УДК 004.428.2 DOI 10.31114/2078-7707-2018-3-144-147

Реализация функции копирования массивов на векторном сопроцессоре

С.И. Аряшев, П.С. Зубковский, В.В. Цветков

ФГУ ФНЦ НИИСИ РАН, г. Москва

aserg@cs.niisi.ras.ru, zubkovsky@cs.niisi.ras.ru, vasily2002@mail.ru

***Аннотация* — В работе рассматривается реализация функции копирования массивов на векторном сопроцессоре. Приводятся результаты измерения ускорения выполнения копирования массивов на векторном сопроцессоре по отношению к выполнению копирования на сопроцессоре вещественной арифметики.**

***Ключевые слова* — векторный сопроцессор, сопроцессор вещественной арифметики, коэффициент ускорения, функция копирования, команды загрузки, команды сохранения.**

I. Введение

С целью ускорения работы процессоров семейства КОМДИВ при решении задач линейной алгебры и задач с комплексными данными в архитектуру процессоров добавлен векторный сопроцессор (CPV) [1].

Основными характеристиками сопроцессора являются:

- поддержка комплексных и векторных типов данных, представленных числами с плавающей запятой одинарной и двойной точности;
- максимальная ширина вектора 128 бит;
- регистровый файл на 64 128-разрядных регистра;
- выполнение до 10 арифметических операций над вещественными числами двойной точности и до 20 арифметических операций над вещественными числами одинарной точности за один такт;
- расширенный набор векторных команд;
- возможность загрузки/сохранения вектора через кэш память 1-го уровня или пары векторов через кэш память 2-го уровня

В рамках создания программного обеспечения разрабатывается специализированная библиотека подпрограмм линейной алгебры [2].

В данной работе приводятся результаты реализации библиотечной функции копирования на векторном сопроцессоре. Предложенная реализация поддерживает выполнение копирования массивов с различным уровнем выравнивания адресов и осуществляет автоматический выбор команд загрузки/сохранения, соответствующих уровню выравнивания массивов. Так на массивах, выравненных по границе 256-ти разрядного слова (align32), используются векторные команды vldq/vsdq, выполняющие загрузку/сохранение двух векторов с использованием КЭШ памяти второго уровня. На массивах, выравненных по границе 128-ми разрядного слова (align16), используются векторные команды vldm/vsdm, выполняющие загрузку/сохранение одного вектора с использованием КЭШ памяти первого и второго уровня. На массивах, выравненных по границе 64-х или 32-х разрядного слова (align8 или align4), копирование выполняется через регистры сопроцессора вещественной арифметики (FPU) с использованием команд ldc1/sdc1 или lwc1/swc1, выполняющих загрузку/сохранение регистров FPU.

II. Тестирование программы копирования массивов

Тестирование программы копирования выполнялось на различных вариантах моделей и различных вариантах аппаратной реализации векторного сопроцессора, включая поведенческую модель, регистровую RTL модель, реализацию сопроцессора на программируемых логических интегральных схемах (ПЛИС) и интегральное микропроцессорное исполнение в виде системы на кристалле.

A. Описание тестовой программы

Программа теста, написанная на СИ, многократно вызывает ассемблерную программу ***copy***, выполняющую функцию копирования на массивах вещественных чисел двойной (double) и одинарной (float) точности с различным уровнем выравнивания адресов. Программа ***copy*** трижды вызывается для double и для float массивов. При первом вызове программа выполняется на FPU на массивах align4 для float или align8 для double, при втором и третьем вызовах программа выполняется на CPV соответственно на массивах align32 и align16 как для float, так и для double.

*A. Описание программы **copy***

Блок схема программы ***copy*** приведена на рис. 1. Программа включает три функциональных блока, выполняющих копирование массива X в массив Y, реализованных с использованием векторных команд

vldm/vsdm в первом блоке, векторных команд vldq/vsdq во втором блоке и команд загрузки/сохранения регистров сопроцессора вещественных операций lwc1/swc1, ldc1/sdc1 в третьем блоке. Программы первых двух блоков исполняются на векторном сопроцессоре, а программа третьего блока исполняется на сопроцессоре вещественной арифметики.

