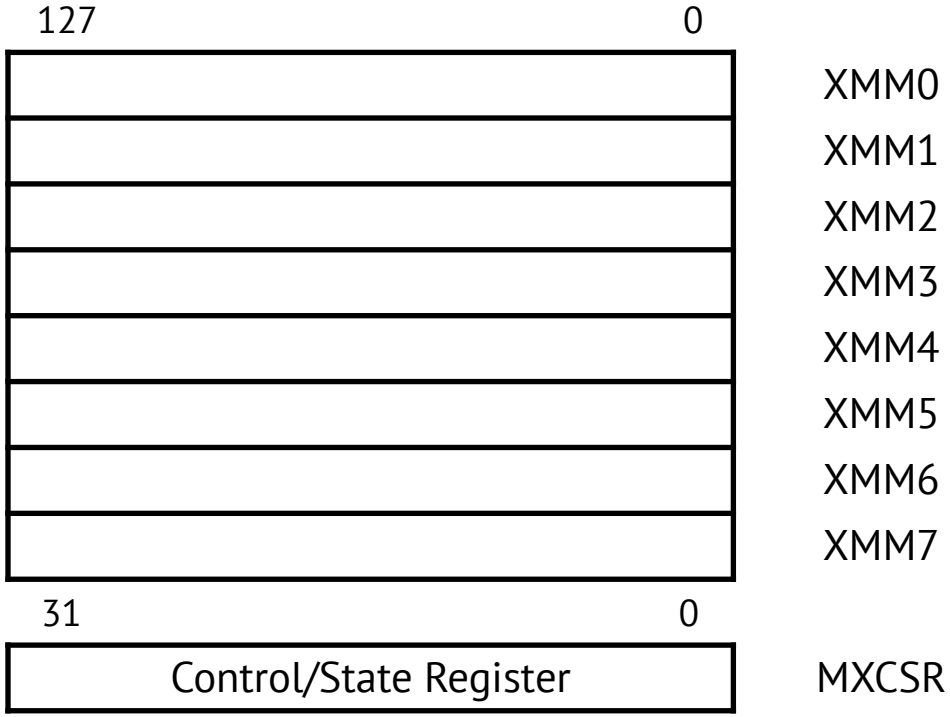
Регистры SSE

Streaming **S**IMD **E**xtension — потоковые расширения

SIMD — Single Instruction — Multiple Data

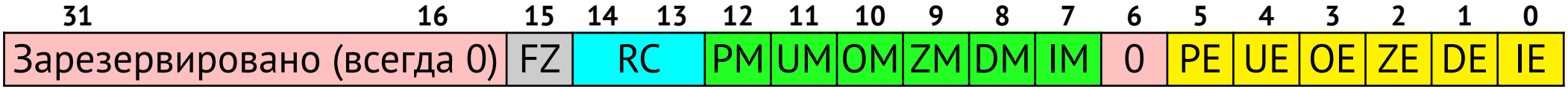
Расширение предназначено для современных приложений, работающих с 2d и 3d графикой, видео-, аудио- и другими видами потоковых данных.

В отличие от MMX расширение SSE не использует ресурсы FPU, а вводит 8 новых независимых регистров размером 128 бит XMM0..XMM7.



Поэтому не требуется команд переключения режимов, типа EMMS — освободить регистры MMX, и параллельно с SSE можно использовать FPU.

Регистр управления/состояния — MXCSR (32 бит)



Биты 0..5 — флаги исключений:

IE (0) — запрошена невыполнимая операция (команда).

DE (1) — денормализованный операнд — запрошена операция над денормализованным числом;

ZE (2) — деление на 0 — запрошено деление на 0;

OE (3) — переполнение — результат слишком большой;

UE (4) — антипереполнение — результат слишком маленький;

PE (5) — неточный результат — результат не может быть представлен точно;

Биты 7..12 — маски исключений.

IM (7) — маска исключения IE;

DM (8) — маска исключения DE;

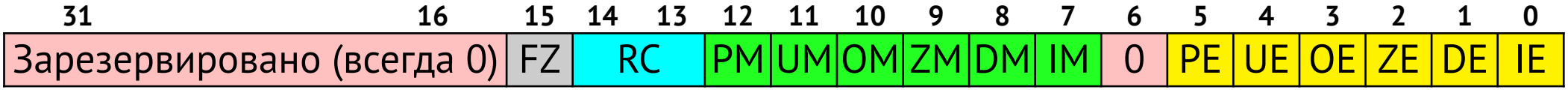
ZM (9) — маска исключения ZE;

OM (10) — маска исключения OE;

UM (11) — маска исключения UE;

PM (12) — маска исключения PE.

