doi: 10.14498/tech.2024.1.2

УДК 004.272.4

ОЦЕНКА ЭФФЕКТИВНОСТИ РАБОТЫ АВТОМАТИЗИРОВАННЫХ СИСТЕМ НА БАЗЕ ПРОЦЕССОРОВ С АРХИТЕКТУРАМИ VLIW И X86-64 С ПОМОЩЬЮ ТЕСТИРОВАНИЯ*

***М.С. Гнутов*[1], *С.К. Гнутов*[2], *В.Н. Титаренко*[1]**

[1]Краснодарское высшее военное орденов Жукова
и Октябрьской Революции Краснознаменное училище
им. генерала армии С.М. Штеменко,
Россия, 350063, г. Краснодар, ул. Красина, 4

[2] Военный учебно-научный центр Военно-воздушных сил
«Военно-воздушная академия
им. профессора Н.Е. Жуковского и Ю.А. Гагарина» филиал в г. Сызрани
Россия, 446006, Самарская область, г. Сызрань-7, ул. Маршала Жукова, 1

E-mail: monk666@bk.ru, votung@bk.ru, peresl77@mail.ru

Аннотация. *Рассмотрены типовые архитектурные концепции популярных семейств процессоров, их особенности, преимущества и недостатки. Приведены примеры структур известных процессоров. Обозначены технические ограничения для дальнейшего развития популярных архитектур RISC (CISC) в соответствии с парадигмой машины фон Неймана. Обращается внимание на увеличения размера аппаратного планировщика и возникающий дефицит места на кристалле при росте количества функциональных модулей на ядрах с «последовательной» структурой. Описаны преимущества концепции VLIW и компилятора, работающего с данной архитектурой. Для оценки эффективности функционирования автоматизированных систем с различными архитектурами проведен ряд синтетических тестов, использованы тестовые программы. Проведены измерения производительности для сравнения электронно-вычислительных машин, основанных на ядрах VLIW и x86-64. Предложен ряд алгоритмов для ускорения вычислительных задач автоматизированной системы с параллельной структурой. Подчеркивается необходимость применения концепции параллельного программирования при доработке существующего и разработке нового программного обеспечения для современных автоматизированных систем. Приведен пример преобразования программного кода в параллельную структуру для описания алгоритма последовательной задачи, иллюстрирующего сложные информационные связи.*

*Максим Сергеевич Гнутов, кандидат технических наук, преподаватель кафедры «Защита информации».
*Сергей Константинович Гнутов, кандидат технических наук, доцент кафедры «Общетехнические дисциплины».
Виктор Николаевич Титаренко, кандидат технических наук, слушатель магистратуры.

Ключевые слова: *автоматизированная система, параллельное программирование, архитектура центрального процессора, Astra Linux*

Введение

Центральный процессор – наиважнейшая часть компьютера, осуществляющая выполнение последовательности команд, реализованных с помощью логических операций. Переход от первых процессоров, имевших несложную однопроцессорную архитектуру и работавших на частотах 2,5–4 МГц, к современным процессорам, включающим в себя до нескольких миллиардов транзисторов и работающих на частотах 2–5 ГГц (например, Intel Core i9, AMD Ryzen 9), происходит на фоне и вследствие лавинообразного развития информационных технологий [1]. Растущие потребности пользователей и приложений, требования защиты информации и огромные потоки цифровых данных приводят к необходимости повышения производительности автоматизированных систем не только за счет увеличения их мощности. Концепция параллельных вычислений, реализованная, в том числе, за счет использования «параллельных» процессорных архитектур (например, Intel Itanium), позволит повысить эффективность применения многопроцессорных и многопоточных систем.

Для обеспечения безопасности информации, а также в целях развития национальной экономики, поддержки российских организаций, осуществляющих свою деятельность в сфере информационных технологий и защиты внутреннего рынка Российской Федерации, правительством Российской Федерации были введены ограничения на допуск иностранного программного обеспечения, используемого для государственных и муниципальных нужд. Это простимулировало отечественных производителей электроники и программного обеспечения, а также создало более благоприятные условия для распространения и продвижения национальной продукции внутри страны [2].