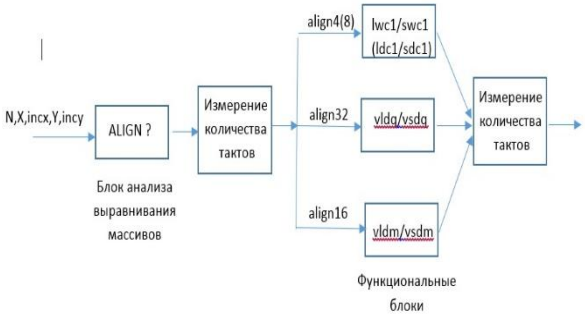

Рис. 1. Блок схема программы *copy*

Программа включает также блок контроля выравнивания массивов X и Y, осуществляющего анализ выравнивания массивов и переключение выполнения программы на блок, команды которого соответствуют уровню выравнивания копируемых массивов.

Количество затраченных на копирование тактов оценивается по разнице показаний счетчика производительности на входе и выходе функционального блока.

Параметрами программы ***copy*** являются количество элементов N в копируемом массиве, указатели на массивы X и Y, и инкременты incX и incY, устанавливающие размеры шагов по элементам массивов при выполнении процедуры копирования. При вызове программы активизируется функциональный блок с командами, соответствующими уровню выравнивания обрабатываемых массивов.

Функциональные блоки имеют общую структурную схему. Отличие состоит в применяемых командах загрузки/сохранения и способах копирования «хвостов» массивов, то есть остатков не кратных количеству копируемых элементов в одной итерации цикла.

На рис. 2 приведена структурная схема функциональных блоков, исполняемых на CPV.

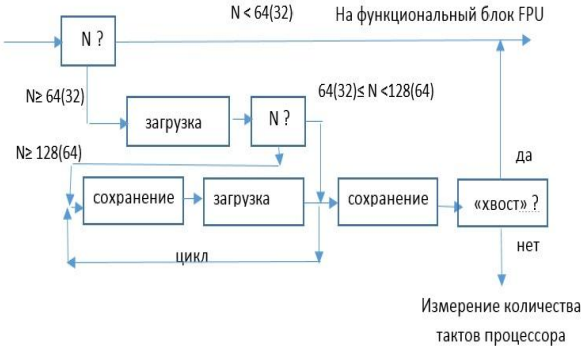

Рис. 2. Структура функционального блока. В скобках указаны цифры для массивов double

Итерация цикла копирования включает выполнение загрузки данных из массива X в блок регистров сопроцессора и последующего сохранения этих данных в массиве Y в памяти процессора. В функциональных блоках, которые исполняются на CPV, блок регистров включает 16 векторных регистров, что позволяет за одну итерацию скопировать 64 элемента float массива или 32 элемента double массива. В функциональном блоке, который исполняется на FPU, используются 8 регистров, что позволяет за одну итерацию скопировать 8 элементов массивов float или double.

III. Анализ результатов тестирования программы копирования массивов

В процессе тестирования измерялось количество тактов, затраченных на выполнение копирования при использовании различных векторных команд загрузки/сохранения. Каждый вызов программы копирования дублируется с целью выполнения предварительной загрузки кэш памяти. Измерение производительности выполняется при каждом втором вызове функции. Тестирование выполнялось при значениях incX=incY=1. По результатам измерений количества тактов процессора, затраченных на выполнение программы, оценивался коэффициент ускорения выполнения программы копирования на векторном сопроцессоре (CPV) по отношению к выполнению варианта реализации программы на сопроцессоре вещественных операций (FPU).

На рис. 3 представлены результаты измерения зависимости коэффициента ускорения (K) от размеров массивов (N) при использовании различных векторных команд загрузки/сохранения. На верхнем рисунке измерения производились на массивах чисел одинарной точности, на нижнем на массивах двойной точности.

