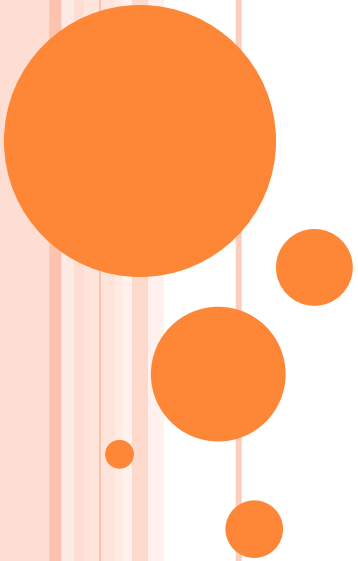
MMX расширение

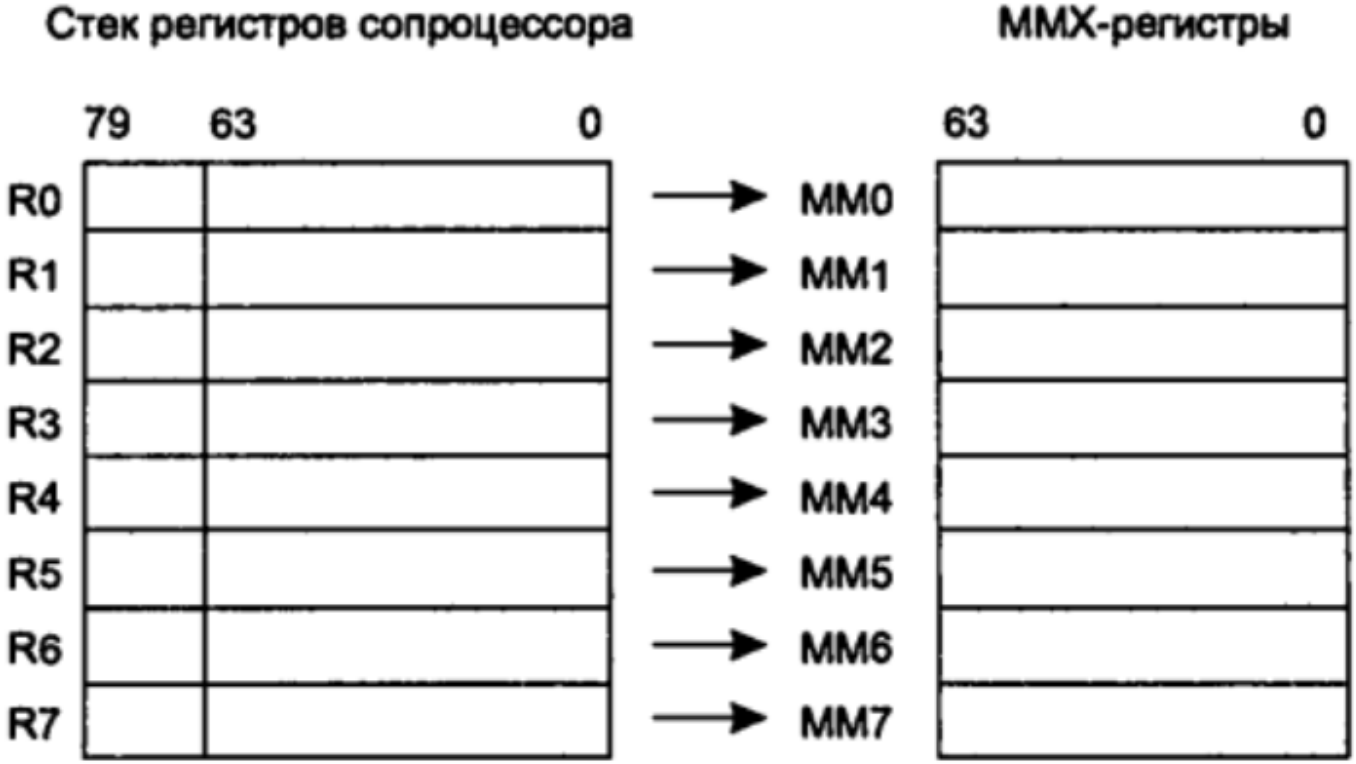


ТЕХНОЛОГИЯ SIMD

- расширение базовой архитектуры процессоров Intel;
- SIMD - Single Instruction, Multiple Data = одна команда, много данных;
- параллельная обработка нескольких данных одной командой;
- две взаимосвязанные технологии:
  - технологии MMX (MultiMedia extensions — мультимедийные расширения) – для эффективной обработки мультимедийных данных целочисленного типа длиной до 64 бит;
  - SSE (Streaming SIMD Extensions — потоковые SIMD-расширения) – эффективная обработки данных вещественного типа разрядностью 128 бит.
- эффективное решение следующих задач:
  - цифровая обработка сигналов;
  - распознавание речи;
  - обработка и захват видеосигналов;
  - манипулирование объектами 3О-графики;
  - обработка 3О-звука;
  - промышленное проектирование (CAD/CAM).

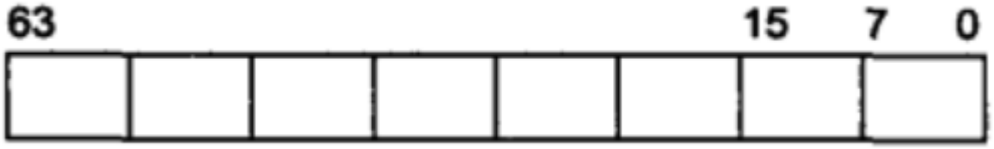
MMX-регистры

- Используется 8 64-разрядных регистров MM0 – MM7.
- MMX- регистры «накладываются» на регистры сопроцессора R0 – R7 (используются младшие 64 бит).
- Стековая организация регистров сопроцессора не используются: MMX-регистры независимы друг от друга.
- Любая MMX команда «захватывает» все регистры сопроцессора, обнуляя регистр тегов TWR (00 – регистр занят ненулевым числом).
- Последней MMX командой должна быть EMMS, которая заносит единицы во все биты TWR: все регистры сопроцессора пусты.

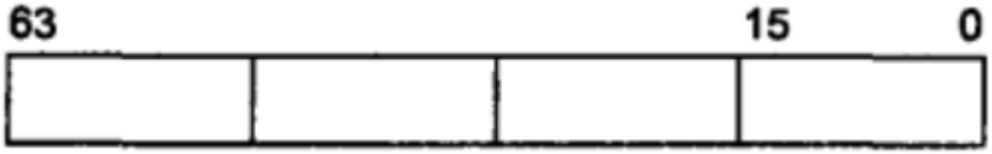


ТИПЫ ДАННЫХ КОМАНД MMX

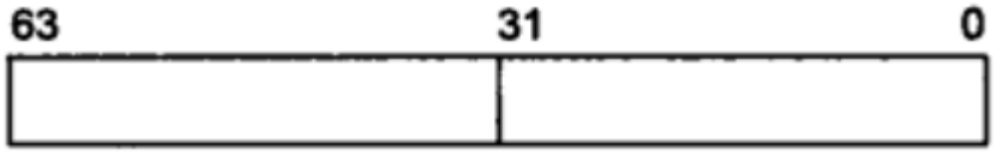
- Упакованные байты – MMX регистр содержит 8 байт, которые могут обрабатываться параллельно.



- Упакованные слова – MMX регистр содержит 4 16-разрядных слова, которые могут обрабатываться параллельно.

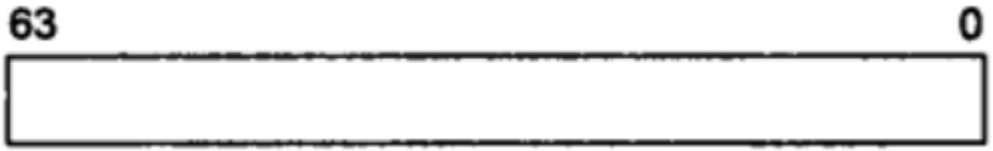


- Упакованные двойные слова – MMX регистр содержит два 32-разрядных слова, которые могут обрабатываться параллельно.



ТИПЫ ДАННЫХ КОМАНД ММХ

- 64-разрядные слова– действие выполняется с ММХ регистром как единым целым (например, пересылка данных).



- Нельзя одновременно использовать сопроцессор для вычислений с плавающей запятой и ММХ расширение!

АРИФМЕТИКА MMX

Арифметика с циклическим переносом

- Если результат превышает максимальное значение на n, то результатом является минимальное значение плюс n минус 1.
- Если результат меньше минимального значения на n, то результатом является максимальное значение минус n плюс 1
  - 01h + FFh = 00h
  - 00h – 01h = FFh

Арифметика с насыщением

- Если результат превышает максимальное значение, то он равен максимальному значению.
- Если результат меньше минимального значения, то он равен минимальному значению.
  - 01h + FFh = FFh
  - 00h – 01h = 00h

Формат команд MMX

имя_команды приемник, источник

- Приемник – регистр MM0 – MM7. Содержит один из операндов, результат записывается в приемник.
- Источник – регистр MM0 – MM7 или ячейка памяти.
- Имя команды начинается с p, далее идет мнемокод действия и завершается одним или несколькими суффиксами:
  - us – беззнаковое насыщение;
  - s или ss – знаковое насыщение; если этот суффикс отсутствует – циклический перенос;
  - b, w, d, q – обрабатываемый тип данных; две буквы обозначают преобразование данных из одной длины в другую.
- Все команды MMX не изменяют флаги процессора!!

Пример.

paddusw MM0, meml

- сложение, беззнаковое насыщение, сложение данных длиной слово (16 разрядов)

Группы команд MMX

- Команды передачи данных.
- Команды сложения и вычитания.
- Команды сдвига.
- Логические команды.
- Команды умножения.
- Команды сравнения.
- Команды упаковки и распаковки.

