КОНСТРУИРОВАНИЕ ПРОГРАММ

Лекция № 17

Арифметические расширения команд процессора. Технология MMX

+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

2022.05.04(04.29)

Оглавление

Арифметические расширения команд процессора. Технологии SIMD

SIMD (Single Instruction, Multiple Data) — одна команда, много данных.

Технологии **SIMD** представляют собой расширения базовой архитектуры процессоров x86 и включают в себя:

- дополнительные регистры;
- типы данных;
- команды.

Основная цель включения этих расширений в архитектуру x86 — добиться более высокой производительности работы мультимедийных приложений, а также систем обработки и передачи данных. В практическом плане **SIMD** реализована как две взаимосвязанные технологии обработки данных:

- технология **MMX** (**M**ulti**M**edia e**X**tensions) — мультимедийные расширения — предназначена для высокоэффективной обработки данных целочисленного типа, имеющих разрядность 64 бита;
- технология **SSE** (**S**treaming **S**IMD **E**xtensions) — потоковые SIMD-расширения — предназначена для эффективной обработки данных вещественного типа с разрядностью 128 бит.

```
$ lscpu | grep Флаги
Флаги:  fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ht tm pbe syscall nx rdtscp lm co nstant_tsc
arch_perfmon pebs bts nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq
dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic
popcnt tsc_deadline_timer aes xsave avx lahf_lm epb pti ssbd ibrs ibpb stibp
tpr_shadow vnmi flexpriority ept vpid xsaveopt dtherm ida arat pln pts md_clear
flush_l1d
```

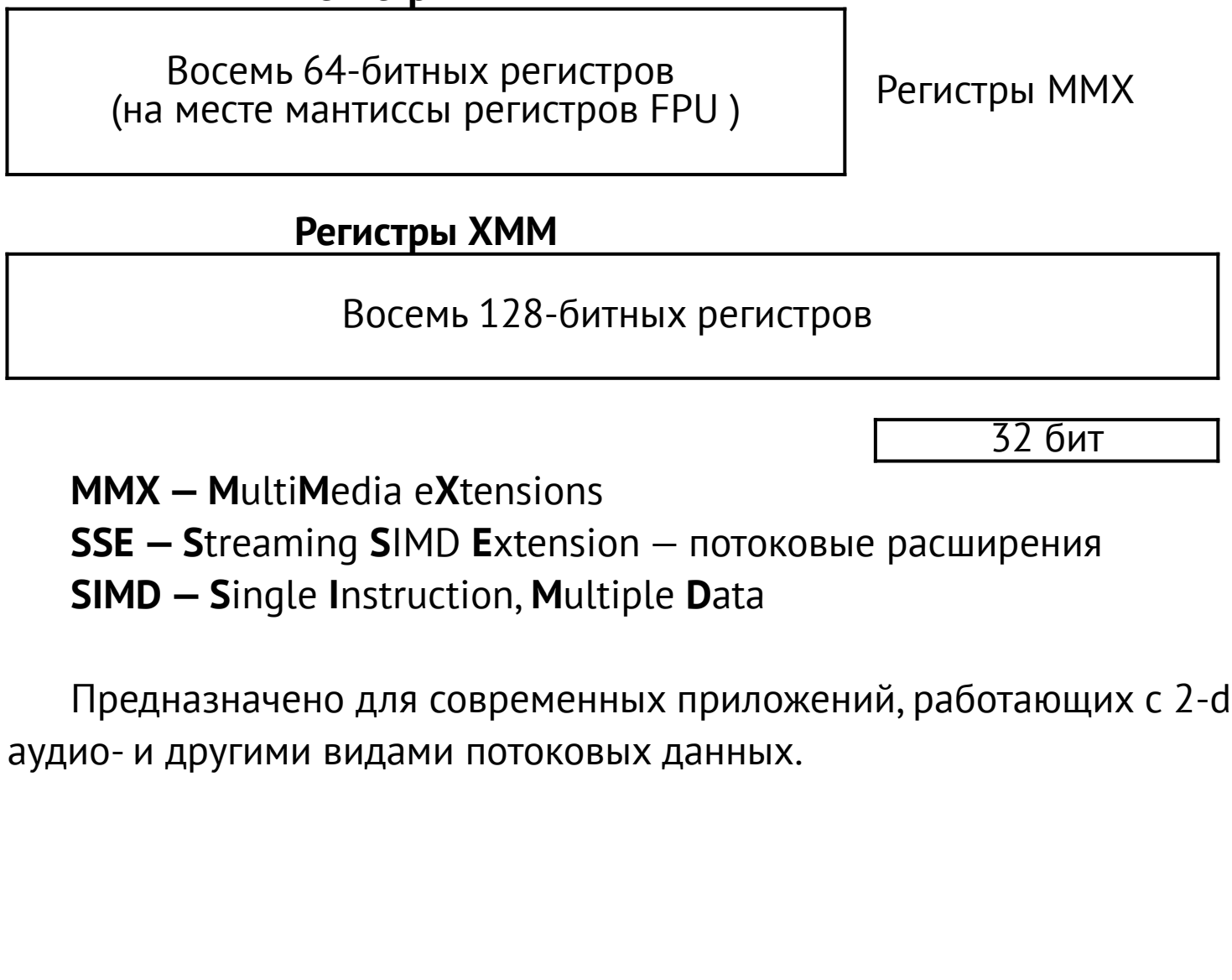
Главное преимущество архитектуры **SIMD** заключается в том, что многие вычисления можно выполнять одновременно или, как говорят, параллельно над несколькими операндами, что позволяет увеличить быстродействие программ.