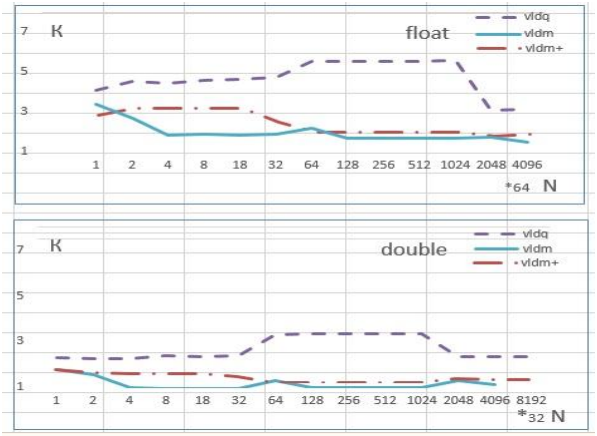

Рис. 3. Зависимости коэффициента ускорения от размеров массива

Наибольшее ускорение наблюдается при использовании команд vldq/vsdq на float массивах, выравненных по границе 32-х байтов. На рисунке эта зависимость представлена пунктирной линией, обозначенной как vldq. Для float массивов, размером не превышающих размеры КЭШ второго уровня (256Kb или примерно 65 тысяч элементов), максимальное значение коэффициента ускорения составляет K=5.5. Для массивов double соответствующее значение коэффициента ускорения примерно в полтора раза меньше. При выполнении команд загрузки/сохранения для массивов, не превышающих размеры КЭШ 2, фиксируется попадание в КЭШ. В этом случае записи в память не происходит, а осуществляется запись только в КЭШ 2 (напомним, что команды vldq/vsdq используют только КЭШ память второго уровня). Для массивов, превышающих размеры КЭШ 2, ускорение падает и равно примерно 2.5 для float массивов. То есть эффект ускорения выполнения копирования на векторном сопроцессоре по отношению к FPU наблюдается и для массивов, не помещающихся в КЭШ. Ускорение наблюдается и при копировании некэшируемых массивов. Однако время выполнения процедуры копирования при этом возрастает.

На массивах, выравненных по границе 16-ти байтов, полученные результаты существенно отличаются. Результаты измерения коэффициента ускорения при использовании команд vldm/vsdm представлены на рисунке сплошной линией, обозначенной как vldm. Копирование в этом случае выполнялось для массивов, выравненных по границе 16-ти байтов. При использовании команд vldm/vsdm заметное ускорение наблюдается только для массивов размером примерно до 256(128) элементов для float(double) массивов, соответственно. Для размеров массивов превышающих эти значения заметного ускорения не наблюдается. Анализ состояния КЭШ памяти показал, что при размерах массивов, для которых наблюдается ускорение, адреса, по которым данные сохраняются в памяти, находятся в КЭШ памяти обоих уровней. В этом случае при выполнении команды сохранения выполняется только модификация данных в КЭШ памяти первого и второго уровней.

Объяснить низкое значение ускорения для массивов, размером превышающих 256(128) элементов, можно, если проанализировать состояние кэш памяти. Анализ состояния КЭШ памяти в этом случае показал, что при этих размерах массивов адреса, по которым данные сохраняются в массиве Y, находятся в КЭШ памяти только второго уровня, а в кэш памяти первого уровня отсутствуют. Это связано с вытеснением этих адресов из КЭШ1 адресами загрузки данных из массива X. Согласно используемой политики кэширования в случае промаха в КЭШ 1 и при попадании в КЭШ 2 осуществляется запись в КЭШ 2, а затем из КЭШ 2 строка считывается в КЭШ 1. Это приводит к дополнительным тактам процессора и является причиной снижения коэффициента ускорения.

Для подтверждения предложенного объяснения в КЭШ 1 были прописаны адреса записи в массив Y путем искусственного добавления в код программы копирования фрагмента загрузки этого массива. Результат измерения коэффициента ускорения в этом случае отображается штрих пунктирной линией, обозначенной как vldm+. Как видно из рисунка максимальное значение коэффициента ускорения для float массивов возросло до трех.

Устранение указанных причин низкого значения коэффициента ускорения при использовании команд vldm/vsdm возможно потребует корректировки политики кэширования при использовании векторного сопроцессора и внесении изменений в аппаратуру сопроцессора. По мнению авторов, устранение этих причин позволит при применении в программе копирования команд vldm/vsdm получить более высокие значения коэффициента ускорения.