КОМАНДЫ ПЕРЕДАЧИ ДАННЫХ

movd

- Копирование 32-разрядных чисел.
- Из ячейки памяти или 32-разрядного РОН в ММХ регистр
- Из младших 32 разрядов ММХ регистра в 32-разрядный РОН или ячейку памяти.
- При копировании в ММХ регистр старшая его часть обнуляется.
- Никакая другая команда ММХ не может содержать операнд в 32-разрядном РОН.

movq

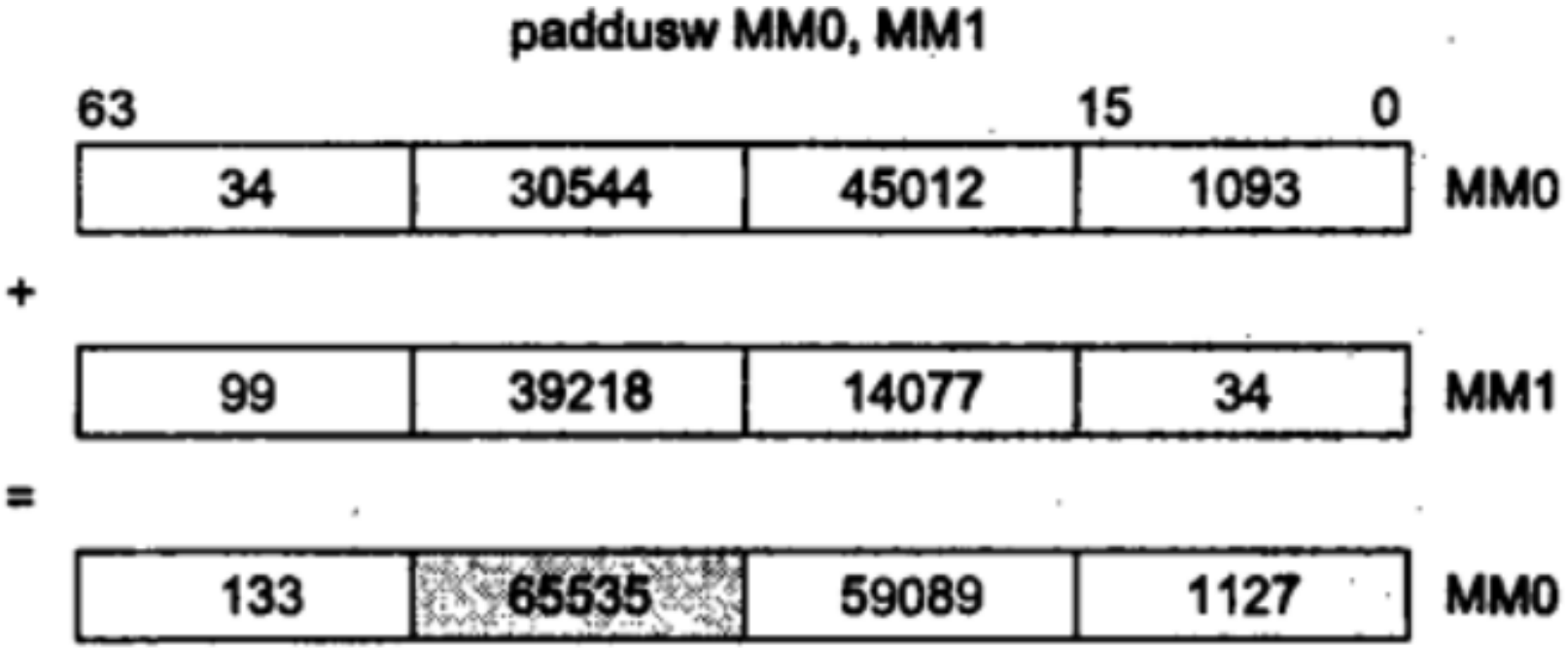
- Копирование 64-разрядных чисел.
- Между ММХ регистрами.
- Между ММХ регистром и ячейкой памяти.

- Среди всех ММХ команд только movd и movq могут иметь приемник в памяти.

КОМАНДЫ СЛОЖЕНИЯ

- **paddb, paddw, paddd**– сложение байт, слов, двойных слов с циклическим переносом.
- **paddsb, paddsw** – сложение байт или слов знаковым насыщением.
- **paddusb, paddusw** – сложение байт или слов с беззнаковым насыщением.

Пример. Сложение слов с беззнаковым насыщением.



ПРИМЕР СЛОЖЕНИЯ

```
upstr    proc    inst:dword, buf:dword, len:dword

         mov eax, len        ; количество элементов в массивах
         mov ebx, 8          ; подготовка к делению на 8
         xor edx, edx
         div ebx             ; делим на 8 – сколько циклов сложения требуется
         mov ecx,eax         ; количество циклов сложения заносим в счетчик
         mov esi, inst       ; Загрузка адресов массивов в esi, ebx
         mov ebx, buf
l1:      movq MM0, [esi]     ; Загрузка первых 8 байт массива в MMX0
         paddb MM0, [ebx]; Сложение 8 байт массивов
         movq [esi], MM0     ; Запись результата в исходный массив
         add esi, 8          ; Переход к следующим 8 байтам
         add ebx, 8
         dec ecx             ; Если есть следующие 8 байт - повтор
         jnz l1
         cmp edx, 0          ; Есть ли остаток байт после сложения всех восьмерок
```

ПРИМЕР СЛОЖЕНИЯ

```
        jz fin
l2:
        mov ecx, edx     ; Сложение оставшихся байт обычным
способом
l3:
        mov al, [esi]
        add al,[ebx]
        mov [esi], al
        inc esi
        inc ebx
        dec ecx
        jnz l3
fin:
        mov eax, inst    ; Возвращаемое значение – в аккумулятор
        emms
        ret
upstr           endp
```

КОМАНДЫ ВЫЧИТАНИЯ

- **psubb, psubw, psubd**– вычитание байт, слов, двойных слов с циклическим переносом.
- **psubsb, psubsw** – вычитание со знаковым насыщением.
- **psubusb, psubusw** – вычитание с беззнаковым насыщением.

Пример. Вычитание слов со знаковым насыщением.

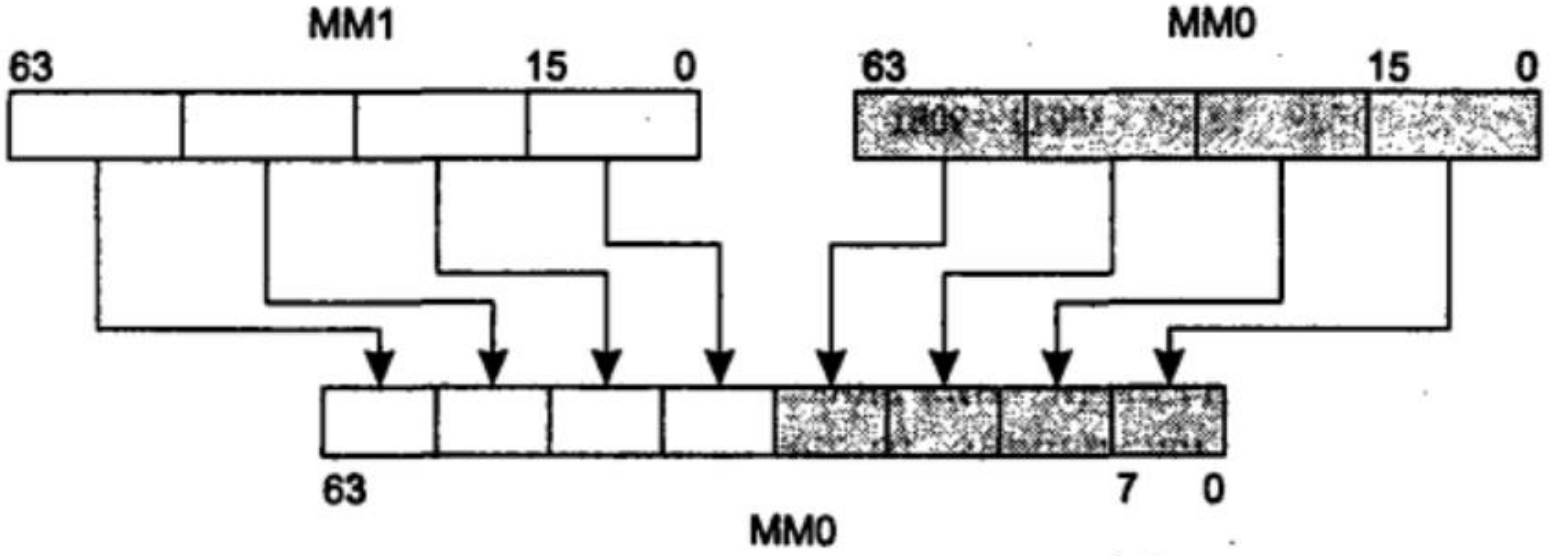
MM0	1609	-13104	-30011	-9081
MM1	27791	-25959	7290	25584
MM0-MM1	-26182	12855	-32768	-32768

