Для чего нужна технология MMX

Прикладные программы все чаще 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.

От применения технологии MMX выигрывают именно такие программы.

Особенности технологии MMX

В основе технологии MMX лежит расширение набора команд процессора с учетом потребностей современных мультимедиа-программ.

Команды технологии MMX работают с новыми типами данных: 64-разрядными целочисленными данными, а также с данными, упакованными в группы (векторы) общей длиной 64 бита. Такие данные могут находиться в памяти или в восьми MMX-регистрах. Эти регистры называются `MM0`, `MM1`, ..., `MM7`.

Одна команда - много данных

В технологии MMX использована модель обработки данных SIMD (single instruction, multiple data, т.е. одна команда – много данных). Это повышает производительность программ, поскольку одна команда обрабатывает несколько элементов данных одновременно. MMX-команды обеспечивают параллельную обработку нескольких байтов, слов или двойных слов.

Пример, моделирующий работу двух графических программ. Если первая программа написана без MMX-команд, то она обрабатывает лишь одну точку цветного изображения (8 бит) за одну команду. Во второй программе применена технология MMX; ее команды способны обрабатывать 8 точек (64 бит) одновременно!

Типы данных

Команды технологии MMX работают со следующими типами данных:

- упакованные байты (восемь байтов в одном 64-разрядном регистре)
- упакованные слова (четыре 16-разрядных слова в 64-разрядном регисте)
- упакованные двойные слова (два 32-разрядных слова в 64-разрядном регистре)
- 64-разрядные слова.

По-английски эти типы данных называются, соответственно, packed byte, packed word, packed doubleword и quadword.

Синтаксис команд

MMX-команды имеют следующий синтаксис:

```
instruction [dest,src]
```

Здесь *instruction* – имя команды,
dest обозначает выходной операнд, *src* – входной операнд.
Большинство команд имеют суффикс, который определяет тип данных и используемую арифметику:

- US (unsigned saturation) – арифметика с насыщением, данные без знака.
- S или SS (signed saturation) – арифметика с насыщением, данные со знаком. Если в суффиксе нет ни S, ни SS, используется циклическая арифметика (wraparound).
- B, W, D, Q указывают тип данных. Если в суффиксе есть две из этих букв, первая соответствует входному операнду, а вторая – выходному.

Например, MMX-команда `paddusw MM4,mem1` выполняет сложение слов без знака. Первые слагаемые находятся в MMX-регистре MM4, вторые – в памяти по адресу mem1. Суммы записываются в регистр MM4.

Результат вне допустимого диапазона

Большинство команд технологии MMX обрабатывают данные в циклической арифметике, а некоторые команды используют арифметику с насыщением.
Циклическая арифметика. Если результат операции выходит за пределы допустимого диапазона, то "лишние" старшие биты результата отбрасываются. Например, сложение байтов 01h и FFh дает 00h.
Арифметика с насыщением. Если результат оказался вне допустимого диапазона, то он считается равным граничному значению диапазона. Например, при сложении байтов без знака сумма 01h и FFh считается равной FFh.

Данные со знаком и без знака

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 допустимый диапазон. Следовательно, если используется арифметика с насыщением, то при выходе результата операции за пределы допустимого диапазона в выходной операнд будут записаны различные значения в зависимости от типа данных. Например, если результат превысил 7FFFh, слово со знаком будет считаться равным 7FFFh, а слово без знака – нет.

Пример: при сложении слов со знаком 7F38h (32568) и 1707h (5895) слово-результат считается равным 7FFFh (32767). Математическая сумма оказалась больше предела 7FFFh, и произошло "насыщение". Если те же самые два значения сложить как слова без знака, то получится 963Fh (38463). В данном случае "насыщения" не происходит, так как результат меньше максимально возможного FFFFh.

Типы данных

MMX-команды используют новые типы данных: упакованные байты (packed byte, суффикс команды **b**), упакованные слова (packed word, суффикс **w**), упакованные двойные слова (packed doubleword, суффикс **d**) и 64-разрядные слова (quadword, суффикс **q**).

Одни и те же 64 бита могут трактоваться одной MMX-командой как 8 байт, другой командой - как 4 слова, и т.д. Тип данных определяется суффиксом команды.