Выполненные исследования по реализации функции копирования показали возможность использования предложенной реализации для более быстрого, по сравнению с вещественным сопроцессором, выполнения процедуры копирования на векторном сопроцессоре на массивах с различным уровнем выравнивания. При этом наибольшая эффективность достигается на массивах, выравненных по границе 32-х байтного слова с использованием команд vldq/vsdq. Реализованная в программе возможность автоматического выбора команд загрузки/сохранения в зависимости от уровня выравнивания массивов позволяет организовать копирование массива, выделив в нем области для эффективной обработки с использованием векторных операций загрузки/сохранения.

Литература

[1] С.Г. Бобков, С.И. Аряшев, М.Е. Барских, П.С. Зубковский, Е.В. Ивасюк Высокопроизводительные расширения архитектуры универсальных микропроцессоров для ускорения инженерных расчётов // Информационные технологии. 2014. № 6.С. 27-37.

[2] С.И. Аряшев, П.С. Зубковский, А.С. Кулешов, В.В. Цветков Адаптация библиотеки подпрограмм линейной алгебры GOTOBLAS к архитектуре векторного сопроцессора / Материалы 3-й Всероссийской научно-технической конференции «Суперкомпьютерные технологии» Дивноморское, Геленджик. Т.1. Ростов-на-Дону, 2014. С. 90-95.

The Results of the Implementation of the Copy Function on a Vector Coprocessor

S.I. Aryashev, P.S. Zubkovsky, V.V. Tsvetkov

SRISA

aserg@cs.niisi.ras.ru, zubkovsky@cs.niisi.ras.ru, vasily2002@mail.ru

Abstract — For this paper we present the results of the implementation of the copy function on a vector coprocessor. The proposed implementation supports the execution of copying arrays with different level of address alignment and automatically selects loading/saving instructions corresponding to the level of alignment of arrays. So for arrays aligned on the boundary of the 256-bit word (align32), vector instructions vldq/vsdq are used that load/store two vectors using CACHE memory of the second level, for arrays aligned along the 128-bit word boundary (align16), vector instructions vldm/vsdm are used that load/store one vector using the first and second level cached memory. For arrays aligned on the boundary of a 64-bit or 32-bit word (align8 or align4), copying is performed through real-arithmetic coprocessor (FPU) registers using the ldc1/sdc1 or lwc1/swc1 instructions that load/store FPU registers. The testing of the copy program was performed on different versions of models and different versions of the hardware implementation of the vector coprocessor. Based on the measurement of the number of processor cycles spent on the execution of the program, the acceleration factor of the execution of the copy program on the vector coprocessor (CPV) was evaluated relative to the implementation of the program implementation on the coprocessor for real operations (FPU). The greatest acceleration is observed when using the vldq/vsdq commands for float arrays aligned on the 32-byte boundary, the maximum value of the acceleration coefficient is K = 5.5. For double arrays the corresponding value of the acceleration coefficient is about one and a half times less.

The performed research on the implementation of the copying function showed the possibility of using the proposed implementation for faster execution of the copying procedure on the vector coprocessor for arrays with different levels of alignment, compared with the real coprocessor. The greatest efficiency is achieved for arrays aligned on the 32-byte word boundary using vldq/vsdq commands. The ability to automatically select the commands load/save depending on the level of alignment of arrays allows you to organize the copying of the array, selecting areas for efficient processing using vector operations load / save.

Keywords **- vector coprocessor, coprocessor of real arithmetic, acceleration factor, copy function, loading instructions, save instructions.**

REFERENCES

[1] S.G. Bobkov, S.I. Aryashev, M.E. Barshyn, P.S. Zubkovsky, E.V. Ivasyuk High-Performance Extensions of Microprocessor Architecture for Speeding-Up of Scientific and Engineering Calculations// Information technologies. 2014. № 6.С. 27-37.

[2] S.I. Aryashev, P.S. Zubkovsky, A. S. Kuleshov, V.V. Tsvetkov Adaptation of the library of subroutines of linear algebra GOTOBLAS to the vector coprocessor architecture / Materials of the 3rd All-Russian scientific and technical conference "Supercomputer Technologies" Divnomorskoe, Gelendzhik. T.1. Rostov-on-Don, 2014. pp. 90-95.