КОМАНДЫ УПАКОВКИ ДАННЫХ

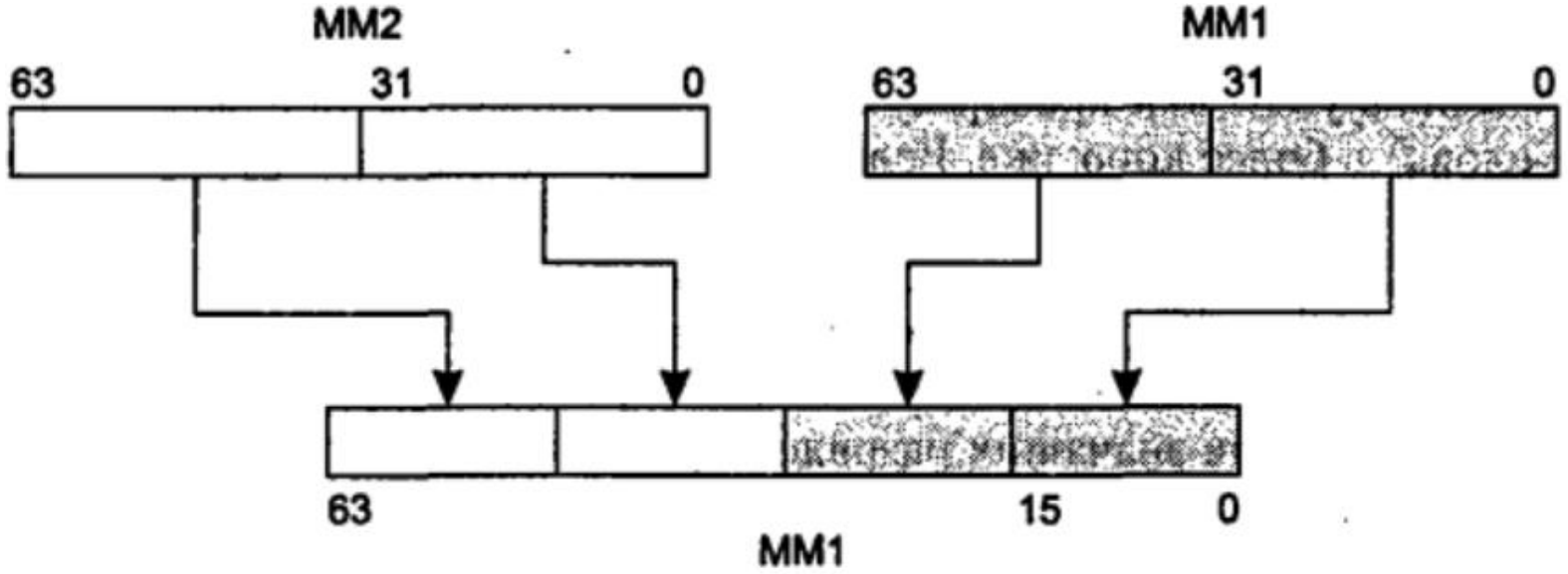
- Преобразуют длинные элементы данных (16 и 32 разряда) в более короткие (8 и 16 разрядов).
- Преобразование с насыщением: если значение «не помещается» в коротком элементе, результатом будет граничное значение короткого элемента.
- Первый операнд – регистр MMX.
- Второй операнд – регистр MMX или ячейка памяти.
- **packsswb, packssdw** — преобразование длинных элементов данных (16- и 32-разрядных слов со знаком) в более короткие (байты или 16-разрядные слова со знаком).
- **packuswb** —преобразовать 16-разрядных слов со знаком из обо-обоих операндов в байты без знака и записать их в выходной операнд.

КОМАНДЫ УПАКОВКИ ДАННЫХ

packsswb MM0, MM1

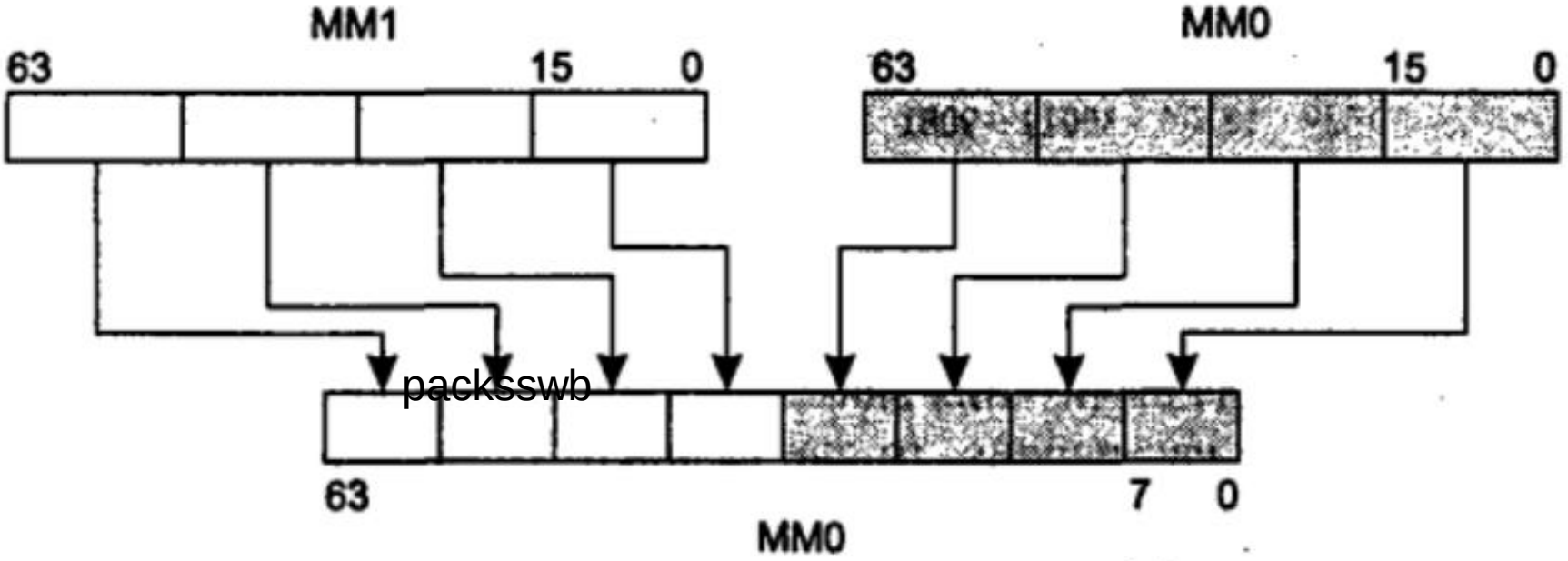


packssdw MM0, MM1



КОМАНДЫ УПАКОВКИ ДАННЫХ

packuswb MM0, MM1



Пример. Упаковка массива слов в массив байт командой packsswb

ПРИМЕР УПАКОВКИ ДАННЫХ

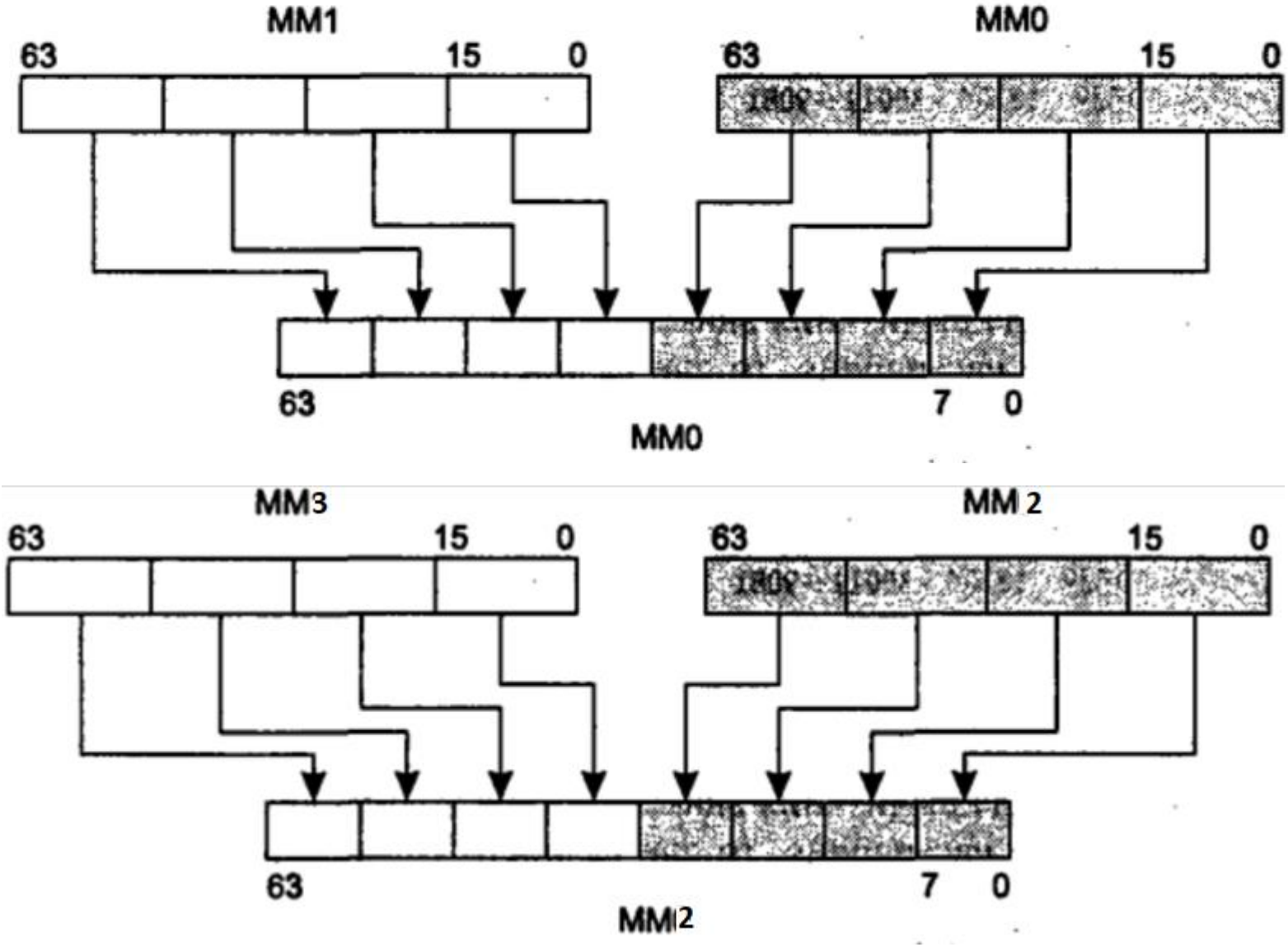
```
.686
.model flat
.MMX
option casemap:none
.data
a1 DW 45, -41, 67, -134, -61, 10, -88, 12, -62, 161,-99
Len EQU $-a1                ; длина массива в байтах
res DB len DUP(0)           ; буфер для упакованных данных
.code
_packsswb_ex proc
mov EAX, len
shr EAX, 1                  ; кол-во слов в исходном массиве
xor EDX, EDX
mov EBX, 8
div EBX             ; сколько пакетов по 8 слов в исходном массиве?
mov ECX, EAX                ; счетчик пакетов для распаковки
lea ESI, al         ; адрес первого упаковываемого пакета в паре
lea EDI. res        ; адрес для записи упакованных данных
```

