Для цитирования: Кожин А. С., Сурченко А. В. Исследование применимости компрессии данных в кэш-памяти микропроцессоров с архитектурой «Эльбрус» // Вопросы радиоэлектроники. 2018. № 2. С. 32–39.
УДК 004.318

А. С. Кожин[1, 2], А. В. Сурченко[1, 2]

[1] АО «МЦСТ», [2] МФТИ (ГУ)

ИССЛЕДОВАНИЕ ПРИМЕНИМОСТИ КОМПРЕССИИ ДАННЫХ В КЭШ-ПАМЯТИ МИКРОПРОЦЕССОРОВ С АРХИТЕКТУРОЙ «ЭЛЬБРУС»

Кэш-память играет важную роль в универсальных микропроцессорах, позволяя уменьшить время доступа к данным и число обращений в оперативную память. Ее объем в современных микропроцессорах достигает сотен мегабайт, основными ограничивающими факторами являются площадь и рассеиваемая мощность. Аппаратное сжатие (компрессия) данных в кэш-памяти может увеличить ее эффективный объем при неизменных физических параметрах, но оно до сих пор не имеет широкого применения в серийных микропроцессорах. Данная работа является первым исследованием в области аппаратного сжатия данных в кэш-памяти микропроцессоров с архитектурой «Эльбрус». Для аппаратной реализации были выбраны алгоритмы ZCA, Base+Delta и Base-Delta-Immediate, которые обладают малой по сравнению с другими алгоритмами задержкой декомпрессии и достаточно высокой степенью сжатия. Схемы компрессии были протестированы в кэш-памяти третьего уровня модифицированного прототипа микропроцессора «Эльбрус-8С2». В статье приведены результаты измерений доли сжатых кэш-строк и степени сжатия на задачах из пакета SPEC CPU2000. Алгоритм Base-Delta-Immediate обеспечил наибольшую степень сжатия среди тестируемых алгоритмов (примерно 1,43 для целочисленных задач и 1,30 для задач с плавающей точкой). Полученные результаты позволяют сделать вывод о практической применимости алгоритма компрессии Base-Delta-Immediate для повышения эффективного объема кэш-памяти.

Ключевые слова: *архитектура, производительность микропроцессора, компрессия данных, кэш-память, прототип.*

Введение

Существенную часть кристалла в современных микропроцессорах занимает кэш-память, суммарный объем которой может достигать сотен мегабайт. Увеличение объема кэш-памяти может повысить коэффициент попаданий и, как следствие, ускорить работу программы. При этом основными факторами, ограничивающими ее максимальный объем, являются площадь и рассеиваемая мощность.

Компрессия (сжатие) данных может повысить эффективный объем хранимой информации. В современных вычислительных системах есть примеры использования компрессии в оперативной памяти [1], однако в отношении кэш-памяти серийных микропроцессоров такие примеры пока не известны, хотя исследования на эту тему ведутся достаточно давно [2]. Основные трудности здесь связаны с необходимостью изменить устоявшуюся структуру кэш-памяти, а также с достаточно большими накладными расходами и увеличением времени доступа при сложной декомпрессии.

Большинство предлагаемых алгоритмов компрессии в кэш-памяти реализует сжатие отдельных блоков – кэш-строк – и размещает большее их количество за счет объединения соседних блоков в «супер-блоки» [3] или увеличения (обычно удвоения) числа хранимых адресных тэгов [4]. Сложные алгоритмы обеспечивают высокую степень сжатия, до двух раз увеличивая эффективный объем хранимой в кэш-памяти информации на задачах с подходящей структурой данных. В качестве известных примеров можно привести алгоритмы FPC [5] и C-Pack [6]. В основе алгоритма FPC лежат разбиение кэш-строки на фиксированные сегменты и проверка этих сегментов по распространенным шаблонам. Более эффективный алгоритм C-Pack дополняет сжатие по статическим шаблонам динамическим справочником. В обоих алгоритмах декомпрессия сегментов одной кэш-строки выполняется последовательно, поэтому имеет достаточно большую задержку (5–8 тактов на частоте 2 ГГц и технологии 28 нм).