Многообещающим направлением, реализующим принципы параллельных вычислений в нашей стране, является семейство микропроцессоров «Эльбрус», основанных на архитектуре VLIW, которые разрабатываются российской компанией АО «Московский центр SPARC-технологий» (базовая организация кафедры ИВТ МФТИ) при участии Института электронных управляющих машин.

Типовые архитектурные концепции популярных семейств процессоров

Слово «архитектура» можно перевести с греческого языка как «искусство строить». Фактически это расположение частей и элементов в общей структуре. Архитектура процессора с точки зрения:

программиста – это совместимость с определенным кластером команд (работающих, например, на Intel x86-64), их систем адресации, организации регистровой памяти (и др.) и способа конечной реализации программы;

непосредственно аппаратной конфигурации – некие определенные свойства, качества и внутренняя конструкция рассматриваемого семейства процессоров.

В настоящее время существует множество видов процессоров и базовых архитектур. Рассмотрим основные виды.

CISC (англ. Complex Instruction Set Computer – «компьютер с полным набором команд») – тип архитектуры, в первую очередь, с нефиксированной длиной команд, а также с кодированием арифметических действий в одной команде

и небольшим числом регистров, многие из которых выполняют строго определенную функцию. Примеры архитектуры: x86 (IA-32) и x86_64 (AMD64).

RISC (англ. Reduced Instruction Set Computer – «компьютер с сокращённым набором команд») – архитектура процессора, в котором быстродействие увеличивается за счёт упрощения инструкций: их декодирование становится более простым, а время выполнения – меньшим. Примеры архитектуры: PowerPC, серия ARM.

VLIW (англ. Very Long Instruction Word – «очень длинная машинная команда») – архитектура с несколькими вычислительными устройствами. Характеризуется тем, что одна инструкция процессора содержит несколько операций, которые должны выполняться параллельно. Примеры архитектуры: Intel Itanium, Эльбрус.

MISC (англ. Minimal Instruction Set Computer – «компьютер с минимальным набором команд») – простая архитектура, используемая в первую очередь для ещё большего снижения энергопотребления процессора и итоговой цены. Используется в IoT-сегменте, роутерах.

OISC (One Instruction Set Computer – «архитектура с единственной инструкцией») – такие архитектуры часто имеют вид: Сделать действие и в зависимости от результата сделать прыжок или продолжить исполнение. Зачастую ее реализация достаточно простая, производительность маленькая, имеется ограничение шиной данных. Примеры архитектуры: BitBitJump, ByteByteJump, SUBLEQ.

TTA (Transport triggered architecture) – вариант архитектуры микропроцессоров, в которой программы непосредственно управляют внутренними соединениями (шинами) между блоками процессора (например, АЛУ, Регистровый файл). TTA является разновидностью OISC. Примеры архитектуры: MOVE Project [13].

Некоторые из указанных выше архитектур сильно разнятся, некоторые схожи, некоторые со временем объединяются. Еще в 80-х годах популярная CISC архитектура начала приобретать некоторые RISC черты. Сейчас в современных процессорах x86-64 и x86 внутри стоит RISC ядро и используется микрокод.

На протяжении многих лет распространенная архитектура x86, x86-64 выигрывала не только за счет совокупности показателей цена – удобство – популярность. Дело в совместимости – x86_64 всё ещё лидер, только как наследник x86. Так как старые программы работают только на x86, то и новые системы должны быть совместимыми для работы привычного программного обеспечения. Развитие архитектуры RISC в современных реалиях оказалось ограниченным из-за парадигмы машины фон Неймана [3], в которой программы реализованы с допущением, что инструкции должны выполняться последовательно, в том же порядке, в каком они указаны в коде. Это не полное отсутствие параллелизма. В суперскалярных процессорах, позволяющих выполнять сразу несколько команд за такт (путем использования нескольких конвейеров) для распознавания зависимостей между машинными инструкциями применяется сложный аппаратный планировщик. Однако при росте количества функциональных модулей в геометрической прогрессии растет и размер этого предсказателя переходов, занимая всё свободное место на кристалле процессора. Это ограничивает развитие ядер на супербыстрых RISC-архитектурах.