ПРИМЕР УПАКОВКИ ДАННЫХ

```
next:
movq MM0, qword ptr [ESI]
packsswb MM0, qword ptr [ESI+8]
movq qword ptr [EDI], MM0
add ESI, 16
add EDI, 8
dec ECX
jnz next
cmp EDX, 0
je exit
; обработка «хвоста» из EDX слов (не полная восьмерка)
; …
exit:
lea EAX, res
ret
_packsswb_ex endp
end
```

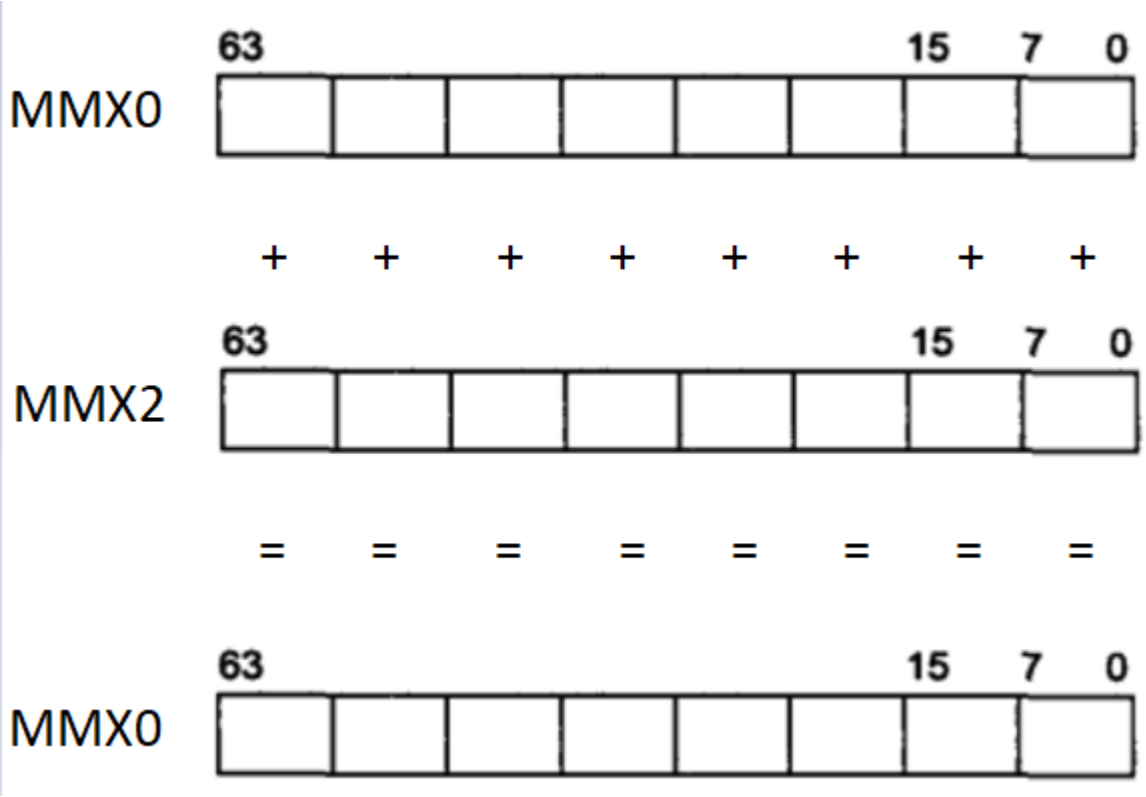
УПАКОВКА И СЛОЖЕНИЕ МАССИВОВ

- Массивы разбиваются на пакеты элементов, которые последовательно упаковываются в MMX регистры.



УПАКОВКА И СЛОЖЕНИЕ МАССИВОВ

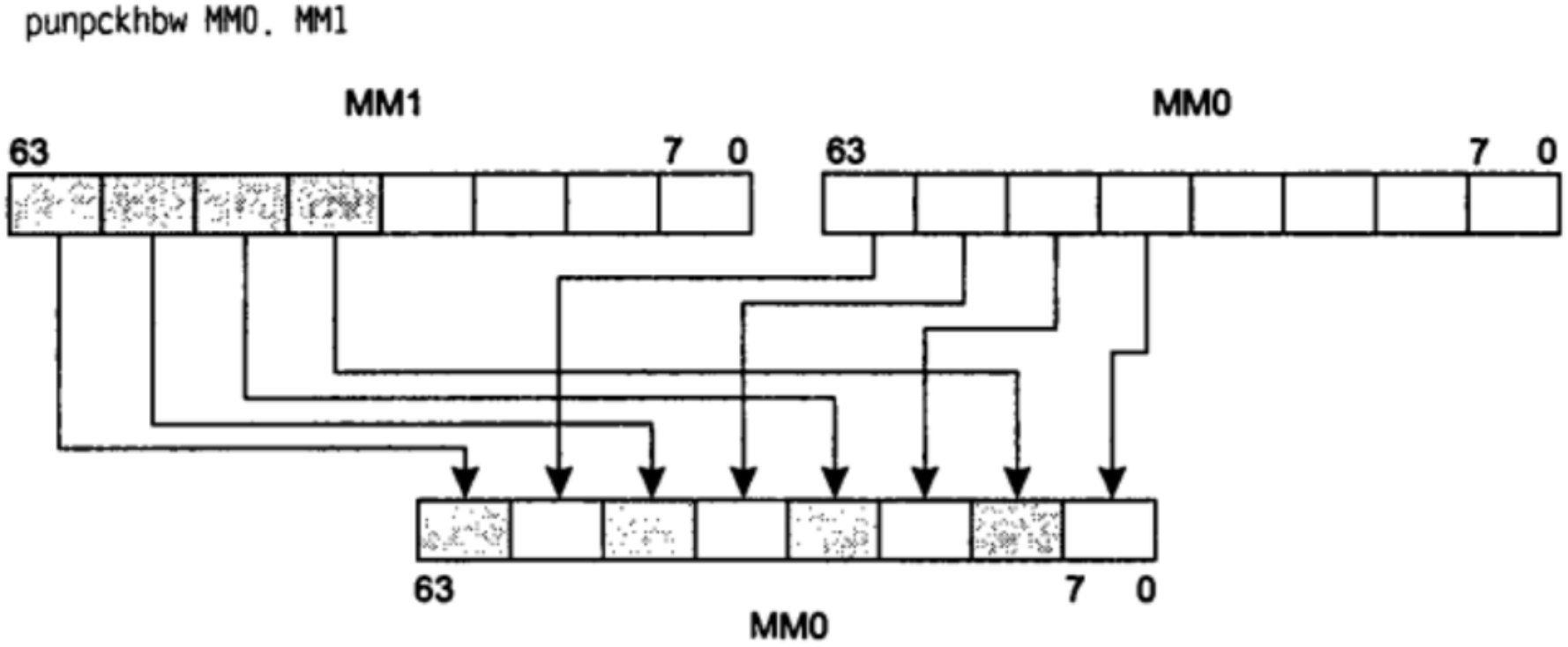
- Упакованные элементы массивов складываются в MMX регистрах и записываются в массив результата.