Громоздкие вычисления при последовательной декомпрессии существенно увеличивают время доступа в кэш-память, поэтому могут замедлить выполнение задач, которые не относятся к категории «Cache Friendly» [7], и задач с низкой степенью сжатия рабочих данных. Для реализации

в универсальных микропроцессорах больше подходят алгоритмы с независимой декомпрессией отдельных сегментов строки и использованием простых арифметических и логических операций. Они имеют малую задержку декомпрессии и могут обеспечить работу без ухудшения производительности на любых задачах. Примером быстрой и простой схемы компрессии является сжатие нулевых кэш-строк (Zero values compression), которое используют во многих алгоритмах. Так, алгоритм ZCA [8] выделяет специальное расширение кэш-памяти для нулевых кэш-строк, в котором хранит только их адресные тэги и состояния. Этот подход не требует никакого дополнительного времени на декомпрессию, но обеспечивает невысокую степень сжатия. Сжатие нулевых кэш-строк используется и в алгоритмах Base+Delta (B+Δ) и Base-Delta-Immediate (BΔI) [9], которые на сегодняшний день считаются одними из самых быстрых, эффективных и производительных. Декомпрессия в них выполняется параллельно во всех сегментах кэш-строки и добавляет всего один такт ко времени доступа в кэш-память, что делает эти алгоритмы наиболее подходящими для реализации в универсальных процессорах.

Статья посвящена исследованию применимости аппаратной компрессии данных для увеличения эффективного объема кэш-памяти микропроцессоров с архитектурой «Эльбрус». В рамках исследования были реализованы алгоритмы компрессии B+Δ и BΔI. Эффективность компрессии проверена для кэш-памяти третьего уровня с использованием модифицированного прототипа микропроцессора «Эльбрус-8С2» на задачах из пакета SPEC CPU2000.

Алгоритм B+Δ (Base+Delta)

Алгоритм компрессии B+Δ основан на наблюдении, что многие кэш-строки содержат структуры данных с малым динамическим диапазоном значений, т.е. могут быть разбиты на сегменты одинакового размера с нулевыми, повторяющимися или близкими значениями данных. Кэш-строка, обладающая таким свойством, может быть представлена как некоторое базовое значение (base) размером в сегмент и набор разностей между значением данных в каждом сегменте и выбранной базой, каждая из которых обозначается как «дельта» (delta). Если такие наборы имеют меньший размер, чем исходные сегменты, то данное представление будет занимать меньше места, чем исходная кэш-строка. Авторы алгоритма [9], рассматривая вопрос о выборе базового значения для компрессии, пришли к выводу о том, что среди всех сегментов в кэш-строке достаточно выбрать самый первый, поскольку поиск оптимального значения для базы увеличивает задержку, связанную с компрессией, и не дает существенного прироста производительности. Алгоритм B+Δ включает несколько схем компрессии с разными размерами базы и дельт, а также схемы компрессии нулевых кэш-строк и кэш-строк с повторяющимися значениями. В таблице приведены применяемые схемы компрессии и соответствующие им размеры сжатых кэш-строк. Компрессия выполняется для отдельных кэш-строк и позволяет разместить большее количество строк в кэш-памяти фиксированного объема при наличии дополнительных адресных тэгов и состояний.

При компрессии исходная кэш-строка проверяется на возможность сжатия по каждой схеме. Для этого она разбивается на сегменты соответствующего размера, причем первый сегмент выбирается в качестве базы, а для остальных сегментов вычисляются значения дельт. Если хотя бы один из сегментов не может быть представлен таким образом, то данная схема компрессии не подходит. Если же кэш-строка может быть сжата по нескольким схемам, то выбирается схема с наименьшим размером сжатой кэш-строки.

Таблица. Схемы компрессии алгоритмов B+Δ и BΔI для кэш-строк размером 64 Б

Название схемы	Размер базы, Б	Размер дельты, Б	Размер маски, Б	Размер сжатой строки для алгоритмов B+Δ / BΔI, Б	Код
Zero values	0	0	0	0 / 0	0000
Repeated values	8	0	0	8 / 8	0001
Base8-Δ1	8	1	1	16 / 17	0010
Base8-Δ2	8	2	1	24 / 25	0011
Base8-Δ4	8	4	1	40 / 41	0100
Base4-Δ1	4	1	2	20 / 22	0101
Base4-Δ2	4	2	2	36 / 38	0110
Base2-Δ1	2	1	4	34 / 38	0111
Нет компрессии	–	–	–	64 / 64	1111

Рисунок 1. Сжатие кэш-строки по алгоритму B+Δ с одной базой

На рис. 1 продемонстрирован пример работы алгоритма B+Δ для кэш-строки размером 32 байта (пример взят из [9] для задачи 400.perlbench пакета SPEC CPU2006; размер кэш-строки в примерах на рис. 1, 2 выбран для наглядности, во всех остальных случаях он равен 64 байтам). Исходная кэш-строка делится на 8 сегментов по 4 байта, в качестве базы выбирается сегмент 0 (отсчет сегментов в данном случае ведется слева направо). Сжатая кэш-строка содержит базовое значение, а затем 8 дельт, каждая из которых, как видно на иллюстрации, имеет размер 1 байт и является разностью между данными в сегменте и базовым значением. Таким образом, алгоритм сжимает исходную строку по схеме Base4-Δ1 до 12 байт, что на 20 байт меньше изначального размера.

