ТЕХНОЛОГИЯ MMX

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Назначение MMX

Технология Intel MMX представляет собой набор расширений к архитектуре Intel, которые были разработаны для того, чтобы увеличить производительность средств мультимедиа и коммуникаций.

Расширение MMX предназначено для ускорения выполнения приложений типа «подвижное видео», комбинированной графики с видеообработкой изображений, звуковым синтезом, синтезом и сжатием речи, телефонией, видео, конференц-связью, и 2D и 3D графикой, которые обычно используют алгоритмы с интенсивными вычислениями, чтобы выполнять повторяющиеся действия на больших множествах простых элементов данных.

Технология MMX определяет простую и гибкую модель программного обеспечения без нового режима или видимого состояния для операционной системы.

Следующие разделы этой работы описывают основную окружающую среду программирования технологии MMX, включая набор регистров MMX, типы данных и набор инструкций.

Модель SIMD

MMX-технология использует методику «одиночная команда, множественные данные» (Single Instruction Multiple Data – SIMD) для выполнения арифметических и логических операций над байтами, словами или двойными словами, упакованными в 64-разрядные регистры MMX. Например, команда paddsb складывает 8 знаковых байт источника с 8 знаковыми байтами в приемнике и сохраняет их в операнде-приемнике. Эта технология ускоряет эффективность выполнения программ, позволяя одну и ту же операцию выполнять параллельно на множестве элементов данных.

Модель выполнения SIMD, обеспечиваемая в MMX-технологии, удовлетворяет потребностям современных средств связи и графических приложений, которые часто используют сложные алгоритмы, в которых выполняются одни и те же операции над большим количеством данных. Например, большие звуковые данные представляются в 16-разрядных словах. Команды MMX могут обрабатывать сразу 4 из этих слов одновременно в одной команде. Видео- и графическая информация обычно представляются как пакетированные байты. Одна MMX-команда может оперировать над 8 из этих байтов одновременно.

Программная модель расширения MMX

Состав программной модели

Технология MMX обеспечивает следующие новые расширения к окружающей среде программирования архитектуры IA-32:

- восемь 64-разрядных MMX-регистров MM0-MM7;
- четыре типа данных MMX (упакованные байты, слова, двойные слова и учетверенное слово);
- систему команд MMX.

Регистры MMX

Набор регистров MMX состоит из восьми 64-разрядных регистров MM0-MM7. Эти регистры могут использоваться только для выполнения вычислений над MMX типами данных и не могут использоваться для адресации памяти. Адресация операндов MMX-команды в памяти осуществляется, используя стандартные способы адресации и регистры общего назначения. Хотя регистры MMX определены в архитектуре IA-32 как отдельные регистры, они являются псевдонимами младших 64-разрядных частей регистров R0..R7 стека математического сопроцессора как изображено на рисунке 1.

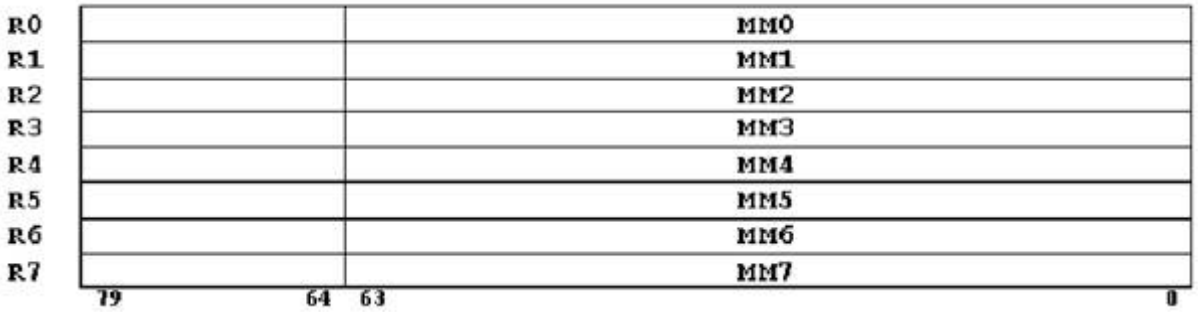

Рисунок 1 – Регистры MMX и регистры сопроцессора

Когда регистры сопроцессора играют роль MMX-регистров, то доступными являются лишь их младшие 64 бита. К тому же, при работе стека сопроцессора в режиме MMX-расширения, он рассматривается не как стек, а как обычный регистровый массив с произвольным доступом. Регистровый стек сопроцессора не может одновременно использоваться и по своему прямому назначению и как MMX-расширение. Забота о его разделении и корректной работе с ним полностью ложится на программиста.

Типы данных MMX

Технология MMX определяет следующие новые 64-разрядные типы данных (рисунок 2):

- упакованные байты – восемь байт, упакованные в одно 64-разрядное поле;
- упакованные слова – четыре слова, упакованные в одно 64-разрядное поле;
- упакованные двойные слова – два двойных слова, упакованные в одно 64-разрядное поле;
- учетверенное слово – одно 64-разрядное поле.