Вышеуказанные технологии позволяют разработать высокопроизводительные приложения при решении следующих задач:

- кодирование, декодирование и обработка сигналов;
- распознавание речи;
- обработка и захват видеосигналов;
- манипулирование объектами 3D-графики;
- обработка звука;
- промышленное проектирование (CAD/CAM).

Регистры MMX и XMM

Регистры MMX



Регистры XMM

MMX — Multi**M**edia e**X**tensions
SSE — Streaming **S**IMD **E**xtension — потоковые расширения
SIMD — Single **I**nstruction, **M**ultiple **D**ata

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

Расширение MMX

Начиная с модификации Pentium P54C все процессоры Intel содержат расширение **MMX**, предназначенное для увеличения эффективности программ, работающих с большими потоками данных (обработка мультимедийной информации), в тех случаях, когда необходимо выполнять **несложные операции над массивами однотипных чисел**.

Многие прикладные программы используют воспроизведение звука и видео, высокоточную трехмерную графику и анимацию, богатые возможности мультимедиа.

Обычно в таких программах

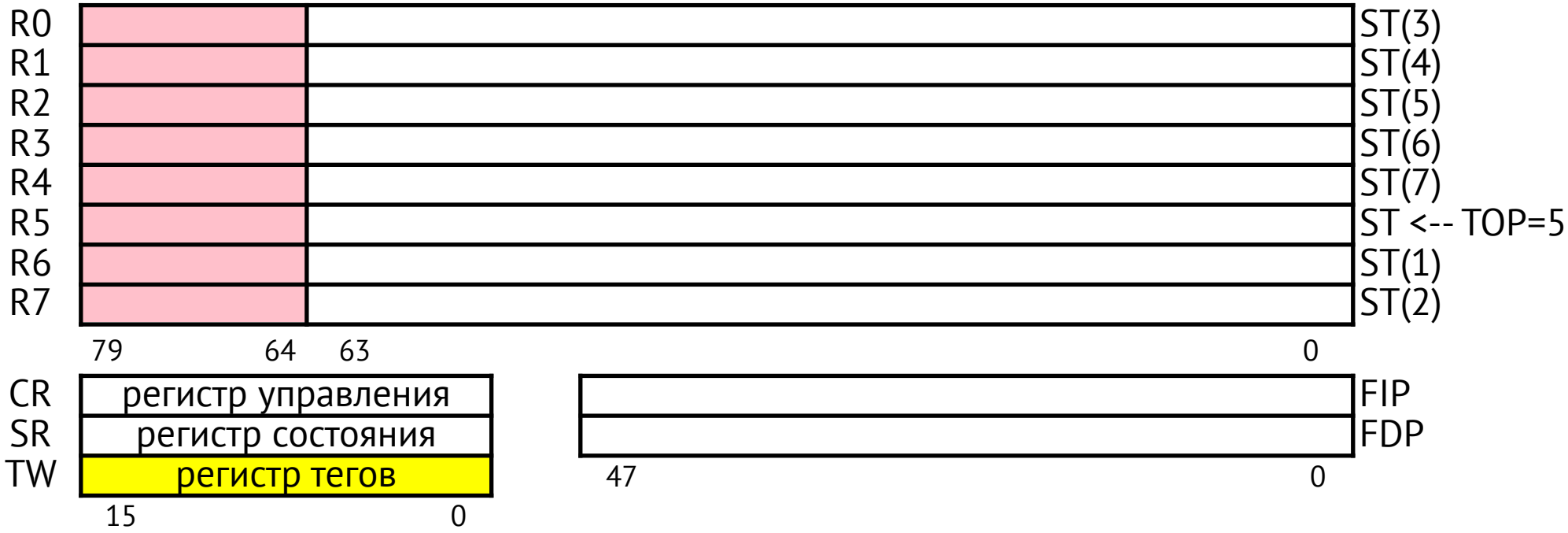
- данные имеют небольшое число разрядов;
- однотипные операции выполняются над многими данными;
- поток команд редко зависит от данных;
- нужна высокая производительность вычислений.

MMX предоставляет несколько новых типов данных, регистров и команд, позволяющих осуществлять арифметические действия и логические операции над несколькими числами одновременно.

Регистры MMX

Расширение MMX включает восемь 64-битных регистров общего пользования **MM0**, **MM1**, ... **MM7**.