При подходе VLIW можно назначить все планирование программному компилятору, который должен выискать в программе независимые инструкции, собрать их

вместе в очень длинные слова-инструкции, после чего отправить на одновременное исполнение функциональными модулями, количество которых (в идеале) равно количеству операций в этой инструкции. Аппаратно VLIW-процессор состоит из нескольких простых функциональных модулей, подключенных к шине процессора, нескольких регистров и блоков кэш-памяти. Максимальная скорость обработки определяется только количеством и внутренними составом самих функциональных модулей.

Сравнение производительности процессоров Эльбрус 8С и Intel Core-i7 на основе синтетических тестов

В рамках данной научно-исследовательской работы был проведен ряд сравнительных исследований автоматизированных систем с различными процессорными архитектурами. Для оценки производительности архитектур VLIW и x86-64 были использованы ЭВМ со следующими характеристиками:

Intel – на базе процессора Intel Core-i7 4570U 1,7 Ghz, процессорная архитектура X86-64, количество ядер/потоков 2/4, видеоадаптер Intel HD Graphics 4600 на операционной системе Astra Linux Special Edition «Смоленск» 1.6;

Эльбрус – на базе процессора Эльбрус 8С 1,3 Ghz, процессорная архитектура VLIW, количество ядер/потоков 8/8, видеоадаптер Radeon HD 7450 1GB на операционной системе Astra Linux Special Edition «Ленинград» 8.1.

Характеристики твердотельных накопителей и оперативной памяти не представлены из-за их малого влияния на результат тестирования ЭВМ.

Для тестирования производительности рабочих станций стенда на базе процессора Эльбрус-8С и его сравнения с ЭВМ на базе процессора Intel решено использовать следующие синтетические тесты: CoreMark, BusSpeed, Dhrystone, Linpack, Memspeed, MPMFLOPS, Whetstone.

Для тестирования производительности стенда было принято решение использовать следующие прикладные программы: PostgreSQL, OpenSSL, Apache 2, PHP 7.0, 7-zip, LibreOffice.

Использование данного программного обеспечения позволяет оценить, каким образом будет вести себя ЭВМ при повседневной работе пользователя или при использовании данного ЭВМ в качестве сервера приложений [4].

Результаты сравнительного тестирования программой mpmfloops.

Результаты тестирования процессора Эльбрус бенчмарком mpmfloops было решено разбить по размеру используемых массивов (1 элемент массива равен 4 байтам). Обобщённые результаты тестирования представлены в табл. 1–3.

В данном тесте процессор Эльбрус 8С демонстрируют преимущество своей архитектуры. При тестировании с использованием массива меньшей длины процессор Эльбрус 8С обгоняет процессор Intel в три раза (при использовании режима тестирования с 2 и 8 операндами). При увеличении размера массива процессор Intel догоняет процессор Эльбрус 8С. При тестировании ЭВМ Эльбрус в режиме 32 операций на малых объёмах данных его преимущество по сравнению с Intel вырастает до пяти раз. Из этого следует, что для увеличения производительности работы программы на процессоре Эльбрус 8С требуется группировать операции и тщательно подбирать размер обрабатываемых данных одного прохода.