Код схемы компрессии, указанный в таблице, сохраняется вместе с адресным тэгом и когерентным состоянием кэш-строки. При чтении строки из кэш-памяти по этому коду выбирается нужная схема декомпрессии. Восстановление сегментов происходит независимо и заключается в сложении базы с соответствующей дельтой.

Алгоритм BΔI (Base-Delta-Immediate)

Алгоритм B+Δ хорошо справляется с компрессией однородных структур данных, но не подходит для кэш-строк, содержащих данные разных типов, например, смесь структур указателей и целочисленных переменных размером 1 байт. Для таких случаев целесообразно использовать несколько баз для увеличения вероятности компрессии кэш-строки. Авторы статьи [9] показали, что оптимально использовать две базы, причем в качестве одной из них выбирать нулевое значение. Этот алгоритм получил название Base-Delta-Immediate (BΔI). Он позволяет сжимать кэш-строки, содержащие одновременно как данные с небольшим динамическим диапазоном относительно какого-то ненулевого значения, так и данные, близкие по значению к нулю (immediate values).

Алгоритм BΔI использует те же схемы компрессии, что и алгоритм B+Δ, но выполняет их в две стадии. На первой стадии делается попытка сжатия кэш-строки по нулевой базе. Если часть сегментов не удалось сжать на первой стадии, они проверяются на возможность сжатия по ненулевой базе, в качестве которой выбирается первый несжатый сегмент. Компрессия считается успешной, если все сегменты удалось представить в виде дельт одинакового размера относительно нулевой базы или выбранной ненулевой базы. Информация о выборе базы для каждого сегмента кэш-строки (1 – нулевая база, 0 – ненулевая база) представляется в виде маски, в которой каждый разряд относится к соответствующему сегменту. Маска сохраняется в сжатой кэш-строке вместе с ненулевой базой и набором дельт. В таблице указаны размеры маски и сжатой кэш-строки для алгоритма BΔI.

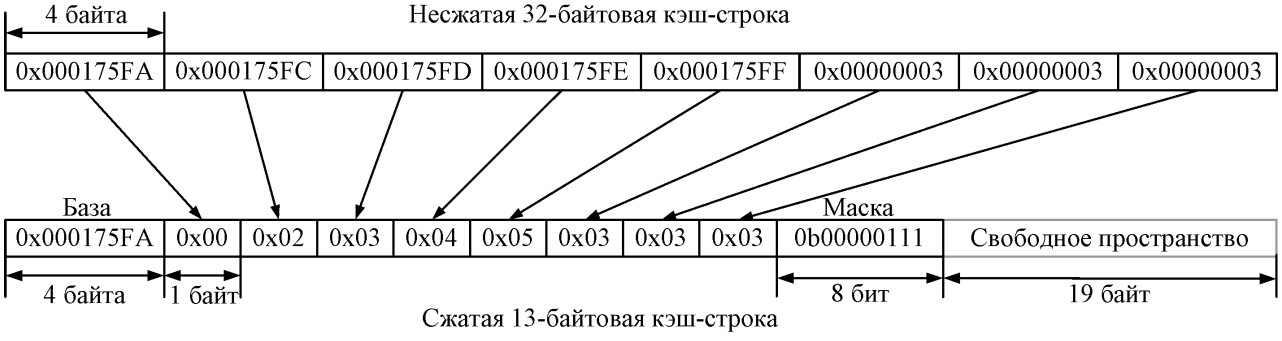

Рисунок 2. Сжатие кэш-строки по алгоритму BΔI

В примере на рис. 2 рассматривается компрессия кэш-строки размером 32 байта из задачи 255.vortex пакета SPEC CPU2000, содержащей две разные структуры данных. Видно, что для данной кэш-строки компрессия B+Δ с использованием одной базы невозможна. Тем не менее при использовании алгоритма BΔI возможно сжатие исходной строки по схеме Base4-Δ1 до размера 13 байт.

Декомпрессия кэш-строк, сжатых по алгоритму BΔI, такая же простая, как и в алгоритме B+Δ. Отличие заключается в обнулении базы для сегментов с единичными значениями битов маски и требует одного дополнительного уровня логики.

Аппаратная реализация и тестовая система

В рамках данной работы были реализованы на языке Verilog алгоритмы компрессии B+Δ и BΔI, а также отдельные схемы компрессии нулевых кэш-строк (Zero values compression) и компрессии повторяющихся значений (Repeated values compression). Алгоритм Zero values проверяет, является ли строка нулевой, и, если это верно, не заносит ее в память данных кэша, сохраняя только адресный тэг и состояние. Алгоритм Repeated values делит кэш-строку на сегменты размером 8 байт и проверяет их на равенство. В случае успеха в сжатой кэш-строке вместо восьми одинаковых сегментов хранится только один.