Физически новых регистров не появилось — **MM0...MM7** — это в точности мантиссы восьми регистров FPU, от **R0** до **R7**.



При работе с MMX-командами используются регистры стека математического сопроцессора R0 - R7. При этом вместо 80 бит задействуются 64, а стековая организация, требуемая для операций сопроцессора, не используется.

Регистровый стек в операциях MMX-расширения рассматривается как группа из восьми независимых 64-разрядных регистров

При записи числа в регистр MMX оно оказывается в битах 63-0 соответствующего регистра FPU **R0..R7**.

Экспонента и знаковый бит заполняются единицами (неопределенность).

Запись числа в регистр FPU приводит к изменению соответствующего регистра MMX и наоборот.

Любая команда MMX кроме **EMMS** приводит к тому, что поле **TOP** регистра **SR** и весь регистр **TW** в FPU обнуляются.

Команда **EMMS** (очистить состояние MMX), освобождающая регистры MMX, заполняет весь регистр тегов TW единицами (регистр пуст).

Таким образом одновременное использование базовой арифметики FPU и MMX оказывается невозможным.

Если необходимо одновременное использование базовой арифметики и MMX, каждый раз перед переходом от FPU к MMX следует сохранять/восстанавливать состояние FPU командами **FSAVE/FRSTOR**. Эти команды сохраняют состояние регистров MMX также, как и FPU.

При совместном использовании математического сопроцессора и MMX-расширения последней выполняемой командой MMX должна быть команда EMMS.

Все MMX-команды выполняются в том же режиме процессора, что и команды FPU, что вызывает изменения содержимого регистра состояния (SR) сопроцессора.

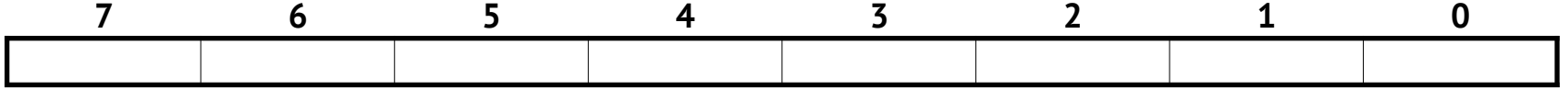
Команда EMMS обеспечивает корректный переход процессора от выполнения фрагмента программного кода с MMX-командами к обработке обычных команд с плавающей точкой.

При этом EMMS устанавливает в 1 все разряды RS. В противном случае все последующие операции с плавающей точкой будут давать некорректные результаты и генерировать исключение **Stack overflow**.

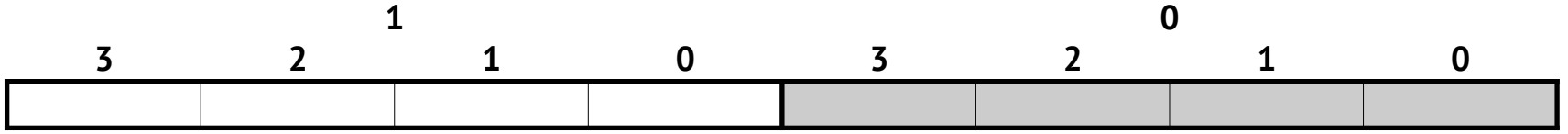
Типы данных MMX

MMX использует четыре новых типа данных:

учетверенное слово — простое 64-битное число (8 байт);



упакованные двойные слова — два 32-битных двойных слова, упакованные в 64-битный тип данных. Двойное слово 1 занимает биты 63..32, двойное слово 0 — биты 31..0;



упакованные слова — четыре 16-битных слова, упакованные в 64-битный тип данных. Слово 3 занимает биты 63..48, слово 0 — биты 15..0;

упакованные байты — восемь байт, упакованных в 64-битный тип данных. Байт 7 за**нимает биты 63..56, байт 0 — биты 7..0.**

Команды MMX перемещают все эти упакованные данные в память и обычные регистры, как многобайтовые целые, но выполняют арифметические и логические операции над каждым объектом в отдельности.

Циклическая арифметика и арифметика с насыщением

Обработка данных MMX-расширением может выполняться одним из двух способов:

- с использованием либо **циклической** арифметики (wraparound arithmetic);
- либо арифметики с **насыщением** (saturation arithmetic).

Если команда задействует циклическую арифметику (другое название — арифметика с циклическим переносом) и результат операции выходит за двоичную разрядную сетку используемого типа данных, то «лишние» старшие биты результата отбрасываются. Например, сложение байтов 01h и FFh дает 00h и бит переноса -->CF).

Если команда использует арифметику с насыщением и результат операции превышает максимальное представимое значение, то в выходной операнд записывается это максимальное значение (происходит «насыщение»).

Аналогично, если результат операции оказывается меньше нижней границы допустимого диапазона, то в выходной операнд записывается минимальное возможное значение.

