МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Кафедра "Программное обеспечение вычислительной техники и автоматизированных систем"

SIMD-расширения архитектуры Intel 8086

Методические указания к практическим работам по дисциплине "Параллельные вычисления"

Ростов-на-Дону

20 г.

Составитель: к.ф.-м.н., доц. Габрельян Б.В.

УДК 512.3

SIMD-расширения архитектуры Intel 8086: методические указания – Ростов н/Д: Издательский центр ДГТУ, 20 . – с.

В методической разработке рассматриваются вопросы использования потоковых (SIMD) инструкций в архитектуре Intel 80x86. Даны задания по выполнению лабораторной работы. Методические указания предназначены для направления 01.05.00 – «Математическое обеспечение и администрирование информационных систем».

Цель работы: Ознакомление с набором SIMD-инструкций процессоров семейства Intel 80x86.

I. SIMD-команды.

Первым SIMD-расширением архитектуры Intel x86 стала технология MMX. Процессоры, построенные по этой технологии, получили восемь новых 64-битных регистров данных (MMX-регистры), три новых упакованных типа данных (упакованные целые байтовые, упакованные целые двухбайтовые и упакованные целые четырехбайтовые значения), новые инструкции процессора, поддерживающие работу с новыми типами данных. MMX-регистры получили названия MM0 - MM7. В один MMX-регистр можно поместить (упаковать) восемь байт, четыре двойных байта (двойной байт называют словом – word) или два учетверенных байта (двойное слово – doubleword). Одна арифметическая или логическая MMX-инструкция работает одновременно с несколькими (восемью, четырьмя или двумя) парами целочисленных операндов, т.е. выполняются сразу для многих данных (SIMD-модель). В результате выполнения операции результат может выйти за допустимое значение, например, при сложении двух беззнаковых 16-битных целых значений результат может не поместиться в два байта. В технологии MMX выход за допустимые границы можно обрабатывать одним из трех путей:

1) заворачивание слов (wraparound). Информация о выходе за границы игнорируется, т.е. не устанавливаются флаги переноса или потери порядка;

2) знаковое насыщение (signed saturation). Сохраняется правильный знак. Например, при сложении двух положительных 16-битовых чисел, если результат не помещается в 16-бит, старший (знаковый) бит обнуляется, а остальные биты получают единичное значение, давая тем самым наибольшее возможное положительное значение;

3) беззнаковое насыщение (unsigned saturation). Если происходит переполнение (слишком большое положительное значение), то все биты прописываются единичными значениями, если получился отрицательный результат, все биты обнуляются.

Всего существует 47 MMX-инструкций, разбитых на следующие категории:

передачи данных, арифметические, сравнения, преобразования, распаковки, логические, сдвига и операция сброса флага состояния MMX.

MMX-инструкции

Категория	wraparound	signed saturation	unsigned saturation
1. Арифметические			
сложение	PADDB, PADDW, PADDD	PADDSB, PADDSW	PADDUSB, PADDUSW
вычитание	PSUBB, PSUBW, PSUBD	PSUBSB, PSUBSW	PSUBUSB, PSUBUSW
умножение	PMULL, PMULH		
умножение и сложение	PMADD		
2. Сравнения			
на равенство	PCMPEQB, PCMPEQW, PCMPEQD		
на больше	PCMPGTPB, PCMPGTPW, PCMPGTPD		
3. Преобразования			
упаковать		PACKSSWB, PACKSSDW	PACKUSWB
4. Распаковки			
старшая часть	PUNPCKHBW,PUNPCKHWD, PUNPCKHDQ		
младшая часть	PUNPCKLBW,PUNPCKLWD, PUNPCKLDQ		
5. Логические	упакованные		все 64-бита
and			PAND
and not			PANDN
or			POR
exclusive or			PXOR
6. Сдвига			
логический влево	PSLLW, PSLLD		PSLLQ
логический вправо	PSRLW, PSRLD		PSRLQ
арифметический вправо	PSRAW, PSRAD		
7. Пересылки данных	двойное слово		все 64-бита
регистр-регистр	MOVD		MOVQ
из памяти	MOVD		MOVQ
в память	MOVD		MOVQ
8. Очистка состояния MMX	EMMS		

MMX-регистры отображаются на регистры арифметического сопроцессора.

Дальнейшим развитием SIMD-инструкций стала технология SSE (Streaming SIMD Extensions). Процессоры получили дополнительно восемь 128-битовых регистров в 32-разрядном (XMM0-XMM7) и 16 в 64-разрядном режиме (XMM0-XMM15), 32-разрядный регистр управления и состояния для XMM-регистров (MXCSR), упакованный тип данных для чисел с плавающей точкой одинарной точности (четыре числа в регистре), инструкции для работы с этими

типами данных. Следующим шагом стала технология SSE2. Появились 6 новых типов данных, все 128-битные упакованные: с плавающей точкой двойной точности, целочисленный байтовый, двух-, четырех- и восмибайтовый, а также инструкции для поддержки новых типов. В технологии SSE потоковые регистры уже не связаны с регистрами арифметического сопроцессора.