Таблица 1

Максимальные показатели на процессоре Эльбрус

Размер массива (количество элементов)	Количество операций	Количество проходов	Время выполнения (сек)	Итоговый результат (Мфлопс)
102400	2	x10000	0,07513	54519
1024000	2	x1000	0,061275	66846
10240000	2	x100	1,809834	2263
102400	8	x10000	0,168349	97322
1024000	8	x1000	0,150419	108923
10240000	8	x100	1,796708	9119
102400	32	x10000	0,439658	149061
1024000	32	x1000	0,401153	163369
10240000	32	x100	1,831856	35776

Таблица 2

Максимальные показатели на процессоре Intel

Размер массива (количество элементов)	Количество операций	Количество проходов	Время выполнения (сек)	Итоговый результат (Мфлопс)
102400	2	x10000	0,102671	19947
1024000	2	x1000	0,165377	12384
10240000	2	x100	0,894984	2288
102400	8	x10000	0,221797	36935
1024000	8	x1000	0,231863	35331
10240000	8	x100	0,898049	9122
102400	32	x10000	0,931825	35165
1024000	32	x1000	0,938907	34900
10240000	32	x100	0,998724	32810

Таблица 3

Показатели ЭВМ «Эльбрус» по отношению к «Intel»

Размер массива (количество элементов)	Количество операций	Количество проходов	Время выполнения (сек)	Итоговый результат (Мфлопс)
102400	2	x10000	136,66 %	36,59 %
1024000	2	x1000	269,89 %	18,53 %
10240000	2	x100	49,45 %	101,10 %
102400	8	x10000	131,75 %	37,95 %
1024000	8	x1000	154,14 %	32,44 %
10240000	8	x100	49,98 %	100,03 %
102400	32	x10000	211,94 %	23,59 %
1024000	32	x1000	234,05 %	21,36 %
10240000	32	x100	54,52 %	91,71 %

Результаты сравнительного тестирования с помощью архиватора «7zip» (рис. 1).

В данном тесте процессор Эльбрус 8С оказывается быстрее процессора Intel. Это можно объяснить большим количеством ядер и возможностью оптимизации под особенности архитектуры VLIW. В задаче архивации процессор Эльбрус оказался быстрее на 10 %, в задаче разархивирования – на 30 %. Из этого можно сделать вывод, что работа с алгоритмами сжатия рабочая станция Эльбрус работает достаточно эффективно, несмотря на малую частоту.

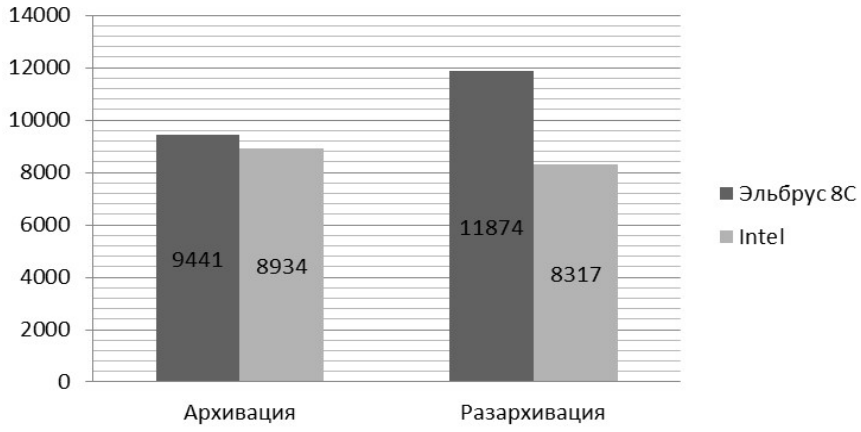

Рис. 1. Результаты тестирования «7zip» (в MIPS)

К сожалению, остальные тестовые программы и синтетические тесты показали в целом отставание российской системы от ее зарубежного аналога. Тестирование прикладных программных продуктов показало, что основной недостаток новой архитектуры VLIW – отсутствие оптимизации уже имеющегося программного кода. Этим объясняется низкая эффективность работы серверов баз данных, интерпретаторов языка, алгоритмов шифрования.