Допустимый диапазон данных

тип данных	минимальное значение	максимальное значение
Байт со знаком	80h (-128)	7Fh (127)
Байт без знака	00h	FFh (256)
Слово со знаком	8000h (-32768)	7FFFh (32767)
Слово без знака	0000h	FFFFh (65535)
Двойное слово со знаком	80000000h (-2147483648)	7FFFFFFFh (2147483647)
Двойное слово без знака	00000000h	FFFFFFFFh (4294967295)

Операнд
Большинство MMX-команд имеет два операнда: выходной и входной. Обычно входная информация берется из обоих операндов; результат записывается в выходной операнд.

Пример: `pand MM2,MM4`
Здесь регистр `MM2` - выходной операнд, регистр `MM4` - входной. Эта команда вычисляет поразрядное логическое И между значениями в регистрах `MM2` и `MM4` и записывает результат в регистр `MM2`.

Арифметика с насыщением
(saturation arithmetic)
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 оказался меньше нижней границы допустимого диапазона, то в выходной операнд записывается минимально возможное значение.
Например, если результат меньше 8000h, то 16-разрядное слово со знаком считается равным 8000h; если получилось больше 7FFFh, то слово со знаком считается равным 7FFFh.

Циклическая арифметика (wraparound arithmetic)
Если команда использует циклическую арифметику и результат операции выходит за двоичную разрядную сетку используемого типа данных, то "лишние" старшие биты результата отбрасываются. Иначе говоря, если результат превышает максимально возможное значение на **n** единиц, то результатом считается **минимальное значение+n-1**.

Примеры:
```
7FFFh + 0002h = 8001h
(32767 + 2 = -32767 или 32769)
FFFFh + 0002h = 0001h
(65535 + 2 = 1 или -1 + 2 = 1)
```

Команда EMMS
Команда EMMS присваивает значение 1 всем разрядам слова состояния регистров с плавающей запятой, что соответствует состоянию Empty. Это обеспечивает переход процессора из режима исполнения MMX-команд в режим исполнения обычных команд с плавающей запятой.

Не забывайте ставить команду EMMS в конце процедур, использующих MMX-команды!

Команды сложения и вычитания
MMX-команды сложения и вычитания работают с упакованными байтами, словами и двойными словами. Обрабатываются данные как со знаком, так и без знака.

Команды сложения	Команды вычитания
paddb/w/d	psubb/w/d
paddsb/w	psubsb/w
paddusb/w	psubusb/w

Команды сдвига
MMX-команды сдвига выполняют арифметический и логический сдвиг разрядов в элементах данных выходного операнда. При логическом сдвиге освободившиеся разряды заполняются нулями. При арифметическом сдвиге крайний из сдвигаемых битов "размножается" и заполняет все освободившиеся разряды.

Команды сдвига
psllw/d/q
psraw/d
psrlw/d/q

Логические команды
Эта группа MMX-команд выполняет поразрядные логические операции над 64 битами данных.

Логические команды	
pand	por
pandn	pxor

Команды умножения
MMX-команды умножения вычисляют произведения 16-разрядных слов своих операндов. Команда `pmulhw` записывает в каждое слово выходного операнда старшие 16 разрядов этих произведений, а команда `pmullw` – младшие 16 разрядов. Команда `pmaddwd` попарно складывает произведения и записывает полученные суммы в 32-разрядные двойные слова выходного операнда.

Команды умножения
pmulhw
pmullw
pmaddwd

Команды сравнения
Эти MMX-команды сравнивают элементы данных в операндах и генерируют маску в выходном операнде. MMX-команды сравнения не устанавливают флагов (признаков).

Команды сравнения

```
pcmpeqb/w/d
pcmpgtb/w/d
```

Команды упаковки и распаковки
MMX-команды упаковки "укорачивают" элементы данных в своих операндах и записывают полученные короткие данные в выходной операнд. MMX-команды распаковки составляют из элементов данных в своих операндах новые элементы данных удвоенной длины и записывают их в выходной операнд.

Команды упаковки	**Команды распаковки**
`packsswb/dw`	`punpckhbw/wd/dq`
`packuswd`	`punpcklbw/wd/dq`

Команды передачи данных
MMX-команды передачи данных выполняют копирование данных из одного MMX-регистра в другой, а также пересылку между MMX-регистрами, целочисленными регистрами и памятью.