Исследование эффективности сжатия данных проводилось на модифицированном прототипе микропроцессора «Эльбрус-8С2». Разработанные схемы компрессии были интегрированы в кэш-память третьего уровня (L3-кэш), общую для восьми процессорных ядер. L3-кэш состоит из восьми независимых банков, имеет объем 16 МБ и ассоциативность 16, размер строки данных L3-кэша – 64 Б [10].

Важной особенностью архитектуры «Эльбрус» является поддержка защищенного режима, в котором каждое четырехбайтовое слово в памяти расширяется двухразрядным тегом. Эти теги хранятся не только в оперативной памяти, но и во всех уровнях кэш-памяти процессора. Таким образом, фактический размер кэш-строки L3-кэша без учета битов ECC составляет 68 байтов, из которых 64 байта занимают данные и 4 байта – теги защищенного режима. В реализованной тестовой системе эти теги сжимаются отдельно от основных данных и только по схеме Zero values – для нулевых тегов устанавливается дополнительный признак в коде компрессии, ненулевые теги не сжимаются и добавляются в исходном виде к сжатой кэш-строке, что позволяет несколько повысить степень компрессии, т.к. в большинстве случаев теги имеют нулевые значения.

В качестве тестовых задач использовались задачи из пакета SPEC CPU2000, которые запускались по очереди в однопоточном режиме. Для анализа работы алгоритмов были введены программно-доступные мониторы событий, отслеживающие частоту срабатываний отдельных схем компрессии, частоту выбора отдельных схем компрессии (когда схема обеспечивает наилучшую степень сжатия для данной кэш-строки), частоту одновременного срабатывания нескольких схем компрессии. Для каждой задачи события подсчитывались независимо.

Результаты измерений

График на рис. 3 показывает долю успешно сжатых строк от общего числа строк, размещенных

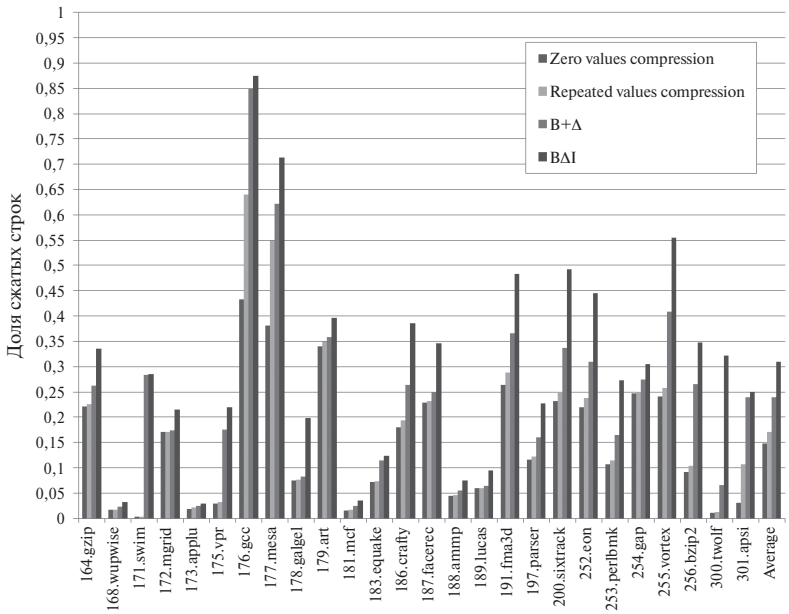

Рисунок 3. Доля сжатых кэш-строк для разных алгоритмов компрессии

в L3-кэше, при использовании четырех алгоритмов компрессии: компрессия нулевых значений (Zero values compression), компрессия повторяющихся значений (Repeated values compression), B+Δ компрессия, BΔI компрессия. В среднем по всем задачам алгоритмы компрессии нулевых значений и повторяющихся значений дают близкие результаты сжатия кэш-строк – 14,8 и 17,1% соответственно. Более сложные алгоритмы B+Δ и BΔI позволяют увеличить количество сжимаемых строк в полтора-два раза – до 23,9 и 31,0% от общего числа строк. Для трех задач (176.gcc, 177.mesa, 255.vortex) число кэш-строк, сжимаемых хотя бы по одному алгоритму, превысило 50%. Пять из 26 задач (168.wupwise, 173.applu, 181.mcf, 188.ammp, 189.lucas) имеют низкую долю сжимаемых строк (меньше 10% для любого алгоритма), причем только одна из них относится к целочисленным (181.mcf).