Следующие расширения – SSE3, SSSE3 (Supplemental Streaming SIMD Extensions), SSE4.1, SSE4.2 не добавили новых типов данных. В SSE3 добавлены 13 новых инструкций, в SSSE2 еще 32 инструкции, в SSE4.1 47 инструкций, в SSE4.2 семь инструкций.

Расширение AVX (Advanced Vector Extensions) привело к появлению восьми в 32-битном режиме и 16 в 64-битном режиме новых 256-битных регистров данных, регистров YMM0-YMM15, младшей частью которых являются соответствующие регистры XMM0-XMM15, расширению инструкций для поддержки 256-битных данных, расширению инструкций для поддержки трехоперандного синтаксиса.

II. Внутренние функции для поддержки SIMD-инструкций.

C++-компиляторы Intel и Microsoft поддерживают так называемы внутренние функции (intrinsics), представляющие потоковые инструкции и, тем самым, позволяющие выполнять эти инструкции в программе без ассемблерных вставок. Внутренние функции разделяются на простые и составные. Последние представляют не одну, но несколько инструкций. Имена встроенных функций задаются по следующей схеме:

_mm_базовая операция_суффикс

здесь базовая операция – это потоковая инструкция, например, сложение или вычитание, суффикс – определяет тип данных, с которым работает инструкция, первые один или два символа суффикса показывают, являются ли данные упакованными (p), расширенно упакованными (ep) или скалярными (s). Следующие символы задают тип:

s	число с плавающей точкой одинарной точности
d	число с плавающей точкой двойной точности
i128	знаковое 128-битное целое
i64	знаковое 64-битное целое

u64 беззнаковое 64-битное целое

i32 знаковое 32-битное целое

u32 беззнаковое 32-битное целое

i16 знаковое 16-битное целое

u16 беззнаковое 16-битное целое

i8 знаковое 8-битное целое

u8 беззнаковое 8-битное целое

Описание внутренней функции выглядит следующим образом

тип возвращаемого значения имя внутренней функции (параметры)

тип возвращаемого значения: может быть void, int, __m64, __m128, __m128d или __m128i.

Например,

Мнемоника	Внутренняя функция
ADDPD	__m128d _mm_add_pd(__m128d a, __m128d b)
ADDPS	__m128 _mm_add_ps(__m128 a, __m128 b)
ADDSD	__m128d _mm_add_sd(__m128d a, __m128d b)
ADDSS	__m128 _mm_add_ss(__m128 a, __m128 b)
DIVPD	__m128d _mm_div_pd(__m128d a, __m128d b)
DIVPS	__m128 _mm_div_ps(__m128 a, __m128 b)
DIVSD	__m128d _mm_div_sd(__m128d a, __m128d b)
DIVSS	__m128 _mm_div_ss(__m128 a, __m128 b)
MOVAPD	__m128d _mm_load_pd(double * p) void_mm_store_pd(double *p, __m128d a)
MOVAPS	__m128 _mm_load_ps(float * p) void_mm_store_ps(float *p, __m128 a)
MOVD	__m128i _mm_cvtsi32_si128(int a) int _mm_cvtsi128_si32(__m128i a) __m64 _mm_cvtsi32_si64(int a) int _mm_cvtsi64_si32(__m64 a)
MULPD	__m128d _mm_mul_pd(__m128d a, __m128d b)
MULPS	__m128 _mm_mul_ss(__m128 a, __m128 b)
MULSD	__m128d _mm_mul_sd(__m128d a, __m128d b)
MULSS	__m128 _mm_mul_ss(__m128 a, __m128 b)

III. Задания.

1. Реализуйте в виде собственных функций следующие алгоритмы численного интегрирования: прямоугольников, трапеций, Симпсона.

2. Отключите поддержку автовекторизации. Создайте программу для тестирования созданных алгоритмов, вычисляющую значение числа π. В процессе тестирования нужно получать результат с большой точностью, замеряя при этом время вычисления.

3. Создайте версии алгоритмов, использующие SIMD-инструкции процессоров семейства x86.

4. Протестируйте векторизованные версии алгоритмов. Измерьте время выполнения операций, сравните с невекторизованными версиями и сделайте выводы.

5. Включите автовекторизацию и повторите тестирование.

IV. Контрольные вопросы.

1. Каковы основные расширения архитектуры процессоров семейства x86 определяет технология MMX?
2. Как загрузить (выгрузить) данные в MMX-регистр?
3. Какие типы данных определены в технологии MMX?
4. Каковы преимущества технологии SSE над технологией MMX?
5. Какая технология позволяет использовать 256-битные регистры?
6. Как выполнить потоковое сложение двух векторов в C++- программе, используя ассемблерную вставку и подходящую SIMD-инструкцию?
7. Как выполнить потоковое сложение двух векторов в C++- программе, используя подходящую встроенную функцию (intrinsic)?

Литература

1. "Intel 64 and IA-32 architectures software developer's guide. Combined volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C " – Intel Corporation, 1997-2013. – 3044 p.

2. "Intel architecture instruction set extensions programming reference" – Intel Corporation, 1997-2012. – 477 p.