Команды передачи данных

```
movd
movq
```

Скалярное произведение векторов
Эта операция часто используется в приложениях линейной алгебры и цифровой обработки сигналов. Скалярное произведение $a \cdot b$ векторов a_k и b_k ($k = 1,..., n$) вычисляется так:
$a \cdot b = a_1 b_1 + a_2 b_2 + ... + a_n b_n$

Абсолютные величины разностей
Абсолютные величины разностей чисел используются в алгоритмах распознавания и сжатия в качестве метрики – меры "расстояния" между числами.

Условный выбор фрагмента изображения
Выделение фрагмента изображения по какому-либо условию – обычная операция в графических программах, например, при наложении объекта или текста на картинку-фон. Такая операция требует многих ветвлений в программе; их можно избежать, если воспользоваться MMX-командами.

Распаковка с "размножением" знака
Распаковка с "размножением" знакового бита часто требуется при преобразовании коротких типов данных к длинным.

Команда movq
Команда `movq` выполняет копирование 64 бит из одного MMX-регистра в другой, а также из MMX-регистра в память и обратно.

Команда pmaddwd
Команда `pmaddwd` попарно перемножает слова (со знаком) входного и выходного операнда, что дает четыре 32-разрядных произведения. Затем первое произведение складывается со вторым, а третье с четвертым. Полученные суммы записываются в 32-разрядные слова выходного операнда.

Команда paddd
Команда `paddd` складывает 32-разрядные слова входного операнда с 32-разрядными словами выходного операнда в циклической арифметике.

Команда pcmpeqb
Команда `pcmpeqb` сравнивает байты входного операнда с соответствующими байтами выходного. Если байт выходного операнда равен байту входного, такой байт заполняется единицами, а если не равен, то он заполняется нулями.

Команда pand
Команда `pand` вычисляет поразрядное логическое И входного и выходного операнда. При выполнении этой команды значение 1 остается только в тех битах выходного операнда, для которых соответствующие биты **обоих** операндов на входе были равны 1.

Команда pandn
Команда `pandn` вычисляет обращение (поразрядное логическое НЕ) выходного операнда, а затем поразрядное логическое И между входным операндом и обращенным значением выходного.

Команда por
Команда `por` вычисляет поразрядное логическое ИЛИ входного и выходного операнда. При выполнении этой команды значение 1 записывается в те биты выходного операнда, для которых хотя бы один из соответствующих битов обоих операндов на входе был равен 1. В остальные биты записывается 0.

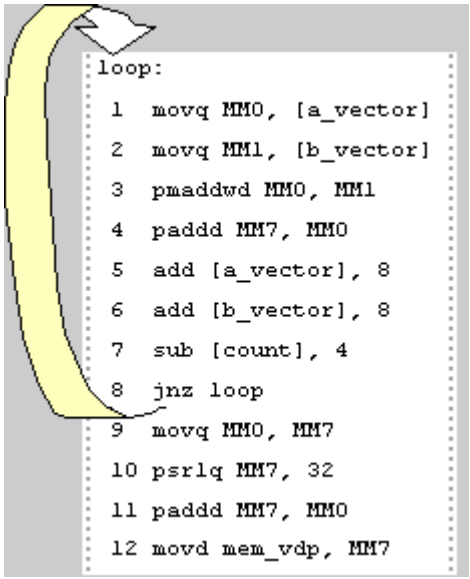
Пример

Скалярное произведение векторов

С помощью MMX-команд можно быстро вычислять скалярное произведение векторов. Пусть векторы состоят из 16-разрядных слов и находятся в памяти по адресам `a_vector` и `b_vector`. Программа будет выполнять следующие шаги:

1. Скопировать 4 слова первого вектора в MMX-регистр командой movq.
2. Скопировать 4 слова второго вектора в MMX-регистр.
3. Попарно перемножить слова и сложить первое произведение со вторым, а третье с четвертым, пользуясь командой pmaddwd.
4. Добавить результаты шага 3 к ранее накопленным суммам командой paddd

Эти шаги следует повторить для всех слов исходных векторов (см. русунок).



Команды 1-4 реализуют шаги 1-4 (см. рис.)
Команды 5-8 организуют цикл по всем элементам векторов.
Команды 9-11 складывают частичные суммы, накопленные в цикле в старших и младших разрядах регистра MM7.
Команда 12 пересылает результат в память.