По умолчанию все маскирующие биты устанавливаются в 1, так что никакие исключения не обрабатываются.



RC (14..13) — управление округлением

0 (00) — к ближайшему целому (устанавливается по умолчанию);
1 (01) — к отрицательной бесконечности;
2 (10) — к положительной бесконечности;
3 (11) — к нулю;

FZ (15) — режим сброса в ноль (flush-to-zero)

По умолчанию выключен.

В этом режиме команды SSE не превращают слишком маленькое число с плавающей запятой в денормализованное, как этого требует IEEE 754, а возвращают 0.

Знак нуля соответствует знаку получившегося бы денормализованного числа, а также устанавливаются флаги PE и UE.

Типы данных SSE

Основной тип данных — упакованные числа с плавающей запятой одинарной точности.
Один 128-битный регистр содержит 4 таких числа (float).
Это стандартные числа, которые используются FPU.

Целочисленные команды SSE могут работать с упакованными байтами, словами или двойными словами. Но эти данные должны располагаться в регистрах MMX.

Команды SSE

Все команды доступны из любого режима процессора.

Команды пересылки

Скалярные типы – **MOVSS**

Упакованные типы – **MOVAPS/MOVUPS**, **MOVLPS/MOVHPS**, **MOVLHPS/MOVHLPS**

Арифметические команды

Скалярные типы – **ADDSS**, **SUBSS**, **MULSS**, **DIVSS**, **RCPSS**, **SQRTSS**, **MAXSS**, **MINSS**, **RSQRTSS**

Упакованные типы – **ADDPS**, **SUBPS**, **MULPS**, **DIVPS**, **RCPPS**, **SQRTPS**, **MAXPS**, **MINPS**, **RSQRTPS**

Команды сравнения

Скалярные типы – **CMPSS**, **COMISS**, **UCOMISS**

Упакованные типы – **CMPPS**

Перемешивание и распаковка

Упакованные типы – **SHUFPS**, **UNPCKHPS**, **UNPCKLPS**

Команды для преобразования типов

Скалярные типы – **CVTSI2SS**, **CVTSS2SI**, **CVTTSS2SI**

Упакованные типы – **CVTPI2PS**, **CVTPS2PI**, **CVTTPS2PI**

Битовые логические операции

Упакованные типы – **ANDPS**, **ORPS**, **XORPS**, **ANDNPS**

Команды пересылки данных

MOVAPS dst, src — переслать выравненные (**A**rranged) упакованные числа

```
MOVAPS xmm1, xmm2/m128
MOVAPS xmm2/m128, xmm1
```

Копирует 128 бит источника в приемник. Каждый из операндов может быть либо регистром SSE, либо переменной в памяти. Пересылки память-память не поддерживаются — один из операндов должен быть регистром SSE.

Переменная в памяти должна быть выровнена на границу 16 байт.

Если адрес переменной не кратен 16 байт (128 бит), вызывается исключение #GP.

MOVUPS dst, src — переслать невыравненные (**U**narranged) упакованные числа

```
MOVUPS xmm1, xmm2/m128
MOVUPS xmm2/m128, xmm1
```

Копирует 128 бит источника в приемник. Каждый из операндов может быть либо регистром SSE, либо переменной в памяти. Пересылки память-память не поддерживаются — один из операндов должен быть регистром SSE.

В отличие от **MOVAPS** умеет работать с невыровненными данными.

Если данные выровнены, рекомендуется использовать **MOVAPS** — она эффективнее.

MOVHPS dst, src — переслать старшие (**H**igh) упакованные числа

Каждый из операндов может быть либо регистром SSE, либо переменной в памяти.
Пересылки память-память не поддерживаются.

MOVHPS xmm1, m64

Загружает два упакованных значения с плавающей запятой одинарной точности из памяти **m64** в старшее квадрослово **xmm1** (копирует *старшие* 64 бит из источника в приемник).
При загрузке в XMM регистр младшая его часть сохраняется (младшие 64 бита приемника не изменяются).

MOVHPS m64, xmm1

Сохраняет два упакованных значения с плавающей запятой одинарной точности из старшего квадрослова xmm1 в память m64.

MOVLPS dst, src — переслать младшие (**L**ow) упакованные числа

Один из операндов должен быть регистром SSE, другой — переменной в памяти.
Пересылки память-память не поддерживаются.