На рис. 4, 5 представлен вклад отдельных схем компрессии при сжатии по алгоритмам B+Δ (<название теста>_1) и BΔI (<название теста>_2). По оси ординат показана доля применения каждой из схем по отношению к общему числу кэш-строк. Если кэш-строка могла быть сжата сразу по нескольким схемам, применялась схема с наименьшим размером сжатой строки согласно таблице. На рис. 4 представлены результаты целочисленных тестов, на рис. 5 – результаты тестов с плавающей точкой. Компрессия нулевых значений в среднем выполняется для 15,9 и 13,8% всех кэш-строк. Компрессия повторяющихся значений позволяет дополнительно сжать примерно 2,3% кэш-строк. Схемы Base*-Δ* алгоритма B+Δ с одной базой увеличивают число сжимаемых кэш-строк до 26,8 и 21,3%. Введение второй базы (нулевой) существенно повышает частоту срабатывания этих схем, обеспечивая сжатие 36,0 и 26,7% всех строк по алгоритму BΔI для целочисленных задач и задач с плавающей точкой.

Дополнительно была измерена средняя по всем задачам частота срабатывания каждой из схем компрессии, независимо от ее применения как оптимальной (рис. 6). Схемы компрессии нулевых и повторяющихся значений срабатывают реже всех остальных, зато просты в реализации и обладают максимальной степенью сжатия и минимальной задержкой декомпрессии. Наибольшее число строк может быть сжато по схемам Base8-Δ4 и Base4-Δ2, но степень сжатия по этим схемам значительно меньше.

Степень сжатия данных

По результатам измерений оценивалась степень сжатия данных в кэш-памяти третьего уровня для каждой из задач пакета SPEC CPU2000 (рис. 7). Степень сжатия рассчитывалась как отношение объема сжатых данных к их исходному объему. Помимо четырех сравниваемых алгоритмов компрессии также рассматривался усовершенствованный вариант компрессии по повторяющимся значениям, в котором нулевые кэш-строки после сжатия занимают 0 байт (вместо 8 байт в алгоритме Repeated values).

Среднее геометрическое степени сжатия самых простых алгоритмов Zero values и Repeated values составляет 1,187 и 1,180 соответственно. Меньший результат второго алгоритма при большей доле сжатых строк (см. рис. 3) объясняется менее эффективным сжатием нулевых кэш-строк (до 8 байт вместо 0 байт). Усовершенствованный вариант компрессии по повторяющимся значениям повышает степень сжатия до 1,226, но добавляет для каждой

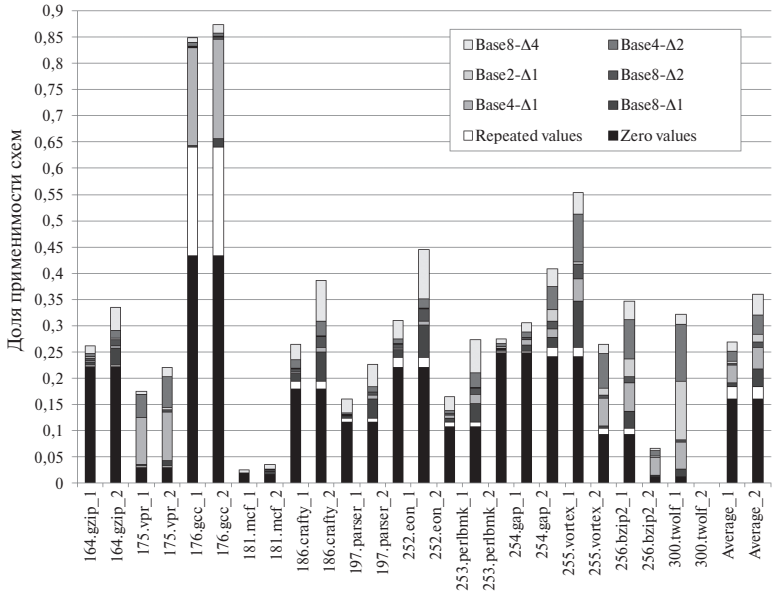

Рисунок 4. Вклад разных схем компрессии алгоритмов B+Δ и BΔI для целочисленных задач