КОМАНДЫ РАСПАКОВКИ

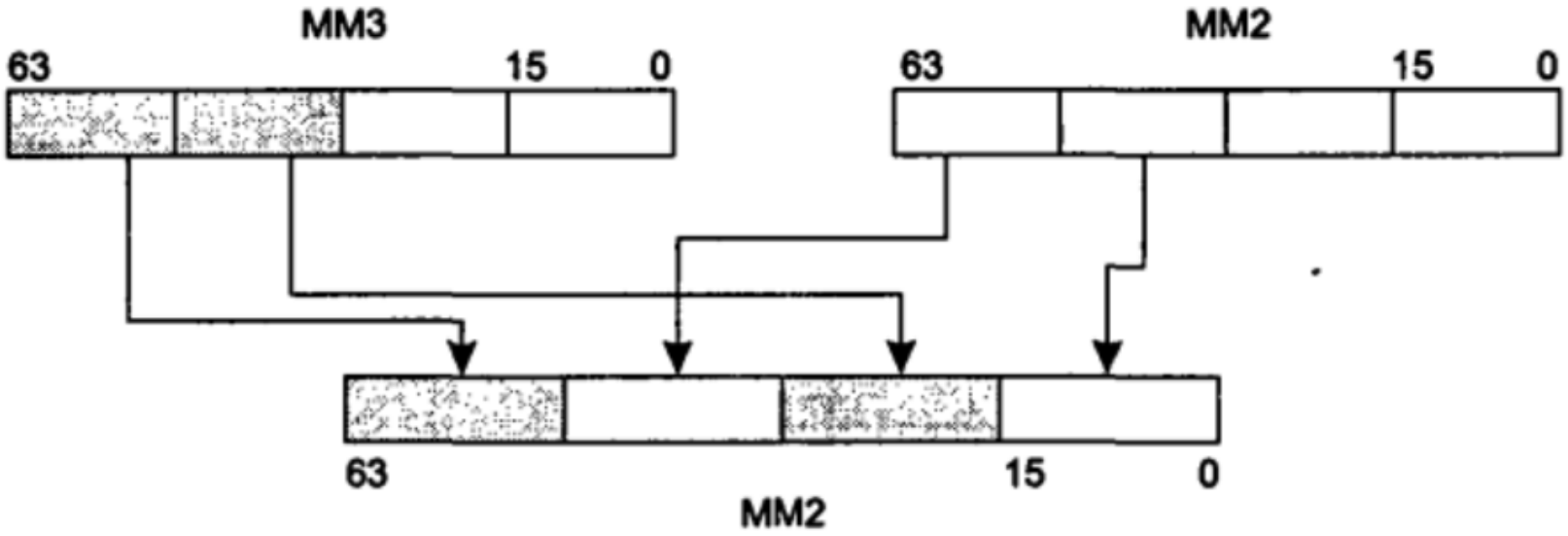
punpckhbw, punpckhwd, punpckhdq

- попарное объединение исходных элементов данных (байтов, 16- или 32-разрядных слов), находившихся в старших 32 разрядах обоих операндов.
- Первый операнд команды – MMX регистр.
- Второй операнд – MMX регистр или ячейка памяти.

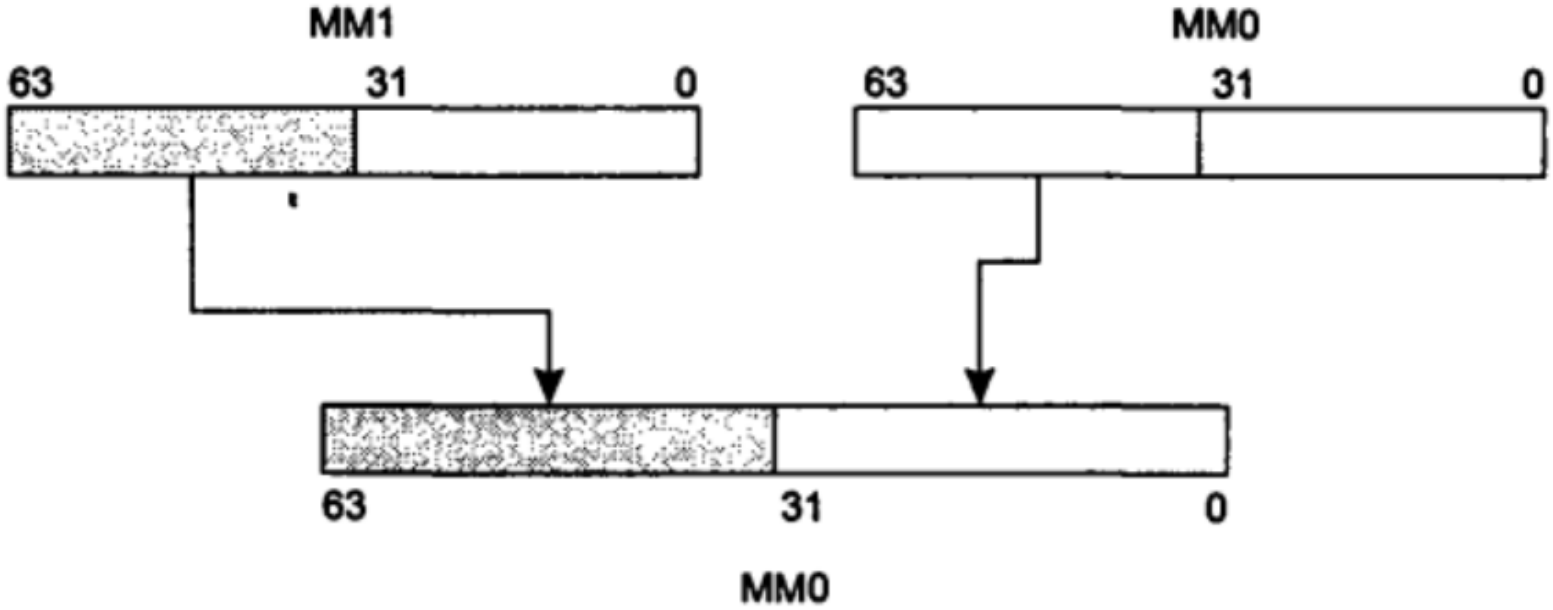


КОМАНДЫ РАСПАКОВКИ

punpckhwd MM2, MM3



punpckhdq MM0, MM1

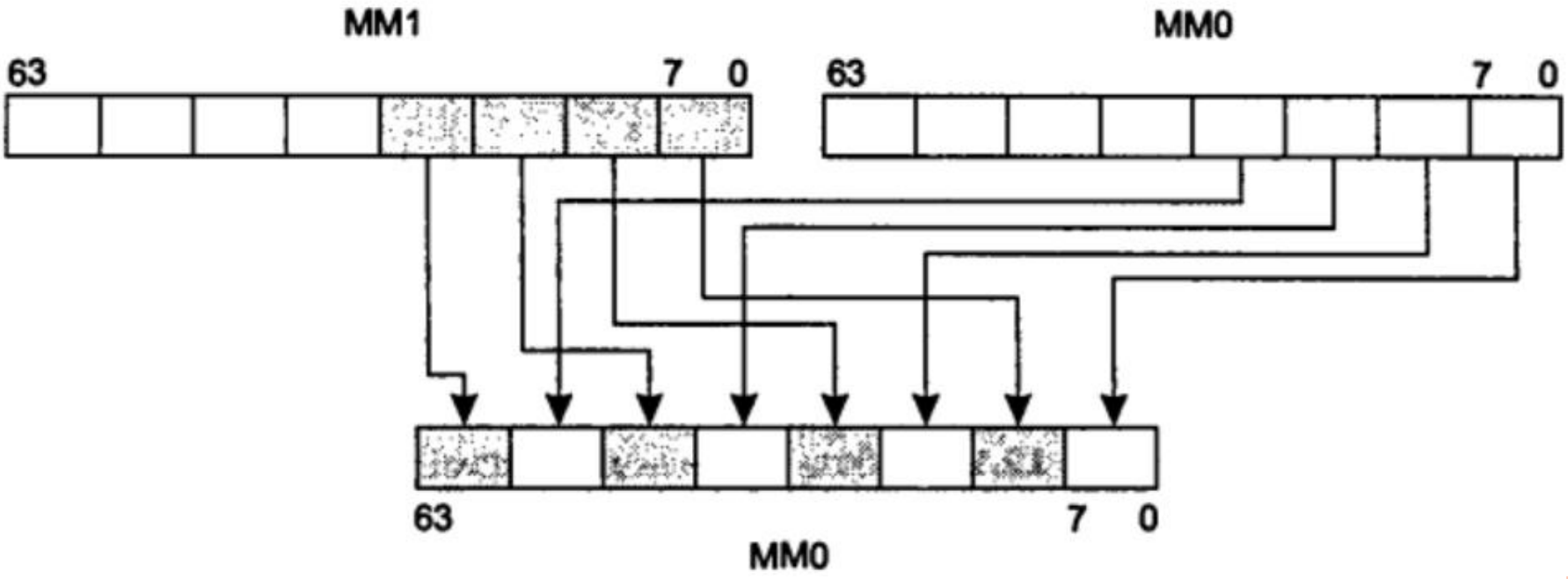


КОМАНДЫ РАСПАКОВКИ

punpcklbw, punpcklwd, punpckldq

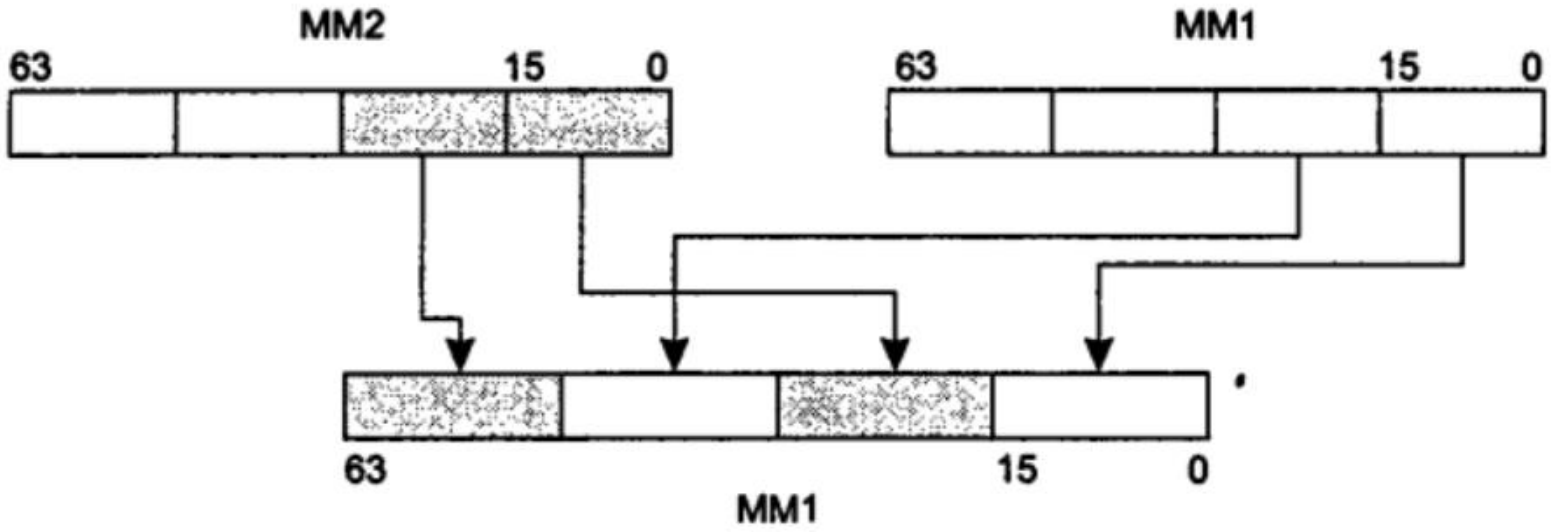
- попарное объединение исходных элементов данных (байтов, 16- или 32-разрядных слов), находившихся в младших 32 разрядах обоих операндов.
- Первый операнд команды – MMX регистр.
- Второй операнд – MMX регистр или ячейка памяти.