Рисунок 2 – Типы данных MMX

При выполнении арифметических или логических операций над упакованными типами данных MMX-команды оперируют параллельно над индивидуальными байтами, словами, или двойными словами, содержащихся в 64-разрядном MMX-регистре. При операциях над байтами, словами и двойными

словами внутри упакованных типов данных, MMX-команды оперируют как со знаковыми, так и беззнаковыми целыми байтами, словами, двойными словами.

Для определения 64-разрядных данных в программах на ассемблере используется директива dq. Однако существует множество других способов определения MMX-данных.

Например:

```
.data
m64_1    dq       0011223344556677h
m64_2    label    qword
_2_dword    dd    00112233h, 44556677h
m64_3    label    qword
_4_word  dw       0011h, 2233h, 4455h, 6677h
m64_4    label    qword
_8_byte  db       00h, 11h, 22h, 33h, 44h, 55h, 66h, 77h
union mmxdata
 m64     dq       0
 m32     dd       2 dup(0)
 m16     dw           4 dup(0)
 m8      db       8 dup(0)
ends
m64_5        mmxdata  <-1, 2, 3, 4, -29, 100, 84, 53>
```

Система команд MMX-расширения

Арифметика с насыщением и арифметика цикличности

Технология MMX поддерживает новую арифметическую возможность, известную как *арифметика с насыщением* (Saturated Arithmetics). Арифметику с насыщением лучше всего определить, противопоставляя ее *арифметике цикличности* (Wraparound Arithmetic). В арифметике цикличности результаты, которые *переполняются* или *антипереполняются*, усекаются и возвращаются только самые младшие биты результата (только те которые входят в разрядную сетку соответствующего типа), т.е. перенос игнорируется. В режиме насыщения результаты операции, которые переполняются или антипереполняются, приводятся к соответствующим значениям границ диапазона для данного типа данных (таблица 1). Результат операции, который превышает верхнюю границу диапазона типа данных, насыщается к максимальному значению диапазона, а результат, который оказывается меньше нижней границы, – к минимальному значению диапазона. Этот метод обработки переполнения и антипереполнения применяется во многих приложениях.

Например, когда результат превышает диапазон данных для знаковых байтов, он обрезается до 7fh для знаковых байтов и 0ffh для байтов без знака. Если значение меньше диапазона, оно обрезается до 80h для знаковых байтов и 00h для байтов без знака.

Таблица 3.1

Тип данных	Нижний предел		Верхний предел	
	шестнадцат.	десятичн.	шестнадцат.	десатичн.
Знаковый байт	80h	-128	7fh	127
Знаковое слово	8000h	-32768	7fffh	32767
Беззнаковый байт	00h	0	0ffh	255
Беззнаковое слово	0000h	0	0ffffh	65535

Система команд

Система команд MMX состоит из 57 команд, сгруппированных в следующие категории:

- команды пересылки данных;
- арифметические команды;
- команды сравнения;
- логические команды;
- команды сдвига;
- команды упаковки и распаковки;
- дополнительные команды;
- команда инициализации.

При оперировании над упакованными данными внутри регистра MMX данные приводятся в соответствии с типом, определенным командой. Например, команда paddb (сложить упакованные байты) обрабатывает упакованные данные как 8 упакованных байтов, в то время как команда paddw (сложить упакованные слова) обрабатывает упакованные данные как 4 упакованных слова.

Операнды команд

Все команды MMX, за исключением команды emms, оперируют двумя операндами: первый операнд – приемник, второй операнд – источник. Результат команды записывает в операнд-приемник.

Операнд-источник для всех MMX-команд (за исключением команд передачи данных) может быть в памяти или в одном из регистров MM0-MM7. Операнд-приемник всегда должен находиться в регистре MMX. Для команд передачи данных операндом-источником и операндом-приемником может также быть целочисленный регистр (для команды movd) или память (для команд movd и movq).

Команды пересылки данных

MMX-команды пересылки данных работают с 32- и 64-разрядными операндами. В данную группу входят следующие команды:

- movd dst, src – пересылает 32-разрядные данные из памяти в регистры MMX и обратно или из целочисленных регистров процессора в регистры MMX и обратно.
- movq dst,src – пересылает 64-разрядные упакованные данные из памяти в регистры MMX и обратно или между регистрами MMX.

Арифметические команды

Упакованное сложение и вычитание выполняют следующие команды:

- paddsb, paddsw, paddwd – выполняют сложение знаковых или беззнаковых упакованных байтов, слов, двойных слов.

- psubb, psubw и psubd – выполняют вычитание знаковых или беззнаковых упакованных байтов, слов, двойных слов.

Команды paddsb и paddsw (упакованное сложение с насыщенностью) и psubsb и psubsw (упакованное вычитание с насыщенностью) выполняют сложение или вычитание знаковых элементов данных операнда источника и операнда адресата и приводят результат к граничным значениям диапазона знакового типа данных. Эти команды поддерживают упакованные байты и упакованные слова.

Команды paddusb и paddusw (упакованное сложение без знака с насыщенностью) и psubusb и psubusw (упакованное вычитание без знака с насыщенностью) выполняют сложение или вычитание элементов данных без знака операнда источника и операнда адресата и приводят результат к граничным значениям диапазона типа данных без знака. Эти команды поддерживают упакованные байты и упакованные слова.