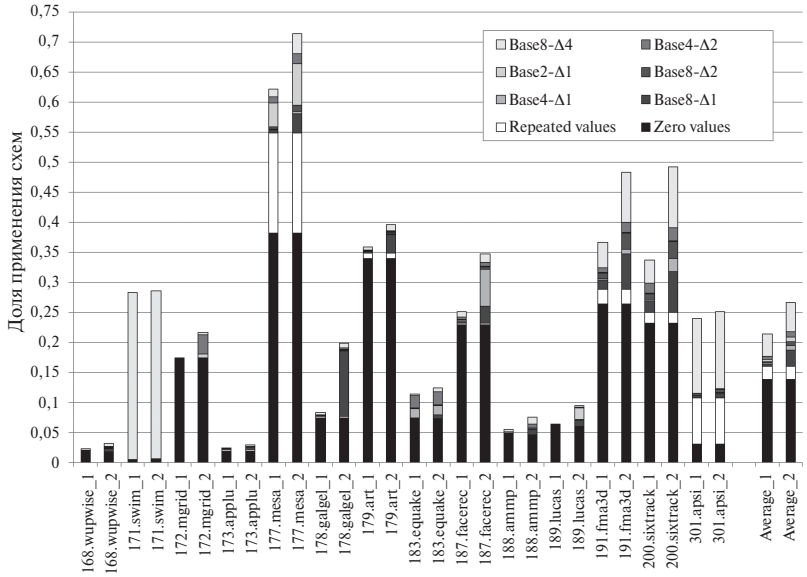

Рисунок 5. Вклад разных схем компрессии алгоритмов B+Δ и BΔI для задач с плавающей точкой

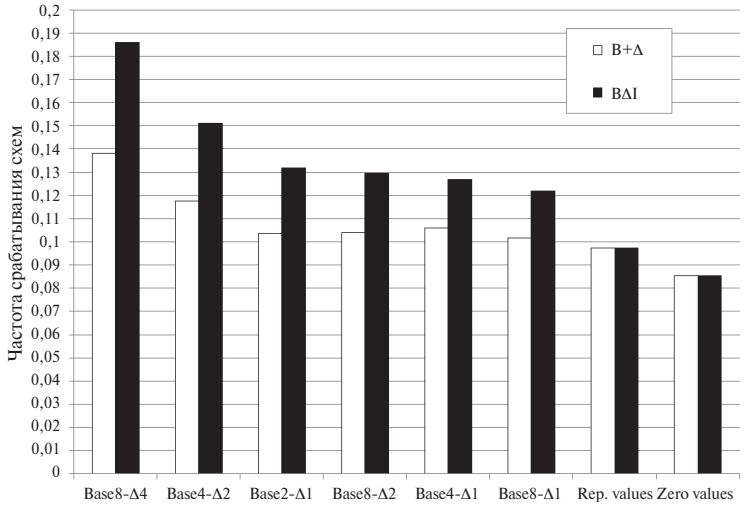

Рисунок 6. Частота независимого срабатывания отдельных схем компрессии

кэш-строки одноразрядный признак выбора схемы компрессии. Алгоритмы B+Δ и BΔI обеспечивают наибольшую степень сжатия данных в кэш-памяти, равную 1,284 и 1,352 соответственно. На семи задачах (176.gcc, 177.mesa, 179.art, 191.fma3d, 200.sixtrack, 252.eon, 255.vortex) степень сжатия по алгоритму BΔI превысила значение 1,5. Только пять задач из 26 (168.wupwise, 173.applu, 181.mcf, 188.ammp, 189.lucas) показали низкую степень сжатия, по значению меньшую, чем 1,1.

Полученные результаты подтверждают следующие предположения. Целочисленные задачи в среднем обладают лучшей сжимаемостью рабочих данных по сравнению с задачами с плавающей точкой (1,424 для целочисленных задач против 1,293 для задач с плавающей точкой на алгоритме BΔI). Рабочий набор данных большинства задач имеет высокую долю нулевых кэш-строк (инициализация нулевыми значениями, разреженные матрицы и др.). Даже такой простой алгоритм, как ZCA [8], может заметно увеличить эффективный объем кэш-памяти на определенных задачах. Алгоритм BΔI требует хранения дополнительного четырехразрядного кода компрессии для каждой строки кэш-памяти и набора сумматоров для компрессии и декомпрессии, но при этом позволяет добиться существенно большей степени сжатия данных.

Стоит отметить, что авторы статьи [9] получили заметно лучшие результаты для алгоритма BΔI, используя программный симулятор x86 и трассы задач пакета SPEC CPU2006 и нескольких известных приложений. Скорее всего, это связано с лучшей сжимаемостью рабочих данных таких задач по сравнению с задачами пакета SPEC CPU2000.