Большинство команд технологии MMX обрабатывают данные по правилам циклической арифметики, а **некоторые команды** задействуют арифметику с насыщением.

В арифметике с насыщением MMX-команды сложения, вычитания и упаковки данных могут обрабатывать числа со знаком или без знака.

Данные со знаком и без знака имеют различный допустимые диапазоны. Поэтому, если используется арифметика с насыщением, то при выходе результата операции за пределы допустимого диапазона в выходной операнд записываются различные значения, в зависимости от типа данных.

Например, если результат превышает 7FFFh, то 16-разрядное слово со знаком считается равным 7FFFh, а слово без знака — нет.

Если результат меньше 8000h, то 16-разрядное слово со знаком считается равным 8000h.

При работе с цветом 8 bpc насыщение позволяет ему превращаться в белый (255) при переполнении и черный (0) при антипереполнении, в то время как обычная арифметика привела бы к инверсии цвета.

Команды MMX

Большинство команд MMX-расширения имеют следующий синтаксис:

```
COP dst, src
```

Здесь **dst** — это выходной операнд, или операнд-приемник, а **src** — входной операнд, или операнд-источник. В качестве операндов *могут* использоваться:

- регистр MMX;
- обычный регистр CPU;
- память.

Но один из операндов обязательно должен быть регистром MMX.

Обычно входная информация извлекается из обоих операндов, а результат записывается в выходной операнд.

Команды MMX условно можно разделить на следующие группы:

- команды пересылки данных;
- команды преобразования типов (упаковки и распаковки);
- команды сложения и вычитания;
- команды сравнения;
- логические команды;
- команды сдвига;
- команды умножения.

Синтаксис MMX-команд

Большинство команд имеют *суффикс*, который определяет тип данных и используемую арифметику:

us (unsigned saturation) — арифметика с насыщением, данные без знака или, по-другому, беззнаковое насыщение.

s или **ss (signed saturation)** — арифметика с насыщением, данные со знаком или, по-другому, знаковое насыщение;

если в суффиксе нет ни символа **s**, ни символов **ss**, то применяется циклическая арифметика (wraparound).

b, **w**, **d**, **q** — эти буквы указывают тип данных.

Если в суффиксе есть две из этих букв, первая соответствует входному операнду, вторая — выходному.

Пример

```
PADDUSW MM0, meml
```

Здесь суффикс **us** означает, что в команде используется арифметика с насыщением без знака, а операнды имеют разрядность слов.

Первое слагаемое находится в MMX-регистре **MM0**, а второе — в памяти по адресу **meml**.

Результат сохраняется в регистре **MM0**.

Команды управления состоянием MMX

EMMS — освободить регистры MMX

Если выполнялись какие-нибудь команды MMX, кроме **EMMS**, все регистры FPU помечаются как занятые в регистре **TW**.

Команда **EMMS** помечает все регистры FPU, как пустые для того, чтобы после завершения работы с MMX можно было передать управление процедуре, использующей FPU.

Команды пересылки данных

MOVD dst, src — пересылка двойного слова
MOVQ dst, src — пересылка учетверенного слова

Команда копирует двойное слово (32 бита) или учетверенное слово (64 бита) из источника (регистр MMX, обычный регистр или переменная) в приемник (регистр MMX, обычный регистр или переменная, но один из операндов обязательно должен быть регистром MMX).

Если приемник — регистр MMX, двойное слово записывается в младшую половину (биты 31-0), а старшая заполняется нулями.

Если источник — регистр MMX, в приемник записывается младшее двойное слово этого регистра.

Используется в MMX, SSE2, AVX, AVX512F

Команды преобразования типов

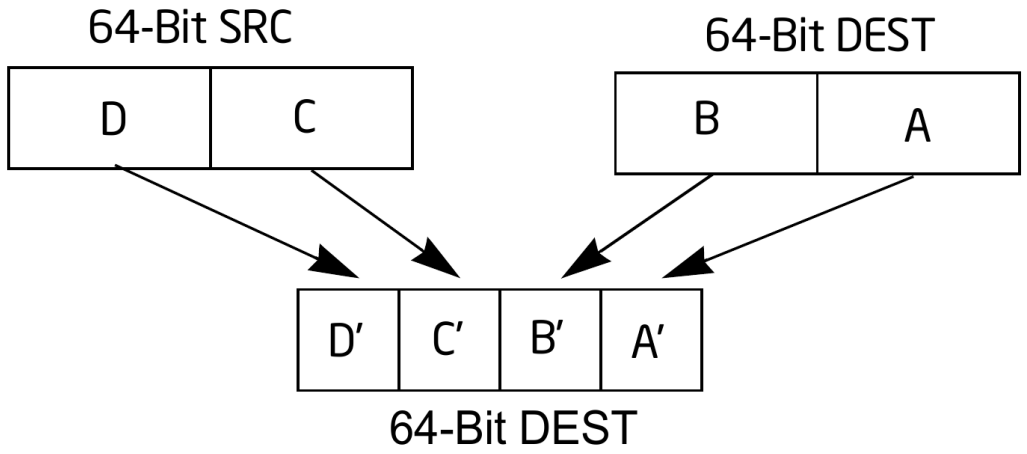
PACKSSWB dst, src — упаковка со знаковым насыщением (**S**igned **S**aturation)
PACKSSDW dst, src

Команды упаковывают слова со знаковым насыщением в байты (PACKSSWB) и двойные слова со знаковым насыщением в слова (PACKSSDW).