Упакованное умножение. Команды упакованного умножения выполняют четыре умножения на парах 16-разрядных знаковых операндов, производя 32-разрядные промежуточные результаты. Пользователи могут выбирать старшие или младшие части каждого 32-разрядного результата.

Команды pmulhw и pmullw умножают знаковые слова операндов источника и адресата и записывают старшую или младшую часть результата в операнд адресата.

Упакованное умножение/сложение. Команда pmaddwd вычисляет произведение знаковых слов операндов адресата и источника. Четыре промежуточных 32-разрядных произведения суммируются в парах, чтобы произвести два 32-разрядных результата.

Команды сравнения

Команды pcmpeqb, pcmpeqw, pcmpeqd (упакованное сравнение на равенство) и pcmpgtb, pcmpgtw, pcmpgtd (упакованное сравнение на «больше») сравнивают соответствующие элементы данных в операндах источника и адресата на равенство или оценивают, кто из них больше. Эти команды генерируют маску единиц или нулей, которые записываются в операнд адресата. Логические операции могут использовать маску, чтобы выбрать элементы. Это можно использовать, чтобы выполнить упакованную условную операцию пересылки без ветвления или набора команд ветвления. Никакие флаги не устанавливаются. Эти команды поддерживают упакованные байты, упакованные слова и упакованные двойные слова.

Команды преобразования

Команды преобразования преобразовывают элементы данных внутри упакованного типа данных.

Команды packsswb и packssdw (упакованный со знаковой насыщенностью) преобразовывают знаковые слова в знаковые байты или знаковые двойные слова в знаковые слова в режиме знаковой насыщенности.

Команда packuswb (упакованный насыщенностью без знака) преобразовывает знаковые слова в байты без знака в режиме насыщенности без знака.

Команды punpckhbw, punpckhwd и punpckhdq (распаковать старшие упакованные данные) и punpcklbw, punpcklwd и punpckldq (распаковать младшие упакованные данные) преобразовывают байты в слова, слова в двойные слова или двойные слова в четверное слово.

Логические команды

Команды pand (поразрядное логическое И), pandn (поразрядное логическое И-НЕ), por (поразрядное логическое ИЛИ) и pxor (поразрядное логическое исключающее ИЛИ) выполняют поразрядные логические операции над 64-разрядных данными.

Команды сдвига

Команды логического сдвига влево, логического сдвига вправо и арифметического сдвига право сдвигают каждый элемент на определенное число битов. Логические левые и правые сдвиги также дают возможность перемещать 64-разрядное поле как один блок, что помогает в преобразованиях типа данных и операциях выравнивания.

Команды psllw, pslld (упакованный логический сдвиг влево) и psrlw, psrld (упакованный логический сдвиг вправо) выполняют логический левый или правый сдвиг и заполняют пустые старшие или младшие битовые позиции нулями. Эти команды поддерживают упакованные слова, упакованные двойные слова и четверное слово.

Команды psraw и psrad (упакованный арифметический сдвиг вправо) выполняют арифметический сдвиг вправо, копируя знаковый разряд в пустые разрядные позиции на старшем конце операнда. Эти команды поддерживают упакованные слова и упакованные двойные слова.

Команда EMMS

Команда emms освобождает состояние MMX. Эта команда должна использоваться, чтобы очистить состояние MMX (чтобы освободить tag-слово регистров FPU) в конце MMX подпрограммы перед вызовом других подпрограмм, которые могут выполнять операции с плавающей точкой.

Наличие этой команды обязательно, если MMX-команды комбинируются с командами сопроцессора.

Использование MMX-команд в программах

Когда необходимо использовать MMX-команды в программе на ассемблере в начале программы указывается директива .mmx. Например:

```
.686
.mmx
.model  flat
.data
m64_1 dq    0a0d0b0c12ef093ch
m64_2 dq    1f05db0c93eee9a0h
m64_3 dq    0
.code
      movq        mm0, m64_1
      paddd           mm0, m64_2
      movq        m64_3, mm0
end
```

Однако, как правило, различные алгоритмы, которые используют MMX-команды, значительно меньше по объему кода, чем окружающая их среда программы, основное назначение которой организовать интерфейс для доступа к данным и взаимодействия ее с пользователем. Поэтому целесообразно использовать либо совместное программирование на языке высокого уровня

(например, на языке C++) и ассемблере, либо использовать механизм ассемблерных вставок.

Рассмотрим, как можно использовать MMX в программах на C++, создаваемых в среде программирования Visual Studio 6.0/2003/2005.

Типы данных __m64 и __int64 предназначены для определения в программе 64-разрядных данных, которые могут затем быть использованы в MMX-командах. Представим предыдущий фрагмент кода в виде функции на языке C++.

```
void mmx_func()
{
        __int64 m64_1 = 0x0a0d0b0c12ef093ch;
        __int64 m64_2 = 0x1f05db0c93eee9a0h;
        __int64 m64_3 = 0;

 __asm
 {
        movq  mm0, m64_1
        paddd mm0, m64_2
        movq  m64_3, mm0
 }
}
```