MOVLPS xmm1, m64

Загружает два упакованных значения с плавающей запятой одинарной точности из памяти m64 в младшее квадрослово xmm1 (копирует *младшие* 64 бит из источника в приемник).

При загрузке в XMM регистр старшая его часть сохраняется (старшие 64 бита регистра-приемника не изменяются).

MOVLPS m64, xmm1

Сохраняет два упакованных значения с плавающей запятой одинарной точности из младшего квадрослова xmm1 в память m64.

MOVHLPS dst, src — переслать старшие упакованные числа в младшие

MOVLHPS xmm1, xmm2

Копирует старшие 64 бит из источника в младшие 64 бита приемника.
Старшие 64 бита приемника не изменяются.
И приемник и источник должны быть только регистрами SSE.

MOVLHPS dst, src — переслать младшие упакованные числа в старшие

MOVLHPS xmm1, xmm2

Копирует младшие 64 бит из источника в старшие 64 бита приемника.
Младшие 64 бита приемника не изменяются.
И приемник и источник – только SEE-регистры.

MOVMSKPS dst, src — переслать маску в переменную.

MOVMSKPS reg, xmm

4-х битная маска, отвечающая знакам четырех вещественных чисел, располагающихся в источнике (128 разрядный регистр SSE), записывается в приемник.

В качестве приемника используется 32-разрядный регистр процессора.

Таким образом биты 3..0 приемника устанавливаются в значения битов 127, 95, 63, 31.

Биты 31..4 приемника обнуляются.

MOVSS dst, src — переслать или слить в назначении одно вещественное число.

```
MOVSS xmm1, xmm2
MOVSS xmm1, m32
MOVSS xmm2/m32, xmm1
```

Копирует младшие 32 бита из источника в приемник.

Операнд может быть либо регистром, либо переменной в пямяти.

Пересылки память-память не поддерживаются.

Если приемник и источник регистры, старшие 96 бит приемника не изменяются.

Если приемник регистр, а источник — память, его старшие 96 бит обнуляются.

Если приемник — переменная в памяти, его старшие 96 бит не изменяются.

Арифметические команды

ADDPS dst, src — сложение упакованных вещественных чисел (**P**acked).

ADDPS xmm1, xmm2/m128

Выполняет параллельное сложение четырех пар чисел с плавающей запятой, располагающихся в источнике (регистр SSE или переменная в памяти) и приемнике (регистр SSE).

Результат записывается в приемник.

ADDSS dst, src — Сложение одного вещественного числа (**S**calar).

ADDSS xmm1, xmm2/m32

Выполняет сложение нулевых чисел (31..0 бит) с плавающей запятой, располагающихся в источнике (регистр SSE, переменная в памяти) и приемнике (регистр SSE).

Результат записывается в биты 31..0 приемника.

Биты 127..32 остаются без изменения.

SUBPS dst, src — Вычитание упакованных вещественных чисел.

SUBPS xmm1, xmm2/m128

Выполняет параллельное вычитание четырех чисел с плавающей запятой, располагающихся в источнике (регистр SSE или переменная в памяти) из чисел, располагающихся в приемнике (регистр SSE).

Результат записывается в приемник.

SUBSS dst, src — Вычитание одного вещественного числа.

SUBSS xmm1, xmm2/m32

Выполняет вычитание «нулевого» числа (31..0 бит) с плавающей запятой, располагающегося в источнике (регистр SSE или переменная в памяти) из числа, находящегося в приемнике (регистр SSE).

Результат записывается в биты 31..0 приемника.

Биты 127..32 остаются без изменения.

MULPS dst, src — Умножение упакованных вещественных чисел.

MULPS xmm1, xmm2/m128

Выполняет параллельное умножение четырех пар чисел с плавающей запятой, находящихся в источнике (регистр SSE или переменная в памяти) и приемнике (регистр SSE).

Результат записывается в приемник.

MULSS dst, src — Умножение одного вещественного числа.

MULSS xmm1,xmm2/m32

Выполняет умножение нулевых чисел (31..0 бит) с плавающей запятой, в источнике (регистр SSE или переменная в памяти) и приемнике (регистр SSE).

Результат записывается в биты 31..0 приемника.

Биты 127..32 остаются без изменения.

DIVPS dst, src – Деление упакованных вещественных чисел.

DIVPS xmm1, xmm2/m128

Выполняет параллельное деление четырех пар чисел с плавающей запятой, находящихся в приемнике (регистр SSE) на числа в источнике (регистр SSE или переменная в памяти).

Результат записывается в приемник.

DIVSS dst, src – Деление одного вещественного числа.