PACKSSWB копирует 4 слова (со знаком) из приемника (регистр MMX) в 4 младших байта (со знаком) приемника и копирует 4 слова (со знаком) из источника (регистр MMX или переменная) в 4 старших байта (со знаком) приемника.

Если значение какого-нибудь слова больше +128 или меньше -128, в байты помещаются числа +128 и -128, соответственно.

PACKSSDW аналогично работает с двойными словами, упаковывая их с насыщением и помещая в слова приемника.



PACKUSWB dst, src — упаковка слов в байты с беззнаковым насыщением
PACKUSDW dst, src — упаковка двойных слов в слова с беззнаковым насыщением

Копирует четыре слова (со знаком), находящееся в приемнике (регистр MMX), в 4 младших байта (без знака) приемника и копирует четыре слова (со знаком) из источника (регистр MMX или переменная) в старшие четыре байта (без знака) приемника. Если значение какого-нибудь слова больше 255 или меньше 0, в байты помещаются числа 255 и 0, соответственно.

Распаковка и объединение старших элементов

```
PUNPCKHBW dst, src
PUNPCKHWD dst, src
PUNPCKHDQ dst, src
```

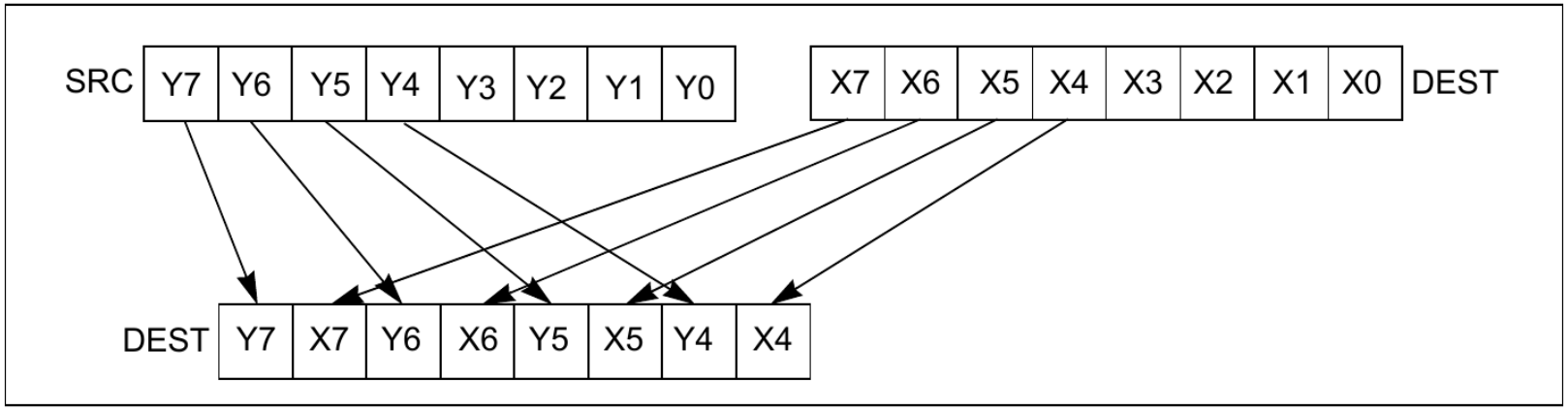
Команды распаковывают старшие элементы источника (регистр MMX, память) и приемника (регистр MMX) и записывают их в приемник через один.

Команда PUNPCKHBW объединяет по 4 старших байта источника и приемника.

Команда PUNPCKHWD объединяет по 2 старших слова, а PUNPCKHDQ — копирует в приемник по одному старшему двойному слову из источника и приемника.

Если источник содержит нули, эти команды переводят старшую половину приемника из одного формата данных в другой, дополняя увеличиваемые элементы нулями.

PUNPCKHBW переводит упакованные байты в упакованные слова, PUNPCKHWD — слова в двойные слова, PUNPCKHDQ — единственное старшее двойное слово приемника в учетверенное.