Это происходит в том числе из-за самой концепции написания программ. Современные системы требуют современных решений – при написании программ необходимо использовать подход так называемого «параллельного программирования». Это требует от программиста определенных усилий и понимания работы компилятора. Однако программное обеспечение, оставшееся как наследство от одноядерных CISC- и RISC-систем, не дает воспользоваться всей мощью архитектуры VLIW. Совместное использование возможностей компилятора, а также методов представления и преобразования алгоритмов при написании программистом кода, даст значительное ускорение работы автоматизированных систем за счет увеличения доли параллельных вычислений и достаточно компактной формы алгоритма.

Варианты ускорения вычислительной задачи в автоматизированной системе с параллельной структурой

Написание программы для параллельной структуры может быть достаточно непростым из-за особенностей человеческого мышления. Сложно держать в голове явное (неявное) распараллеливание программного кода и рассчитывать количество свободных функциональных модулей, способных обработать за такт

максимальное количество данных. Рассмотрим теоретические аспекты преобразования алгоритмов на примере запуска приоритетных процессов автоматизированной системы.

Любой набор процессов использует конечное количество системных ресурсов. Для каждого элемента множества системных ресурсов имеется ряд факторов, определяющих очередность исполнения того или иного процесса. Зададим алгоритм запуска приоритетных процессов на 11 позиций с помощью графа (рис. 2).

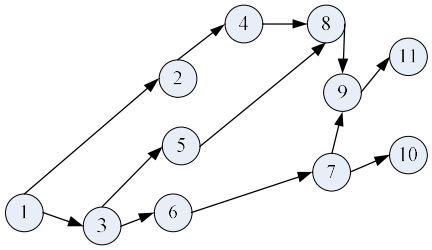

Рис. 2. Произвольный граф алгоритма

Данный граф алгоритма можно преобразовать в ярусно-параллельную форму (рис. 3).

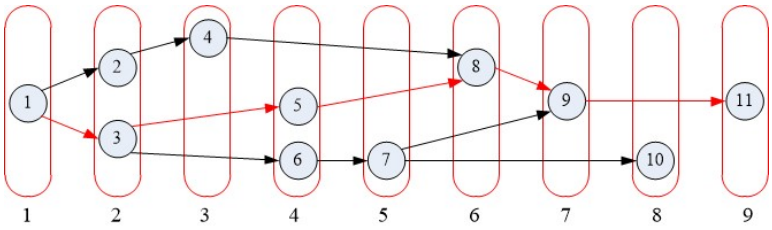

Рис. 3. Граф алгоритма в ярусно-параллельной форме представления

Закон Амдала иллюстрирует, что возможная выгода от использования параллельных вычислений во многом предопределена свойствами применяемых в программах методов и алгоритмов.

Естественно, что максимальное ускорение, которое можно получить при исполнении задач на параллельной вычислительной системе, по закону Амдала рассчитывается как

$$S = \frac{1}{p + \frac{1-p}{c}},$$

где S – коэффициент максимального увеличения скорости работы системы, p – доля параллельных операций в рассматриваемой системе, c – количество ядер.

При большом количестве ядер графического процессора коэффициент максимального увеличения скорости работы системы можно обозначить как

$$S \approx \frac{1}{p}.$$

То есть при доле параллельных операций $p = 50\% = 0.5$, вычисляемых на системе с большим количеством процессоров (или одновременно обрабатываемых параллельных потоков), максимальное ускорение будет всего лишь в два раза выше относительно последовательного варианта расчета. При $p = 9$ – в 10 раз.

В программном исполнении можно добиться максимального ускорения процесса:

первое – увеличивая ширину на ярусно-параллельной форме (далее – ЯПФ) представления алгоритма и уменьшая ее высоту после расчета высоты канонической ЯПФ (поиск кратчайшего пути – путь 1–3–5–8–9–11 на рис. 3 и преобразование ЯПФ на рис. 4). Шириной ЯПФ является количество исполнителей (вычислительных ядер или функциональных модулей), которые могут быть задействованы. Высота ЯПФ – скорость выполнения алгоритма, т. е. количество ярусов можно представить количеством тактов. Построенная ЯПФ дает максимальную скорость и максимальную «жадность» (максимальное количество исполнителей) без изменения программной структуры самого алгоритма.