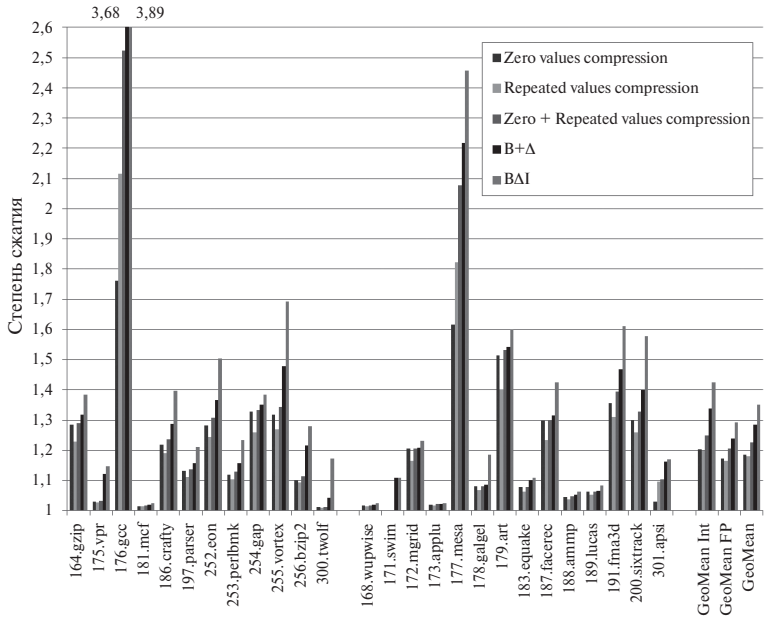

Рисунок 7. Степень сжатия для разных алгоритмов компрессии

Заключение

Данная работа является первым исследованием в области аппаратного сжатия данных для микропроцессоров с архитектурой «Эльбрус». Полученные результаты позволяют сделать вывод о практической применимости алгоритма компрессии BΔI для повышения эффективного объема кэш-памяти. Дальнейшие исследования в этой области будут посвящены повышению степени сжатия данных и интеграции компрессии в структуру кэш-памяти.

СПИСОК ЛИТЕРАТУРЫ

1. Abali B., Franke H., Poff D.E., Saccone R.A., Schulz C.O., Herger L.M., Smith T.B. Memory expansion technology (MXT): software support and performance. IBM Journal of Research and Development, 2001, vol. 45, no. 2, pp. 287–301.
2. Sardashti S., Arelakis A., Stenström P., Wood D.A. A primer on compression in the memory hierarchy. Synthesis Lectures on Computer Architecture, 2015, vol. 10, no. 5, p. 86.
3. Sardashti S., Seznec A., Wood D.A. Yet another compressed cache: a low-cost yet effective compressed cache. ACM Transactions on Architecture and Code Optimization (TACO), 2016, vol. 13, no. 3, art. 27, p. 26.
4. Gaur J., Alameldeen A.R., Subramoney S. Base-victim compression: an opportunistic cache compression architecture. 2016 ACM/IEEE43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, IEEE, 2016, pp. 317–328.
5. Alameldeen A.R., Wood D.A. Adaptive cache compression for high-performance processors. Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA), Munchen, IEEE, 2004, pp. 212–223.
6. Chen X., Yang L., Dick R.P., Shang L., Lekatsas H. C-pack: A high-performance microprocessor cache compression algorithm. IEEE transactions on very large scale integration (VLSI) systems, 2010, vol. 18, no. 8, pp. 1196–1208.
7. Кожин А.С., Нейман-заде М. И., Тихорский В.В. Влияние подсистемы памяти восьмиядерного микропроцессора «Эльбрус-8С» на его производительность // Вопросы радиоэлектроники. 2017. № 3. С. 13–21.
8. Dusser J., Piquet T., Seznec A. Zero-content augmented caches. Proceedings of the 23rd international conference on Supercomputing, New York, ACM, 2009, pp. 46–55.
9. Pekhimenko G., Seshadri V., Mutlu O., Gibbons P.B., Kozuch M.A., Mowry T.C. Base-delta-immediate compression: Practical data compression for on-chip caches. Proceedings of the 21st International conference on Parallel architectures and compilation techniques, Minneapolis, ACM, 2012, pp. 377–388.
10. Kozhin A. S, Polyakov N.Y., Alfonso D.M., Demenko R.V., Klishin P.A., Kozhin E.S., Slesarev M.V., Smirnova E.V., Smirnov D.A., Smolyanov P.A., Kostenko V.O., Gruzdov F.A., Tikhorskiy V.V., Sakhin Y.K. The 5th Generation 28nm 8-Core VLIW «Elbrus-8C» Processor Architecture. Proceedings of the 2016 International Conference on Engineering and Telecommunication (EnT-2016), Moscow, IEEE, 2016, pp. 85–89.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Кожин Алексей Сергеевич, старший инженер, АО «МЦСТ»; ассистент МФТИ (ГУ), 119334, Москва, ул. Вавилова, д.24, тел.: 8 (499) 135-31-08, e-mail: alexey.s.kozhin@mcst.ru.

Сурченко Александр Викторович, студент МФТИ (ГУ); инженер, АО «МЦСТ», 119334, Москва, ул. Вавилова, д.24, тел.: 8 (499) 135-31-08, e-mail: alexander.v.surchenko@mcst.ru.