DIVSS xmm1, xmm2/m32

Выполняет деление нулевого числа (31..0 бит) с плавающей запятой, в приемнике (регистр SSE) на число в источнике (регистр SSE или переменная в памяти).

Результат записывается в биты 31..0 приемника.

Биты 127..32 остаются без изменения.

SQRTPS dst, src — корень из упакованных вещественных чисел

SQRTPS xmm1, xmm2/m128

Определяет значение квадратных корней каждого из четырех чисел с плавающей запятой, находящихся в источнике (регистр SSE или переменная в памяти) и записывает их в приемник (регистр SSE).

SQRTSS dst, src — корень из одного вещественного числа

SQRTSS xmm1, xmm2/m32

Определяет значение квадратного корня нулевого (31..0) числа с плавающей запятой, находящегося в источнике (регистр SSE или переменная в памяти) и записывает их в приемник (биты 31..0 регистра SSE).

RCPPS dst, src — Обратная величина (reciprocal) упакованных вещественных чисел.

RCPPS xmm1, xmm2/m128

Выполняет параллельное деление единицы на каждое из четырех чисел с плавающей запятой, находящихся в источнике (регистр SSE или переменная в памяти) и записывает результат в приемник (регистр SSE).

Относительная ошибка $1,5 \times 2^{-12}$ (≈0.0004).

RCPSS dst, src — Обратная величина одного вещественного числа.

RCPSS xmm1, xmm2/m32

Выполняет деление единицы на младшее число (31..0 бит) с плавающей запятой из источника (регистр SSE или переменная в памяти) и записывает результат в биты 31..0 приемника (регистр SSE).

Относительная ошибка $1,5 \times 2^{-12}$ (≈0.0004).

RSQRTPS dst, src — Обратный корень из упакованных чисел.

RSQRTPS xmm1, xmm2/m128

Определяет обратные величины от квадратных корней (**1/sqrt( )**) каждого из четырех чисел с плавающей запятой, находящихся в источнике (регистр SSE или переменная в памяти) и записывает результаты в приемник (регистр SSE).

Относительная ошибка $1,5 \times 2^{-12}$ (≈0.0004).

RSQRTSS dst, src — Обратный корень из одного числа.

RSQRTSS xmm1, xmm2/m32

Определяет обратную величину квадратного корня (**1/sqrt( )**) из младшего числа с плавающей запятой в источнике (регистр SSE или переменная в памяти) и записывает результаты в биты 31..0 приемника (регистр SSE).

Относительная ошибка $1,5 \times 2^{-12}$ (≈0.0004).

MAXPS dst, src — Максимум для упакованных вещественных чисел.

MAXPS xmm1, xmm2/m128

Определяет максимальные числа с плавающей запятой в каждой из четырех пар чисел, находящихся в источнике (регистр SSE или переменная в памяти) и приемнике (регистр SSE).

Результат записывается в приемник.

Если один из операндов SNAN, он возвращается в приемник без изменений.

При сравнении двух нулей, возвращается нуль из источника.

Если оба из операндов NAN, в приемник возвращается NAN из приемника.

MAXSS dst, src — Максимум для одной пары вещественных чисел.

MAXSS xmm1, xmm2/m32

Определяет максимальные числа с плавающей запятой в младшей паре чисел, находящихся в источнике (регистр SSE, переменная в памяти) и приемнике (регистр SSE).

Результат записывается в приемник. Биты 127..32 приемника не меняются.

Если один из операндов SNAN, он возвращается в приемник без изменений.

При сравнении двух нулей, возвращается нуль из источника.

Если оба из операндов NAN, в приемник возвращается NAN из приемника.

MINPS dst, src — Минимум для упакованных вещественных чисел.

MINPS xmm1, xmm2/m128

Определяет минимальные числа с плавающей запятой в каждой из четырех пар чисел, находящихся в источнике (регистр SSE или переменная в памяти) и приемнике (регистр SSE).

Результат записывается в приемника.

Если один из операндов SNAN, он возвращается в приемник без изменений.

При сравнении двух нулей, возвращается нуль из источника.

Если оба из операндов NAN, в приемник возвращается NAN из приемника.

MINSS dst, src — Минимум для одной пары вещественных чисел.

MINSS xmm1,xmm2/m32

Определяет минимальные числа с плавающей запятой в младшей паре чисел, находящихся в источнике (регистр SSE или переменная в памяти) и приемнике (регистр SSE).

Результат записывается в приемник. Биты 127..32 приемника не меняются.

Если один из операндов SNAN, он возвращается в приемник без изменений.