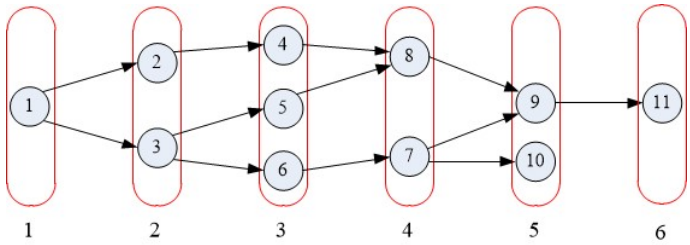

Рис. 4. Преобразованный граф алгоритма в ярусно-параллельной форме представления

и второе – распараллеливанием задач – т. е. преобразовывая отношения между элементами массива. На рис. 5 слева схематично изображена задача сложения элементов, выполненное последовательно за семь тактов, затем то же самое действие, выполненное параллельно попарным сложением с помощью четырех функциональных модулей и за четыре такта

Если на этапе преобразования ЯПФ назначение на процессорные вычислители может совершаться средствами операционной системы и (или) существующим специальным программным обеспечением с определенным наборами библиотек, то на данном этапе необходима работа программиста для преобразования программного кода выполняемого алгоритма [12].

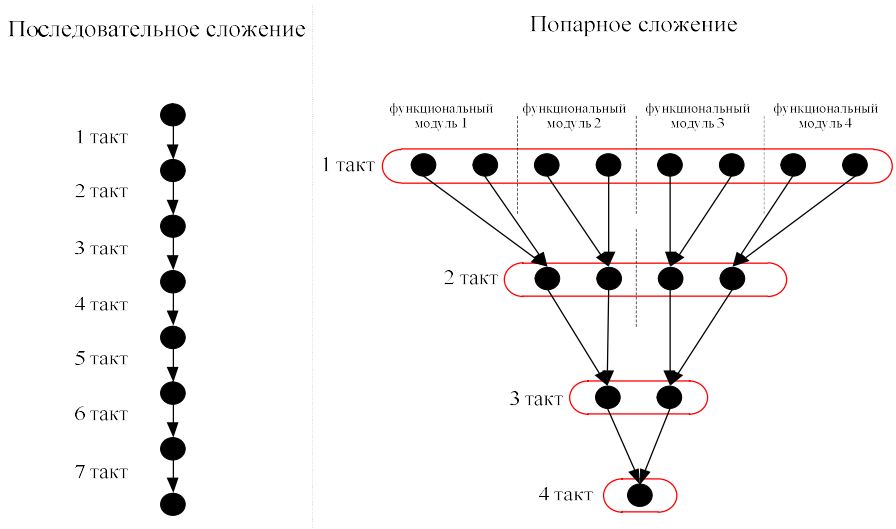

Рис. 5. Эквивалентное преобразование – шифрование в режиме простой замены

Пример преобразования программного кода

Итак, для чего же нужны эти преобразования? Используем алгоритмический язык программирования для описания алгоритма последовательной задачи, иллюстрирующего сложные информационные связи [5]:

(1) read (x,y,z,n);
(2) x:=x*2;
(3) y:=y+3;
(4) z:=-z/4;
(5) n:=n-z;
(6) z:=z*z;
(7) y:=y*y*4;
(8) x:=x+5;
(9) x:=y+x;
(10) write (x,y,z,n).

Оптимизируем данный алгоритм для, условно, трехпоточной системы. Таким образом, в данном случае за один такт система может одновременно обрабатывать m выражений, $0 \le m \le 3$, соответственно, при $m \to max$, найдем максимально возможное ускорение по закону Амдала.

С помощью эквивалентного преобразования из последовательного кода получаем следующую схему в параллельном исполнении (рис. 6).

Количество тактов выполнения алгоритма снизилось с 10 до 5. Данное значение является минимумом, выполнить алгоритм за меньшее количество шагов не представляется возможным. Тем не менее здесь задействовано 4 вычислителя. Можно применить ЯПФ для оптимизации алгоритма (рис. 7).