Распаковка и объединение младших элементов

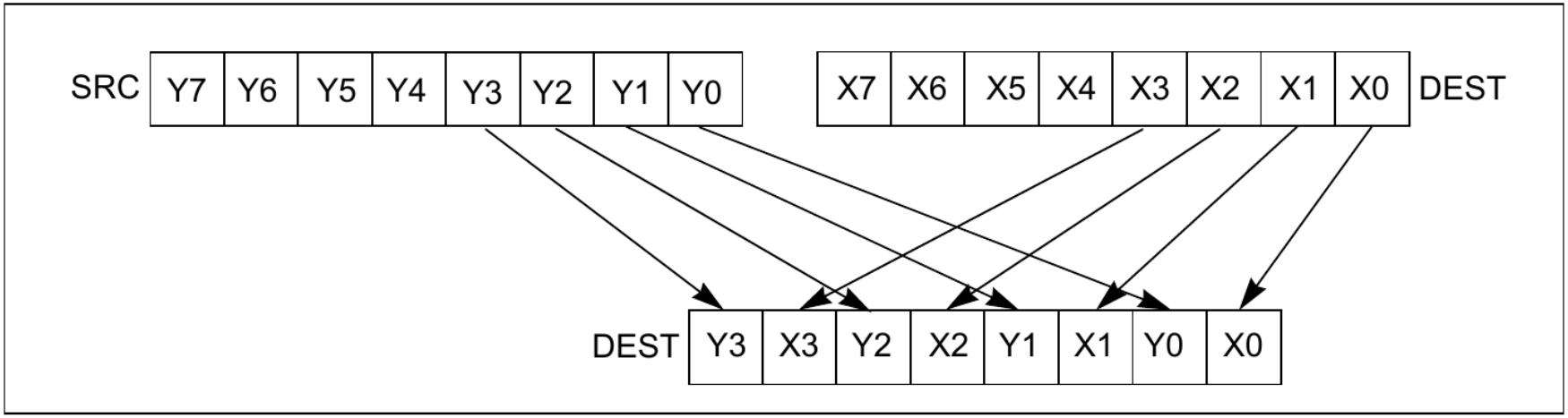
```
PUNPCKLBW dst, src
PUNPCKLWD dst, src
PUNPCKLDQ dst, src
```

Команды распаковывают младшие элементы источника (регистр MMX, память) и приемника (регистр MMX) и записывают их в приемник через один аналогично командам распаковки старших элементов.

Команда PUNPCKHBW объединяет по 4 младших байта источника и приемника.

Команда PUNPCKHWD объединяет по 2 младших слова, а PUNPCKHDQ — копирует в приемник по одному младших двойному слову из источника и приемника.

Если источник содержит нули, эти команды аналогично PUNPCKH* переводят младшую половину приемника из одного формата данных в другой, дополняя увеличиваемые элементы нулями.



Арифметические операции MMX

`PADDB dst, src` — сложение байтов
`PADDW dst, src` — сложение слов
`PADDD dst, src` — сложение двойных слов

Команды выполняют сложение отдельных элементов данных источника (регистр MMX или переменная) и соответствующих элементов приемника (регистр MMX).

Если при сложении возникает перенос, он не влияет ни на элементы, расположенные рядом, ни на флаг переноса, а просто игнорируется (обрыв переноса).

PADDSB dst, src — сложение байтов с насыщением
PADDSW dst, src — сложение слов с насыщением

Команды выполняют сложение отдельных элементов данных источника (регистр MMX или переменная) и соответствующих элементов приемника (регистр MMX).

Если при сложении сумма выходит за пределы элемента данных со знаком, в качестве результата используется соответствующее максимальное или минимальное число.

PADDUSB dst, src — беззнаковое сложение байтов с насыщением
PADDUSW dst, src — беззнаковое сложение слов с насыщением

Команды выполняют сложение отдельных элементов данных источника (регистр MMX или переменная) и соответствующих элементов приемника (регистр MMX).

Если при сложении сумма выходит за пределы элемента данных без знака, в качестве результата используется соответствующее максимальное число или 0 (минимальное число).

PSUBB dst, src — вычитание байтов
PSUBW dst, src — вычитание слов
PSUBD dst, src — вычитание двойных слов

Команды выполняют вычитание отдельных элементов данных источника (регистр MMX или переменная) и соответствующих элементов приемника (регистр MMX).

```
dst <- dst - src
```

Если при вычитании возникает заем, он не влияет ни на элементы, расположенные рядом, ни на флаг переноса, а просто игнорируется.

PSUBSB dst, src — вычитание байтов с насыщением
PSUBSW dst, src — вычитание слов с насыщением

Команды выполняют вычитание отдельных элементов данных источника (регистр MMX или переменная) и соответствующих элементов приемника (регистр MMX).

Если при вычитании разность выходит за пределы элемента данных со знаком, в качестве результата используется соответствующее максимальное или минимальное число.