При сравнении двух нулей, возвращается нуль из источника.

Если оба из операндов NAN, в приемник возвращается NAN из приемника.

Команды сравнения

CMPPS dst, src, predicat — Сравнение упакованных вещественных чисел

CMPPS xmm1, xmm2/m128, imm8

Для каждой из четырех пар вещественных чисел, находящихся в источнике (регистр SSE или переменная в памяти) и приемнике (регистр SSE), возвращает либо 0 (ложь) либо 1 (истина), в зависимости от результата сравнения.

Тип сравнения определяется предикатом (число):

Предикат		Проверяемое утверждение	NAN
0	EQ	приемник равен источнику	0
1	LT	приемник строго меньше источника	0
2	LE	приемник меньше либо равен источнику	0
3	UNORD	приемник или источник являются NAN	1
4	NEQ	приемник неравен источнику	1
5	NLT	приемник больше либо равен источнику	1
6	NLE	приемник строго больше источника	1
7	ORD	ни приемник ни источник не являются NAN	0

Если один из операндов — не-число, результатом сравнения является 0 для предикатов 0, 1, 2, 7 и истина для предикатов 3, 4, 5, 6.

Псевдооперации сравнения и их реализация

Pseudo-Op	CMPPS Реализация
CMP**EQ**PS xmm1, xmm2	CMPPS xmm1, xmm2, 0
CMP**LT**PS xmm1, xmm2	CMPPS xmm1, xmm2, 1
CMP**LE**PS xmm1, xmm2	CMPPS xmm1, xmm2, 2
CMP**UNORD**PS xmm1, xmm2	CMPPS xmm1, xmm2, 3
CMP**NEQ**PS xmm1, xmm2	CMPPS xmm1, xmm2, 4
CMP**NLT**PS xmm1, xmm2	CMPPS xmm1, xmm2, 5
CMP**NLE**PS xmm1, xmm2	CMPPS xmm1, xmm2, 6
CMP**ORD**PS xmm1, xmm2	CMPPS xmm1, xmm2, 7

Отношения GT/GE процессор не реализует и, следовательно, соответствующая псевдооперация также будет, скорее всего, не реализована (ассемблер решает).

Соответственно, для эмуляции GT/GE в программном обеспечении требуется более одной инструкции. Для этого программист должен поменять местами операнды соответствующих отношений LT/LE и в случае AVX использовать инструкции перемещения, чтобы гарантировать перемещение маски записи на правильный регистр назначения, а исходный операнд оставить нетронутым.

CMPSS dst, src, predicat — Сравнение одной пары вещественных чисел

CMPSS xmm1, xmm2/m32, imm8

Для младшей пары вещественных чисел, находящихся в источнике (регистр SSE или переменная в памяти) и приемнике (регистр SSE), возвращает либо 0 (ложь) либо 1 (истина), в зависимости от результата сравнения аналогично как для команды CMPPS.

COMISS/UCOMISS dst, src — Сравнение пары чисел с установкой флагов.

COMISS xmm1, xmm2/m32

Выполняет сравнение младшей пары вещественных чисел, находящихся в источнике (регистр SSE или переменная в памяти) и приемнике (регистр SSE) и устанавливает флаги **ZF**, **PF**, **CF** регистра **EFLAGS** в соответсвии с результатом.

Флаги **OF**, **SF**, **AF** обнуляются.

Если одно из сравниваемых чисел — не-число, все три флага (**ZF**, **PF**, **CF**) устанавливаются в 1.

Если сравниваемые числа равны, **ZF=1**, **PF=0**, **CF=0**.

Если **dst < src**, **ZF=0**, **PF=0**, **CF=1**.

Если **dst > src**, **ZF=0**, **PF=0**, **CF=0**.

COMISS приводит к исключению IE, если любой из операндов **sNAN** или **qNAN**.

UCOMISS приводит к исключению IE, если один из операндов **sNAN**.

Команды преобразования типов

CVTPI2PS dst, src — преобразовать упакованные целые в вещественные

CVTPI2PS xmm, mm/m64

Преобразует два 32-битных целых со знаком из источника (регистр MMX или 64-битная переменная) в два упакованных вещественных числа в приемнике (регистр SSE).

Если преобразование нельзя выполнить точно, результат округляется в соответствии с **MXCSR::RC**.

Биты 127..64 приемника не изменяются.

CVTPS2PI dst, src — преобразовать упакованные вещественные в целые

CVTPS2PI mm, xmm/m64

Преобразует младшие два 32-битных вещественных числа из источника (регистр SSE или 64-битная переменная) в два упакованных целых со знаком в приемнике (регистр MMX).