punpcklbw MM0, MM1

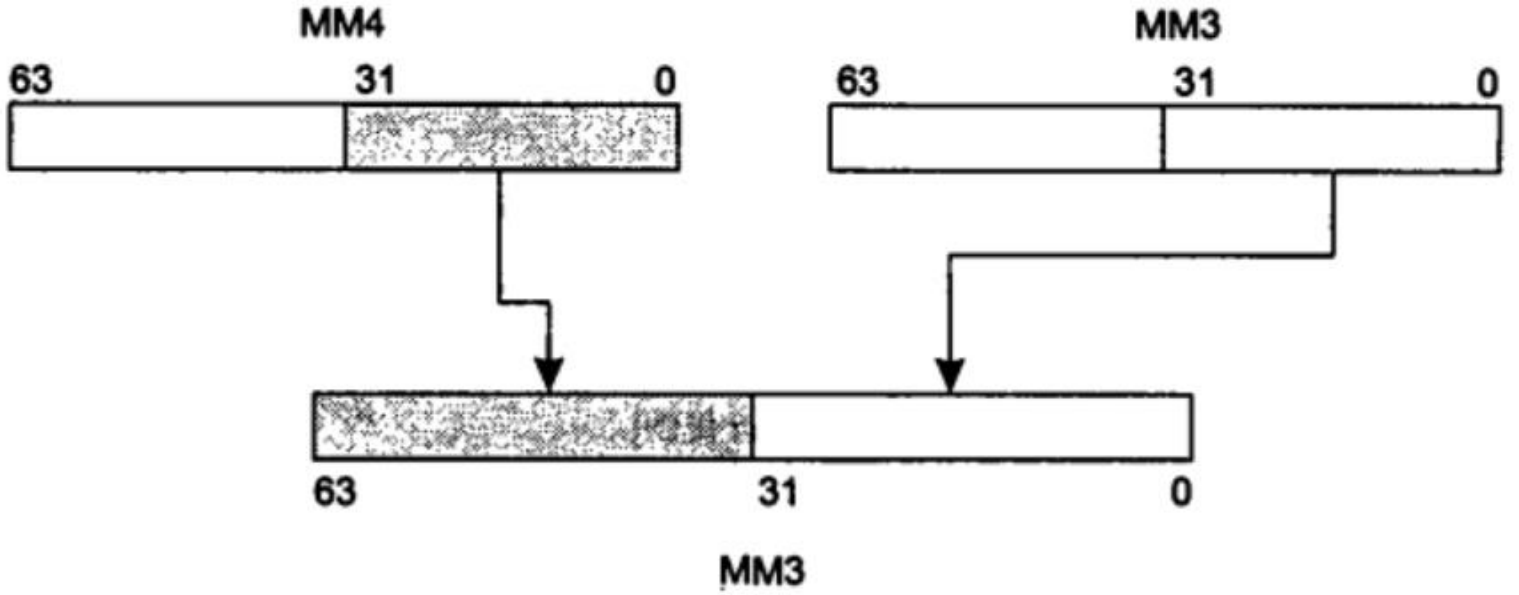


КОМАНДЫ РАСПАКОВКИ

punpcklwd MM1, MM2



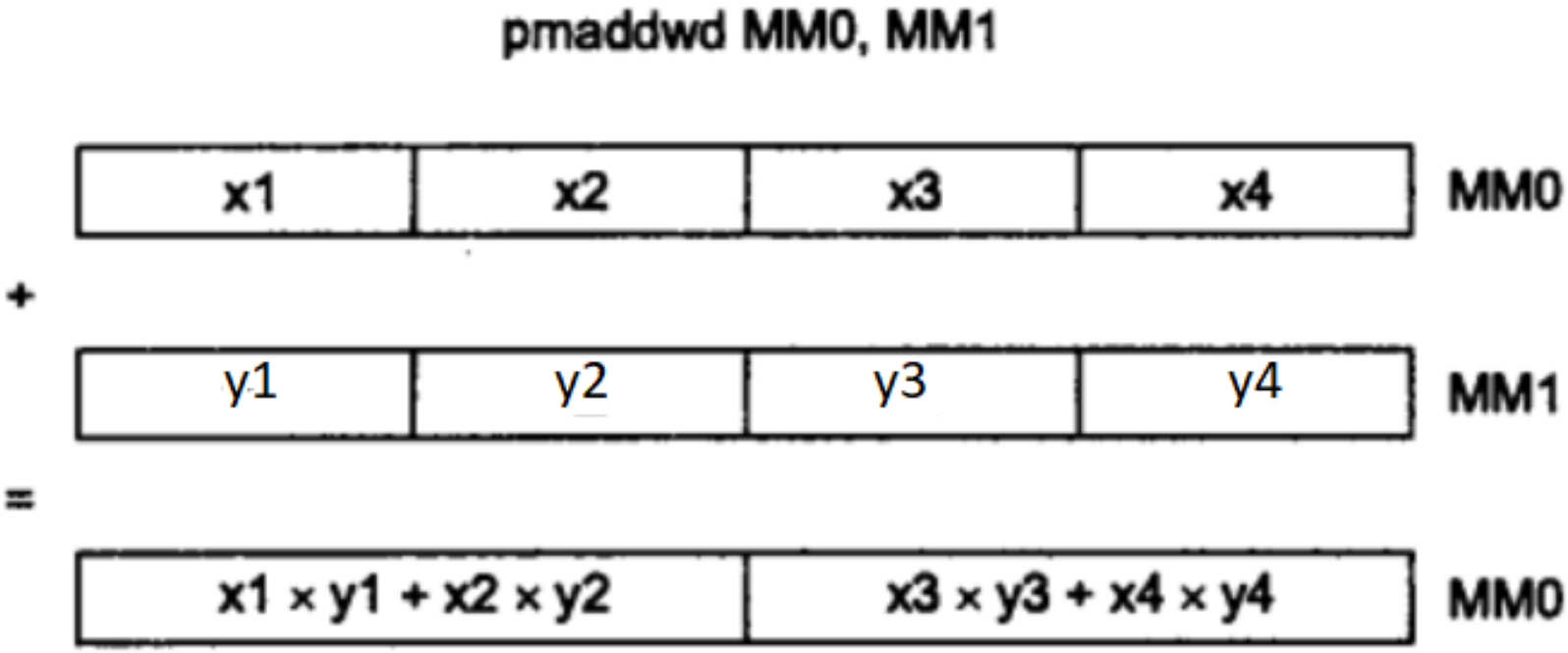
punpckldq MM3, MM4



КОМАНДЫ УМНОЖЕНИЯ

pmaddwd

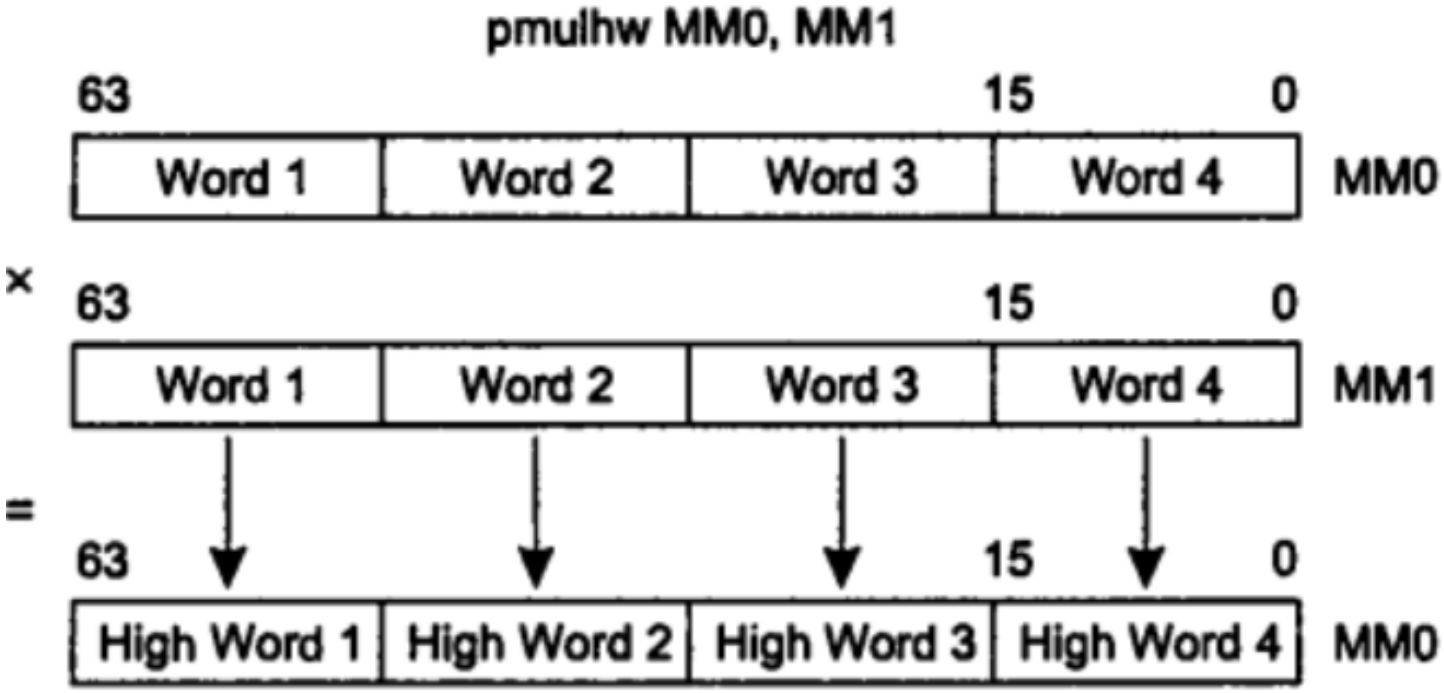
- попарное умножение 16-разрядных слов со знаком, находящихся в двух операндах.
- После получения в результате четырех 32-разрядных произведений первое произведение складывается со вторым, а третье — с четвертым. Суммы записываются в 32-разрядные слова выходного операнда.
- Первый операнд команды – MMX регистр.
- Второй операнд – MMX регистр или ячейка памяти.