PSUBUSB dst, src — беззнаковое вычитание байтов с насыщением
PSUBUSW dst, src — беззнаковое вычитание слов с насыщением

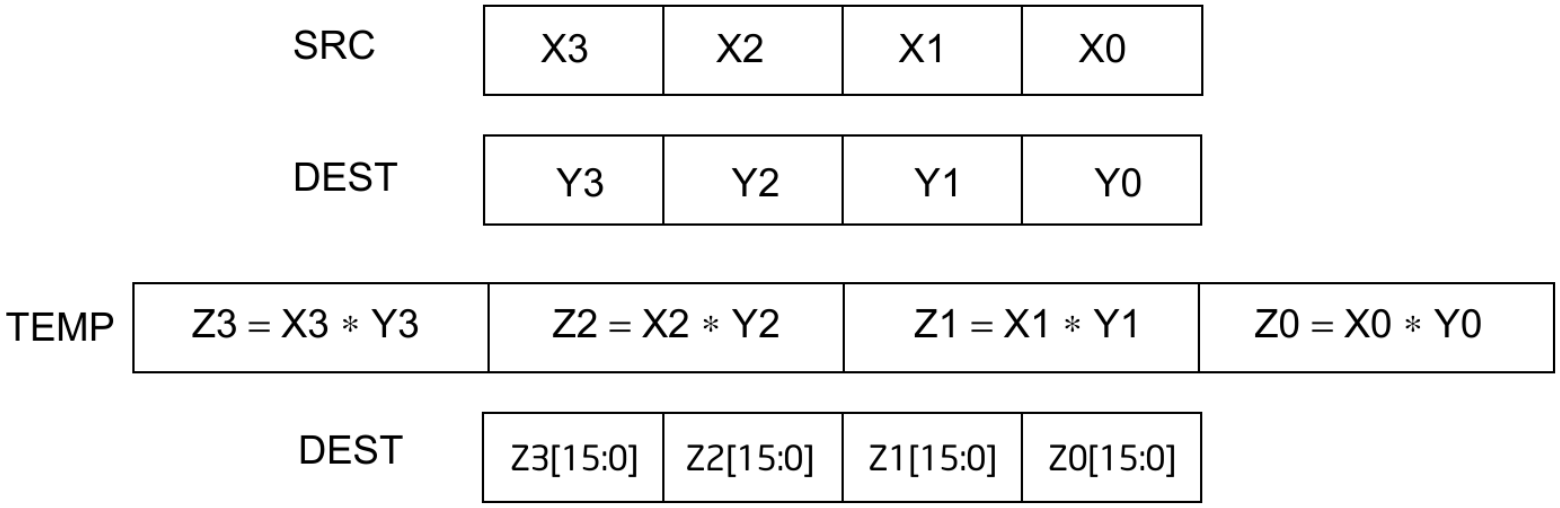
Команды выполняют вычитание отдельных элементов данных источника (регистр MMX или переменная) и соответствующих элементов приемника (регистр MMX).

Если при вычитании разность выходит за пределы элемента данных без знака, в качестве результата используется соответствующее максимальное число или 0 (минимальное число).

PMULLW dst, src — младшее умножение

Умножает каждое из четырех слов со знаком из источника (регистр MMX или переменная) на соответствующее слово со знаком из приемника (регистр MMX).

Младшее слово каждого из результатов записывается в соответствующую позицию приемника.



PMULHW dst, src — старшее умножение

Умножает каждое из четырех слов со знаком из источника (регистр MMX или переменная) на соответствующее слово со знаком из приемника (регистр MMX).

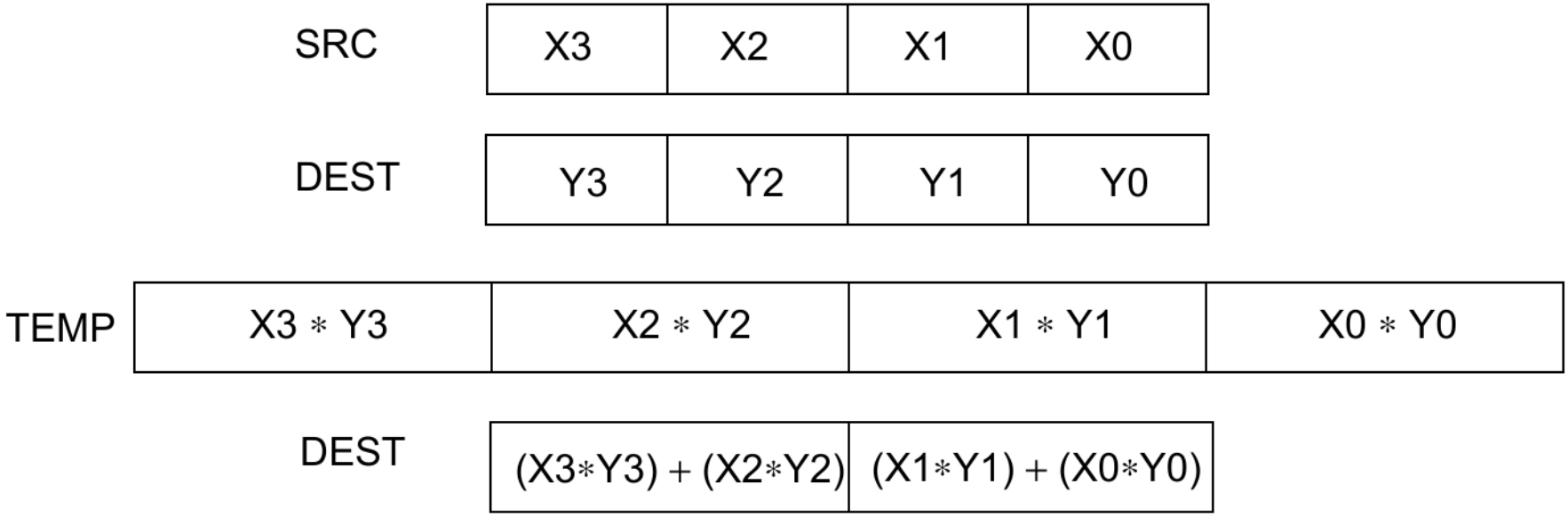
Старшее слово каждого из результатов записывается в соответствующую позицию приемника.