Если преобразование нельзя выполнить точно, результат округляется в соответствии с **MXCSR::RC**.

Если результат больше максимального 32-битного числа со знаком, возвращается целая неопределенность (80000000h).

CVTSI2SS dst, src — преобразовать целое в вещественное

CVTSI2SS xmm1, r/m32

Преобразует 32-битное целое со знаком из источника (32-битный регистр или переменная) в вещественное число в приемнике (регистр SSE).

Если преобразование нельзя выполнить точно, результат округляется в соответсвии с **MXCSR::RC**.

Биты 127..32 приемника не изменяются.

CVTSS2SI dst, src — преобразовать вещественное в целое

CVTSS2SI r32, xmm1/m32

Преобразует младшее 32-битное вещественное число из источника (регистр SSE или 32-битная переменная) в 32-битное целое со знаком в приемнике (32-битный регистр).

Если преобразование нельзя выполнить точно, результат округляется в соответствии с **MXCSR::RC**.

Если результат больше максимального 32-битного числа со знаком, возвращается целая неопределенность (80000000h).

CVTTPS2PI dst, src — преобразовать вещественные в целые с обрезанием (trunk)

CVTTPS2PI mm, xmm/m64

Аналогична CVTPS2PI, но если результат не может быть представлен точно, он всегда округляется в сторону нуля (обрезается).

CVTTSS2SI dst, src — преобразовать вещественное в целое с обрезанием

CVTTSS2SI r32, xmm1/m32

Аналогична **CVTSS2SI**, но если результат не может быть представлен точно, он всегда округляется в сторону нуля (обрезается).

Битовые логические операции

ANDPS/ANDNPS/ORPS/ dst, src — Логическое И/НЕ-И/ ИЛИ/исключающее ИЛИ для SSE

ANDPS xmm1, xmm2/m128

Выполняет операцию побитового «логического И» для источника (регистр SSE или 128-битная переменная) и приемника (регистр SSE). Результат помещается в приемник.

ANDNPS xmm1, xmm2/m128

Выполняет операцию НЕ над содержимым приемника (регистр SSE), затем выполняет операцию «И» над результатом и содержимым источника (регистр SSE или 128-битная переменная) и приемника (регистр SSE). Результат помещается в приемник.

ORPS xmm1, xmm2/m128

Выполняет операцию побитового «логического ИЛИ» для источника (регистр SSE или 128-битная переменная) и приемника (регистр SSE). Результат помещается в приемник.

XORPS xmm1, xmm2/m128

Выполняет операцию побитового «исключающего ИЛИ» для источника (регистр SSE или 128-битная переменная) и приемника (регистр SSE). Результат помещается в приемник.

Целочисленные SIMD-команды

Помимо работы с упакованными вещественными числами расширение включает возможность работы с упакованными целыми, которые располагаются в регистрах **MMX** (mm1, mm2, mm).

`PAVGB/PAVGW dst, src` — усреднение байтов/слов с округлением

```
PAVGB mm1, mm2/m64
PAVGW mm1, mm2/m64
```

Каждый элемент (байт или слово) источника (регистр MMX или 64-битная переменная) добавляется к соответствующему элементу приемника (регистр MMX) как беззнаковое целое.

Каждый из результатов сдвигается вправо на один разряд (делится на два).

В старший бит каждого элемента записывается бит переноса от соответствующего сложения.

В результате этих действий получаются средние арифметические исходных чисел со знаком.

PEXTRW dst, src, idx — Распаковать одно слово

```
PEXTRW reg, mm, imm8   ; mm[idx] -> reg
```

Выделяет одно 16-битное слово источника (регистр MMX) с номером, равным двум младшим битам **idx** (задается непосредственно) и помещает его в младшую половину 32-битного регистра-приемника (РОН).

PINSRW dst, src, idx — Запаковать одно слово

```
PINSRW mm, r32/m16, imm8  ;  r32/m16 -> mm[idx]
```

Считывает слово из источника (16-битная переменная или младшая часть 32-битного регистра) и помещает его в приемник (регистр MMX) в положение, задаваемое двумя младшими битами **idx** (задается непосредственно).

PMAXUB/PMINUB dst, src — Найти максимум/минимум для упакованных беззнаковых байтов

```
PMAXUB mm1, mm2/m64
PMINUB mm1, mm2/m64
```

Для каждой из восьми пар упакованных байт источника (регистр MMX или 64-битная переменная) и приемника (регистр MMX) в приемник записывается максимальный/минимальный байт в паре. Сравнение беззнаковое.