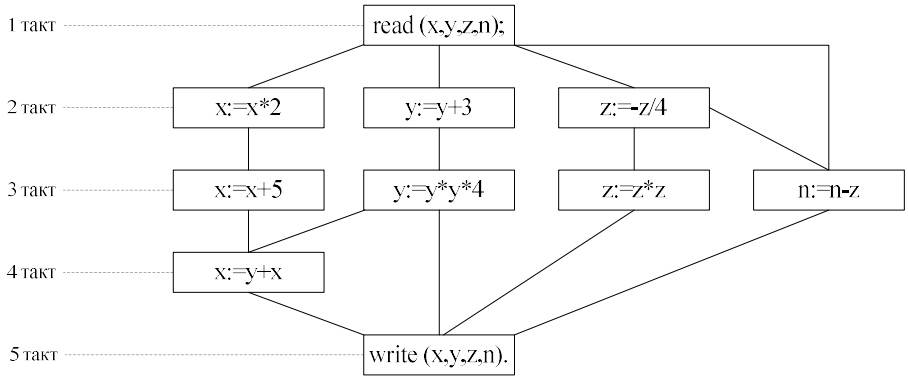

Рис. 6. Эквивалентное преобразование последовательного алгоритма

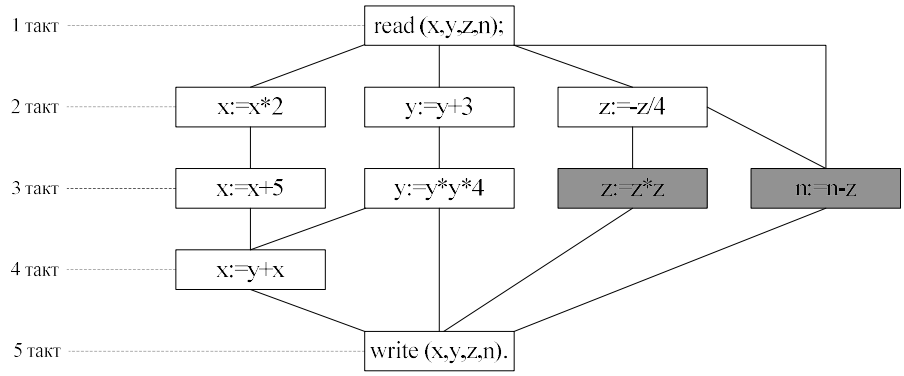

Рис. 7. Выбор блоков для преобразования в ярусно-параллельной форме

Блоки «z:=z*z» и «z:=n-z» можно выполнить как на третьем, так и на четвертом такте, это конечные значения переменных, они не влияют на вычисление переменных x или y. Перенос одного из этих действий на четвертый такт позволит сохранить скорость выполнения и уменьшить количество задействованных потоков до трех (рис. 8).