`PMADDWD dst, src` — умножение и сложение

Умножает каждое из четырех слов со знаком из источника (регистр MMX или переменная) на соответствующее слово со знаком из приемника (регистр MMX).

Произведения двух старших пар складываются между собой и их сумма записывается в старшее двойное слово приемника.

Сумма произведений двух младших пар слов записывается в младшее двойное слово.



Команды сравнения MMX

PCMPEQB dst, src — проверка байт на равенство
PCMPEQW dst, src — проверка слов на равенство
PCMPEQD dst, src — проверка двойных слов на равенство

Команды сравнивают отдельные элементы данных источника (регистр MMX или переменная) с соответствующими элементами приемника (регистр MMX).

Если пара сравниваемых элементов равна, соответствующий элемент приемника заполняется единицами, если неравны — нулями.

PCMPGTB dst, src — сравнение байт
PCMPGTW dst, src — сравнение слов
PCMPGTD dst, src — сравнение двойных слов

Команды сравнивают отдельные элементы данных источника (регистр MMX или переменная) с соответствующими элементами приемника (регистр MMX).

Если элемент приемника больше, чем соответствующий элемент источника, все биты в этом элементе приемника устанавливаются в единицы, если элемент приемника меньше либо равен соответствующему элементу источника — нулями.

Логические операции MMX

`PAND dst, src` — логическое И

Команда выполняет побитовое «логическое И» над источником (регистр MMX или переменная) и приемником (регистр MMX). Результат сохраняется в приемнике.

Каждый бит результата устанавливается в 1 только если соответствующий бит источника и приемника был равен 1, иначе сбрасывается.

`PANDN dst, src` — логическое НЕ-И

Команда выполняет побитовое «логическое НЕ» (инверсию битов) над приемником (регистр MMX) и после этого «логическое И» над приемником и источником (регистр MMX или переменная). Результат сохраняется в приемнике.

Каждый бит результата устанавливается в 1 только если соответствующий бит источника был равен 1, а приемника — 0, иначе сбрасывается.

Эта логическая операция называется «штрих Шеффера»

`POR dst, src` — побитовое логическое ИЛИ

`PXOR dst, src` — побитовое логическое ИСКЛЮЧАЮЩЕЕ ИЛИ ($\oplus \mod 2$)

PSLLW dst, src — логический сдвиг слова влево
PSLLD dst, src — логический сдвиг двойного слова влево
PSLLQ dst, src — логический сдвиг учетверенного слова влево

Команды сдвигают влево биты в каждом элементе приемника (регистр MMX) на число битов, указанное в источнике (8-битовое число, регистр MMX или переменная).

При сдвиге младшие биты заполняются нулями, так что, например, команды

```
psllw    mm0, 15 ;
pslld    mm0, 31 ;
psllq    mm0, 63 ;
```

обнуляют регистр MM0.

PSRLW dst, src — логический сдвиг слова вправо
PSRLD dst, src — логический сдвиг двойного слова вправо
PSRLQ dst, src — логический сдвиг учетверенного слова вправо

Команды сдвигают вправо биты в каждом элементе приемника (регистр MMX) на число битов, указанное в источнике (8-битовое число, регистр MMX или переменная).

При сдвиге старшие биты заполняются нулями.

PSRAW dst, src — арифметический сдвиг слова вправо
PSRAD dst, src — арифметический сдвиг двойного слова вправо

Команды сдвигают вправо биты в каждом элементе приемника (регистр MMX) на число битов, указанное в источнике (8-битовое число, регистр MMX или переменная). При сдвиге самый старший (знаковый) бит используется для заполнения освобождающегося места. Фактически происходит деление на 2 в степени источника.

Команды управления состоянием MMX

EMMS — освободить регистры MMX

Если выполнялись какие-нибудь команды MMX, кроме **EMMS**, все регистры FPU помечаются как занятые в регистре **TW**.

Команда **EMMS** помечает все регистры FPU, как пустые для того, чтобы после завершения работы с MMX можно было передать управление процедуре, использующей FPU.