PMAXSW/PMINSW dst, src — Найти максимум/минимум для упакованных знаковых слов

```
PMAXSW mm1, mm2/m64
PMINSW mm1, mm2/m64
```

Для каждой из четырех пар упакованных слов источника (регистр MMX или 64-битная переменная) и приемника (регистр MMX) в приемник записывается максимальное/минимальное в паре. Сравнение знаковое.

PMOVMSKB dst, src — Считать байтовую маску

```
PMOVMSKB reg, mm
```

В приемнике (32-битный регистр) каждый из младших 8 бит устанавливается равным старшему (знаковому) биту соответствующего байта источника (регистр MMX).

Биты 31-8 обнуляются.

PMULHUW dst, src — Старшее умножение без знака (MMX)

PMULHW mm, mm/m64

Умножить упакованные **беззнаковые** слова из источника (регистр MMX или 64-битная переменная) и из приемника (регистр MMX) и поместить старшие 16-бит 32-битного результата в соответствующее слово в приемнике.

PSADBW dst, src — Сумма абсолютных разностей (абсолютное отклонение)

PSADBW mm1, mm2/m64

Вычисляет абсолютные разности восьми пар байтов из источника (регистр MMX или 64-битная переменная) и приемника (регистр MMX) как целые без знака, после чего суммирует результаты и помещает их в младшее слово приемника.

Старшие слова приемника обнуляются.

PSHUFW dst, src, idx — Переставить упакованные слова

PSHUFW mm1, mm2/m64, imm8

Вместо каждого из четырех слов приемника (регистр MMX) размещается слово из источника (регистр MMX или 64-битная переменная) с номером, указанным соответствующей парой бит индекса (непосредственно заданное 8-битовое число).

Например, если индекс равен 10101010b, второе слово источника будет скопировано во все четыре слова приемника.

SHUFPS dst, src, idx — Переставить упакованные вещественные

SHUFPS xmm1, xmm2/m128, imm8

Помещает в старшие два вещественных числа приемника (регистр SSE) любые из четырех чисел из источника (регистр SSE или 128-битная переменная).

В младшие два числа приемника помещаются любые из четырех чисел приемника.

Индекс определяет, какие числа куда переставлять.

Биты 1 и 0 указывают номер числа из приемника, записываемое в нулевую позицию приемника.

Биты 3 и 2 — номер числа из приемника, записываемое в первую позицию приемника.

Биты 5 и 4 — номер числа из источника, записываемое в третью позицию приемника.

Биты 7 и 6 — номер числа из источника, записываемое в четвертую позицию приемника.

UNPCKHPS dst, src — Распаковать и распределить старшие вещественные числа

UNPCKHPS xmm1, xmm2/m128

dst — регистр SSE
src — регистр SSE или 128-битная переменная.

В нулевую позицию приемника записывается второе число из источника.
В первую — второе число приемника
Во вторую — третье (старшее) число из источника.
В третью (старшую) — третье (старшее) число приемника.

UNPCKLPS dst, src — Распаковать и распределить младшие вещественные числа

UNPCKLPS xmm1, xmm2/m128

dst — регистр SSE
src — регистр SSE или 128-битная переменная.

В нулевую позицию приемника записывается нулевое число из источника.
В первую — нулевое число приемника
Во вторую — первое число из источника.
В третью (старшую) — первое число приемника.

Требование к обеим инструкциям — выравнивание данных в памяти на 16 байт (128 бит).

Команды управления состоянием

STMXCSR dst — Сохранить регистр MXCSR в памяти

STMXCSR m32

Помещает значение регистра SSE MXCSR в приемник (32-битная переменная).

LDMXCSR src — Загрузить регистр MXCSR из памяти

LDMXCSR m32

Помещает значение источника (32-битная переменная) в регистр управления и состояния SSE MXCSR.

FXSAVE dst — Сохранить состояние FPU, MMX и SSE

FXSAVE m512byte

Сохраняет содержимое всех регистров FPU, MMX и SSE в приемнике (512-байтная область памяти).

FXRSTOR src — Восстановить состояние FPU, MMX и SSE

FXRSTOR m512byte

Восстанавливает содержимое всех регистров FPU, MMX и SSE из источника (512-байтная область памяти, заполненная командой **FXSAVE**).