КОМАНДЫ УМНОЖЕНИЯ

pmulhw

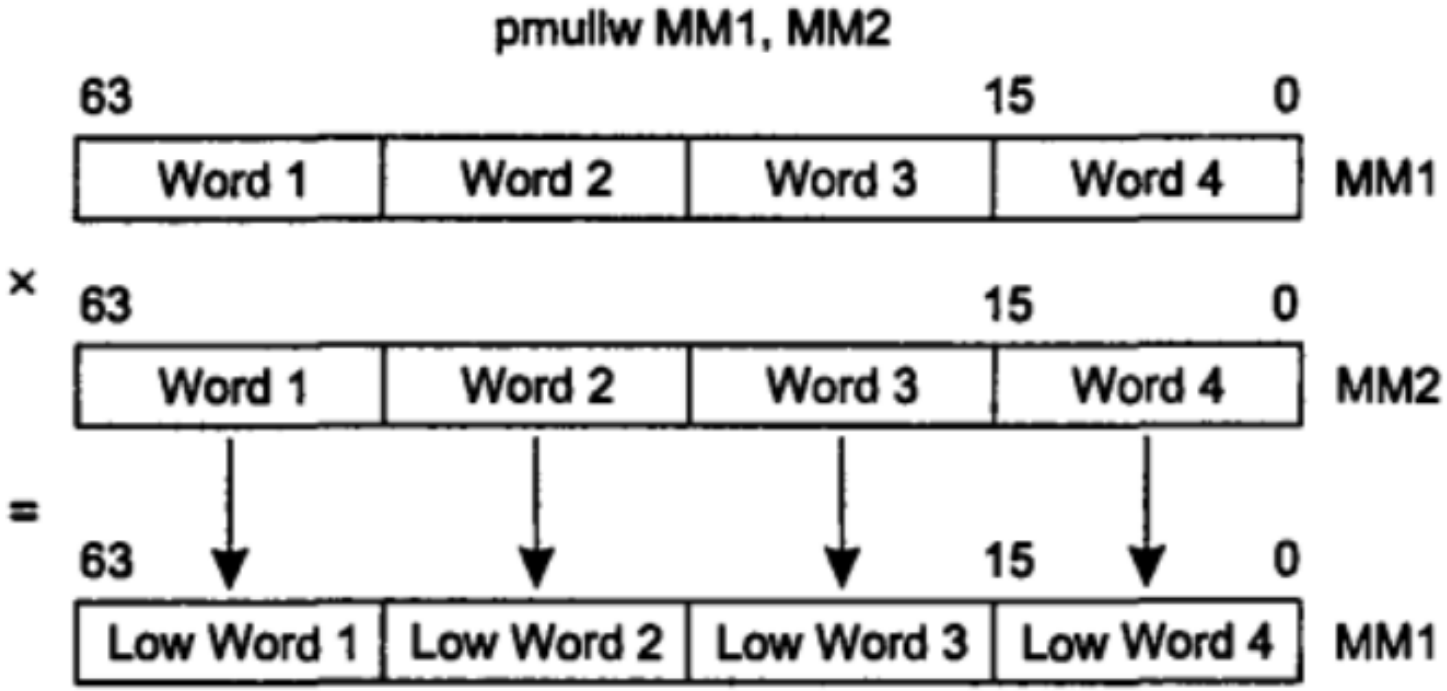
- попарное умножение 16-разрядных слов со знаком, находящихся в двух операндах.
- Результатом операции являются четыре 32-разрядных произведения, при этом старшие разряды произведений сохраняются в 16-разрядных словах выходного операнда, а младшие разряды произведений теряются.
- Первый операнд команды – MMX регистр.
- Второй операнд – MMX регистр или ячейка памяти.



КОМАНДЫ УМНОЖЕНИЯ

pmullw

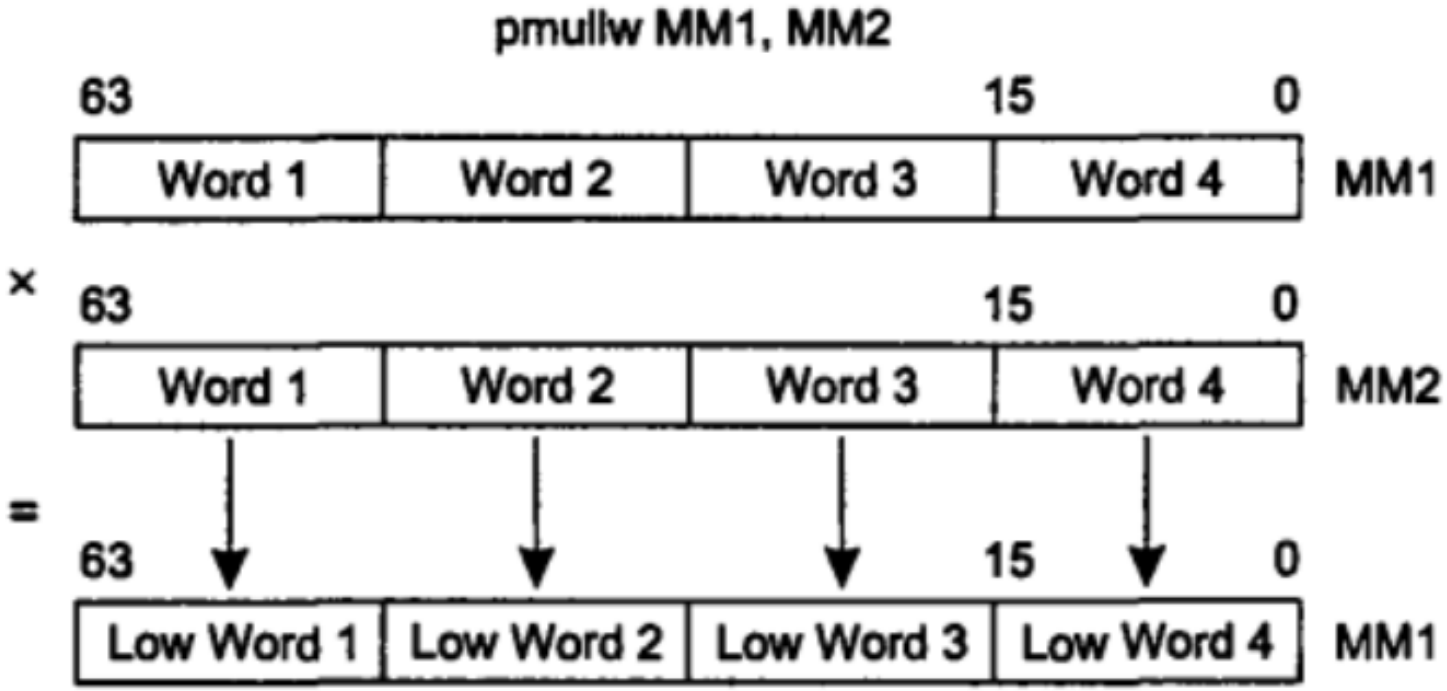
- попарное умножение 16-разрядных слов со знаком, находящихся в двух операндах.
- Результатом операции являются четыре 32-разрядных произведения, при этом младшие разряды произведений сохраняются в 16-разрядных словах выходного операнда, а младшие разряды произведений теряются.
- Первый операнд команды – MMX регистр.
- Второй операнд – MMX регистр или ячейка памяти.



КОМАНДЫ УМНОЖЕНИЯ

pmullw

- попарное умножение 16-разрядных слов со знаком, находящихся в двух операндах.
- Результатом операции являются четыре 32-разрядных произведения, при этом младшие разряды произведений сохраняются в 16-разрядных словах выходного операнда, а младшие разряды произведений теряются.
- Первый операнд команды – MMX регистр.
- Второй операнд – MMX регистр или ячейка памяти.



ПОЛНЫЙ РЕЗУЛЬТАТ УМНОЖЕНИЯ

Параллельное умножение 4 16-разрядных чисел

- Получить старшие 16 бит произведения, используя команду pmulhw.
- Получить младшие 16 бит произведения, используя команду pmullw.
- Объединить частичные результаты в одно двойное слово с помощью команд punpckhwd и punpcklwd.