For citation: Kozhin A.S., Surchenko A.V. Evaluation of cache compression for Elbrus processors. Voprosy radioelektroniki, 2018, no. 2, pp. 32–39.

A.S. Kozhin, A.V. Surchenko

EVALUATION OF CACHE COMPRESSION FOR ELBRUS PROCESSORS

Cache memories play an important role in general purpose microprocessors. It helps to reduce memory access time and off-chip main memory traffic. Aggregate cache capacity of modern microprocessors can reach hundreds of megabytes with the main restricting factors being die area and power consumption. Cache compression has the potential to improve the effective capacity of a cache at the same physical restrictions, however, it is not being used widely in serial microprocessors yet. This is the first research on the cache compression in Elbrus processors. ZCA, Base+Delta and Base-Delta-Immediate algorithms, which possess lower decompression latency in comparison to other algorithms and sufficiently higher compression ratio, were selected for hardware realization. These schemes were evaluated in L3 cache of Elbrus-8C2 processor prototype. In this paper the measurement results for percentage of compressed cache lines and the compression ratio on SPEC CPU2000 benchmarks are shown. Base-Delta-Immediate algorithm has shown the highest compression ratio among the tested algorithms (about 1.43 for integer benchmarks and 1.30 for floating point benchmarks). These results allow to consider Base-Delta-Immediate algorithm suitable for increasing the effective cache capacity.

Keywords: architecture, CPU performance, data compression, cache, prototype.

REFERENCES

1. Abali B., Franke H., Poff D.E., Saccone R.A., Schulz C.O., Herger L.M., Smith T.B. Memory expansion technology (MXT): software support and performance. *IBM Journal of Research and Development,* 2001, vol. 45, no. 2, pp. 287–301.
2. Sardashti S., Arelakis A., Stenström P., Wood D.A. A primer on compression in the memory hierarchy. *Synthesis Lectures on Computer Architecture,* 2015, vol. 10, no. 5, p. 86.
3. Sardashti S., Seznec A., Wood D.A. Yet another compressed cache: a low-cost yet effective compressed cache. *ACM Transactions on Architecture and Code Optimization (TACO),* 2016, vol. 13, no. 3, art. 27, p. 26.
4. Gaur J., Alameldeen A.R., Subramoney S. Base-victim compression: an opportunistic cache compression architecture. *2016 ACM/IEEE43rd Annual International Symposium on Computer Architecture (ISCA),* Seoul, IEEE, 2016, pp. 317–328.
5. Alameldeen A.R., Wood D.A. Adaptive cache compression for high-performance processors. *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA),* Munchen, IEEE, 2004, pp. 212–223.
6. Chen X., Yang L., Dick R.P., Shang L., Lekatsas H. C-pack: A high-performance microprocessor cache compression algorithm. *IEEE transactions on very large scale integration (VLSI) systems,* 2010, vol. 18, no. 8, pp. 1196–1208.
7. Kozhin A.S., Neiman-zade M.I., Tikhorskiy V.V. Memory subsystem impact on the 8-core Elbrus-8C processor performance. *Voprosy radioelektroniki,* 2017, no. 3, pp. 13–21 (In Russian).
8. Dusser J., Piquet T., Seznec A. Zero-content augmented caches. *Proceedings of the 23rd international conference on Supercomputing,* New York, ACM, 2009, pp. 46–55.
9. Pekhimenko G., Seshadri V., Mutlu O., Gibbons P.B., Kozuch M.A., Mowry T.C. Base-delta-immediate compression: Practical data compression for on-chip caches. *Proceedings of the 21st international conference on Parallel architectures and compilation techniques,* Minneapolis, ACM, 2012, pp. 377–388.
10. Kozhin A. S, Polyakov N.Y., Alfonso D.M., Demenko R.V., Klishin P.A., Kozhin E.S., Slesarev M.V., Smirnova E.V., Smirnov D.A., Smolyanov P.A., Kostenko V.O., Gruzdov F.A., Tikhorskiy V.V., Sakhin Y.K. The 5th Generation 28nm 8-Core VLIW Elbrus-8C Processor Architecture. *Proceedings of the 2016 International Conference on Engineering and Telecommunication (EnT-2016),* IEEE, Moscow, 2016, pp. 85–89.

AUTHORS

Kozhin Aleksey, senior engineer, JSC MCST; assistant, MIPT, 24, ulitsa Vavilova, Moscow, 119334, Russian Federation, tel.: +7 (499) 135-31-08, e-mail: alexey.s.kozhin@mcst.ru.
Surchenko Aleksandr, student, MIPT; engineer, JSC MCST, 24, ulitsa Vavilova, Moscow, 119334, Russian Federation, tel.: +7 (499) 135-31-08, e-mail: alexander.v.surchenko@mcst.ru.