Формат области сохранения для Pentium III

Байты	Объект	Байты	Объект
1..0	FCW (FPU)	105..96	ST4 (MM4)
3..2	FSW (FPU)	121..112	ST5 (MM5)
5..4	FTW (FPU)	137..128	ST6 (MM6)
7..6	FOP (FPU)	153..144	ST7 (MM7)
11..8	FIP (FPU)	175..160	XMM0
13..12	FCS (FPU)	191..176	XMM1
19..16	FDP (FPU)	207..192	XMM2
21..20	FDS (FPU)	223..208	XMM3
27..24	MXCSR	239..224	XMM4
41..32	ST0 (MM0)	255..240	XMM5
57..48	ST1 (MM1)	271..256	XMM6
73..64	ST2 (MM2)	287..272	XMM7
89..80	ST3 (MM3)	511..288	зарезервировано

Команды управления кешированием

MASKMOVQ src, mask — Запись байтов минуя кеш в память

MASKMOVQ mm1, mm2

src — регистр MMX;
mask — регистр MMX.

Данные из источника записываются в память по адресу **DS:EDI** (**DS:DI**).
DS может быть заменен с использованием соответствующего префикса.
Выравнивания в памяти не требуется.
Старший бит каждого байта маски определяет, записывается ли соответствующий байт источника в пямять или нет.
Например, бит 7 маски разрешает запись нулевого байта источника.
Если байт не записывается (s-бит соответсвующего байта маски сброшен), соответствующий байт в памяти обнуляется.
Команда введена для того, чтобы по возможности уменьшить загрязнение кэша при работе с потоками данных SSE если основным типом данных является байт.

MOVNTQ dst, src — Запись 64-бит из регистра MMX в память, минуя кеш

MOVNTQ m64, mm

dst — 64-битная переменная в памяти;
src — регистр MMX;

Содержимое источника записывается в приемник минуя кэш.

MOVNTPS dst, src — Запись 128-бит из регистра SSE в память минуя кеш

MOVNTPS m128, xmm1

dst — 128-битная переменная в памяти;
src — регистр SSE;

Содержимое источника записывается в приемник минуя кэш.

PREFETCHT0 **addr** — Перенести данные в кэш T0
PREFETCHT1 **addr** — Перенести данные в кэш T1
PREFETCHT2 **addr** — Перенести данные в кэш T2
PREFETCHNTA **addr** — Перенести данные в кэш NTA

PREFETCHn m8

Выбирает строку кеша, которой принадлежит **addr** для перемещения ближе к процессору.
Команды перемещают данные, располагающиеся по указанному адресу, в кэш.
T0 — помещает данные в кэш всех уровней;
T1 — помещает данные в кэш всех уровней, кроме нулевого;
T2 — помещает данные в кэш всех уровней, кроме нулевого и первого;
NTA — помещает данные в кэш для постоянных данных;

Процессор не обязан выполнять данные команды — это просто «подсказки».

SFENCE — защита записи (барьерная синхронизация)

При работе с памятью процессор может выполнять обращения к памяти не в том порядке, в каком они указаны в программе.

Команда **SFENCE** гарантирует, что все операции записи в память, расположенные в тексте программы до нее, будут выполнены раньше, чем процессор начнет выполнять операции, помещенные в тексте программы после данной команды.

Определение поддержки SSE

Перед тем, как начинать работу с SSE, нужно убедиться, что выполнены три условия:
Бит 2 регистра CR0 (эмуляция сопроцессора) должен быть равен нулю;
Бит 9 регистра CR4 (поддержка команд FXSAVE/FXRSTOR) — равен 1;
Бит 25 регистра EDX после команды CPUID::EAX=01H (поддержка SSE) — равен 1;

Исключения SSE

Особые ситуации при выполнении инструкций SSE вызывают системное исключение #XF (INT 19).

Его обработчик может прочитать содержимое регистра MXCSR, чтобы определить тип исключения и выполнить соответствующие действия.

Команды SSE могут вызывать и обычные системные исключения:

#UD — неопределенная команда;

#NM — расширение отсутствует;

#SS — переполнение стека;

#GP — общая ошибка памяти;

#PF — ошибка страничной защиты;

#AC — невыровненное обращение к памяти.

Собственные исключения, вызываемые командами SSE, отражаются флагами в регистре MXCSR:

#I — невыполнимая команда (вызывается перед выполнением команды);

#Z — деление на ноль (вызывается перед выполнением команды);

#D — денормализованный операнд (вызывается перед выполнением команды);

#O — переполнение (вызывается после выполнения команды);

#U — антипереполнение (вызывается после выполнения команды);

#P — потеря точности (вызывается после выполнения команды);