Пример. Исходные числа – в регистрах MM0 и MM1.

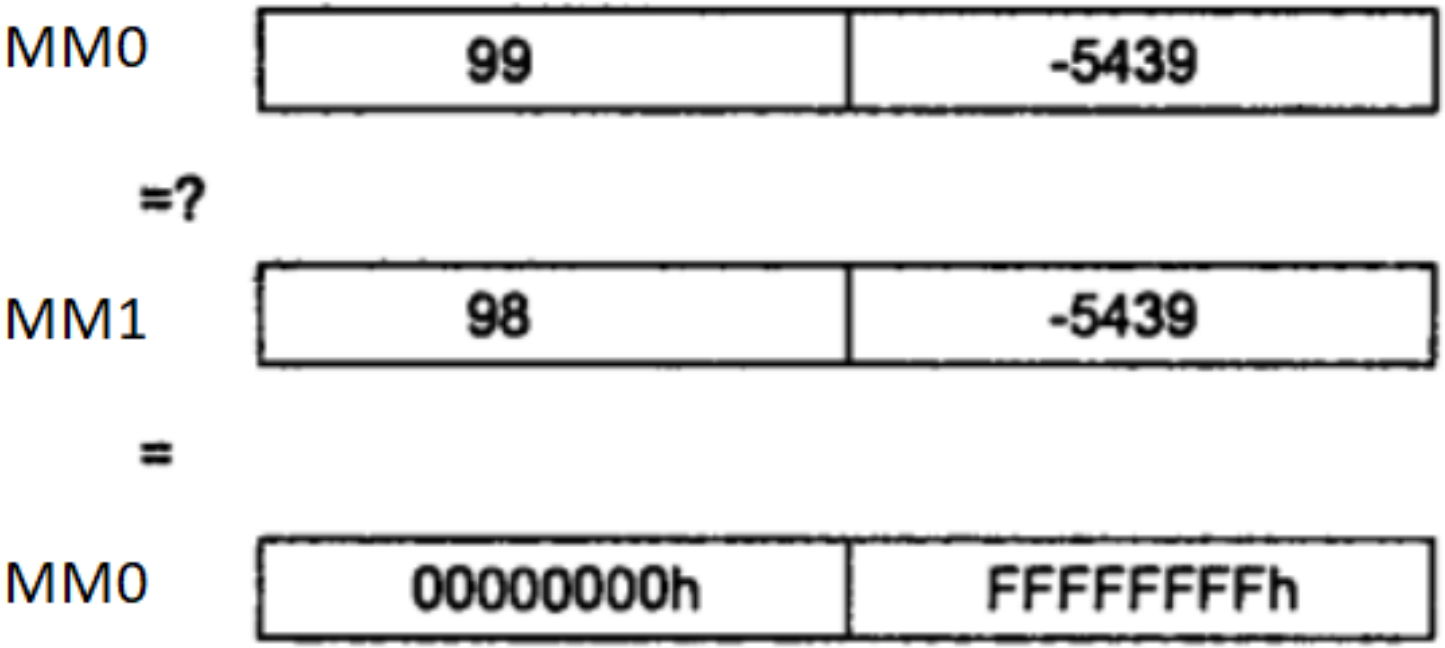
```
movq MM2, MM0          ; копия в MM2
pmullhw MM0, MM1       ; MM0 – старшие слова произведений
pmulllw MM1,MM2        ; MM1 – младшие слова произведений
movq MM2, MM0          ; копии фрагментов произведений
movq MM3, MM1
punpcklwd MM1, MM0 ; в MM1 два первых произведения
 punpckhwd MM3, MM2 ; в MM3 два оставшихся произв.
```

КОМАНДЫ СРАВНЕНИЯ

- Попарно сравнивают элементы данных (байты, 16- или 32-разрядные слова) приемника и источника.
- Приемник – регистр ММХ.
- Источник – регистр ММХ или ячейка памяти.
- Если сравнение пары элементов данных имеет результат «истина», соответствующий элемент данных приемника заполняется единицами, в противном случае – нулями.
- pcmpeqb, pcmpeqw, pcmpeqd – попарное сравнение на равенство байтов, слов или двойных слов, упакованных в приемнике и источнике. Если пара имеет равное значение, ее элемент в приемнике заполняется единицами.В противном случае – нулями.
- pcmpgtb, pcmpgtw, pcmpgtd - попарное сравнение на «больше» байтов, слов или двойных слов, упакованных в приемнике и источнике. Если элемент в приемнике больше элемента в источнике, он заполняется единицами. В противном случае – нулями.

ПРИМЕР СРАВНЕНИЯ

pcmpeqd MM0, MM1



ЛОГИЧЕСКИЕ КОМАНДЫ

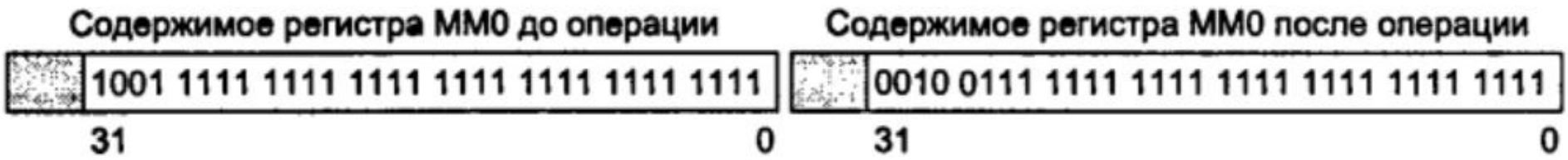
- Выполняют поразрядные логические операции над всеми 64 битами приемника и источника.
- Приемник – MMX регистр.
- Источник – MMX регистр или ячейка памяти.
- pand – поразрядное логическое И приемника и источника.
- pandn – поразрядное логическое И приемника и инверсии источника.
- por – поразрядное логическое ИЛИ приемника и источника.
- pxor – поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ приемника и источника

КОМАНДЫ СДВИГА

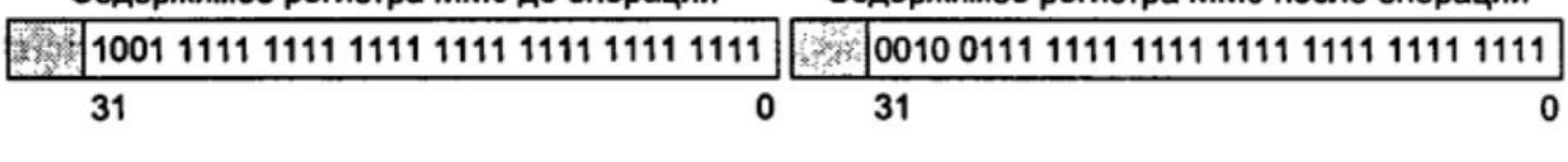
- Сдвиг каждого элемента данных (16-, 32- или 64-разрядного слова) в приемнике на количество бит, задаваемое в источнике.
- Приемник – регистр MMX.
- Источник - регистр MMX, ячейка памяти, непосредственное значение.
- psllw, pslld, psllq – логический сдвиг 16-, 32- или 64-разрядных слов влево.
- psrlw, psrld, psrlq – логический сдвиг 16-, 32- или 64-разрядных слов вправо.
- psraw, psrad – арифметический сдвиг 16- или 32-разрядных слов вправо.

КОМАНДЫ СДВИГА

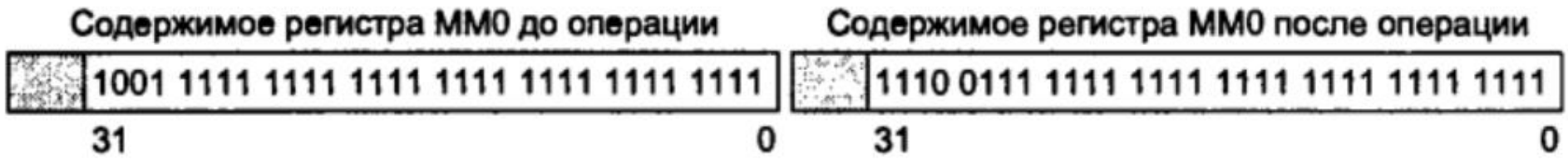
PSLLD MM0, 2 при MM0 = 9FFFFFFFh



PSRLD MM0, 2 при MM0 = 9FFFFFFFh



psrad MM0, 2 при MM0 = 9FFFFFFFh



ДОПОНИТЕЛЬНЫЕ КОМАНДЫ

- pavgb, pavdw - вычисляют среднее значение двух беззнаковых чисел, упакованных в байты или слова.
- pextrw – копирует слово, номер которого задан третьим операндом команды, из источника (MMX регистр) в младшее слово приемника (32-битный РОН).
- pinsrw — вставляет слово, номер которого задан третьим операндом команды, из источника (младшее слово 32-битного РОН) в приемник (MMX регистр).
- pmaxub, pmaxsw — извлекают максимальное значение из каждой пары упакованных элементов (беззнаковые байты/ знаковые слова) и помещает в соответствующий элемент приемника.
- pminub, pminsw — извлекают минимальное значение из каждой пары упакованных элементов (беззнаковые байты/ знаковые слова) и помещает в соответствующий элемент приемника.
- pmovmskb — формирует байт, содержащий знаковые биты восьми байтов, упакованных в источнике (MMX регистр). Приемником является 32-разрядный РОН, младший байт которого будет содержать результат. Эта команда удобна для формирования условных ветвлений в программах;

ДОПОНИТЕЛЬНЫЕ КОМАНДЫ

- psadbw — вычисляет сумму разностей значений беззнаковых байтов, упакованных в приемнике и источнике. Результат помещается в младший байт приемника (остальные байты приемника обнуляются).