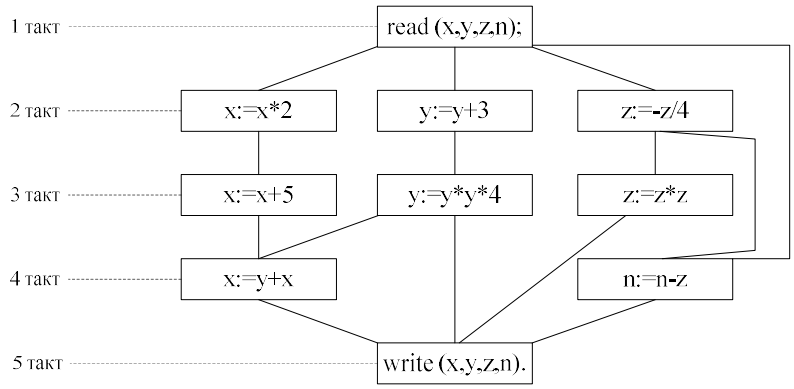

Рис. 8. Оптимальная форма алгоритма

Оптимизационная задача выполнена. Блоки «y:=y+3», «z:=-z/4», «y:=y*y*4», «z:=z*z», «n:=n-z», выполняемые в начале задачи в одном последовательном потоке, были перераспределены и стали блоками, вычисляемыми на соседних потоках, так что значение доли параллельных вычислений p (учитывая операции ввода и вывода) стало равным 0,5. По количеству тактов алгоритм стал вдвое короче, но по закону Амдала максимальное ускорение системы, использующей программу, исполненную с помощью данного алгоритма, составит

$$S = \frac{1}{p + \frac{1-p}{c}} = \frac{1}{0,5 + \frac{1-0,5}{3}} = 1,5.$$

То есть благодаря эквивалентному преобразованию и оптимизации ЯПФ в данной задаче программа на системе с тремя потоками ускорится в полтора раза.

Выводы

Сложным многопроцессорным системам, входящим в состав современных комплексов средств автоматизации, сегодня приходится одновременно обрабатывать множество различных задач. Многопроцессорность и многопоточность на аппаратном уровне, совмещенная с параллелизмом, реализованном на программном уровне с применением алгоритмов преобразования, позволит отечественным процессорам, основанным на архитектуре VLIW, со временем занять одно из лидирующих мест на рынке информационных технологий [6–8].

На данный момент тактовая частота процессора Эльбрус ниже, чем у процессоров Intel, но за счёт организации параллелизма и хорошей оптимизации кода разница в частоте не будет решающим фактором. По пиковой производительности отечественный процессор уже сейчас обгоняет серверный Intel Xeon E5-2609, а по количеству выполняемых команд за один такт превосходит практически любого конкурента.

На основе проведенных исследований можно сделать вывод, что в целом автоматизированная система, основанная на процессоре Эльбрус и операционной системы Astra Linux «Ленинград» 8.1 уже сейчас способна в ряде задач заменить аналогичную на основе ЭВМ с процессором Intel и ОС Windows. ЭВМ с процессором Эльбрус обеспечивает большую надёжность от хакерских атак по сравнению с ЭВМ на основе Intel в связи с использованием собственной защищённой архитектуры и отсутствием известных аппаратных уязвимостей [9,10]. Операционная система Astra Linux Special Edition «Ленинград» также способна обеспечить защищённость данных на программном уровне с использованием как встроенных систем защиты информации [11], так и средств защиты информации, поставляемых сторонними разработчиками. Дополнительная система защиты parsec данной ОС гарантирует защиту от несанкционированного доступа и возможности повышения прав пользователя, а мандатные метки и метки конфиденциальности обеспечивают защищённость от распространённых атак на аналогичные системы семейства Linux.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Концепция RISC архитектур [Электронный ресурс]. Режим доступа: https://life-prog.ru/2_16623_concept_risk_architectyri.html

2. Постановление правительства РФ от 16.11. 2015 N 1236 «Об установлении запрета на допуск программного обеспечения, происходящего из иностранных государств, для целей осуществления закупок для обеспечения государственных и муниципальных нужд» // Собрание законодательства РФ. 2015. №. 47.
3. *Sohi G.S., Breach S.E., Vijaykumar T.N.* Multiscalar processors // In 25 Years ISCA: Retrospectives and Reprints. 1998. Pp. 521–532.
4. *Мишин А.А.* Исследование и сравнительная оценка эффективности архитектуры «Эльбрус» // Инновационные, информационные и коммуникационные технологии. 2019. № 1. С. 91–95.
5. *Плаксин М.А.* «Суперкомпьютеры» vs «параллельное программирование» // Современные информационные технологии и ИТ-образование. 2015. № 11. С. 302–309.
6. ФСТЭК подписал указ об использовании отечественного ПО на объектах критической информационной инфраструктуры [Электронный ресурс]. Режим доступа: https://www.securitylab.ru/news/512136.php.
7. Большое тестирование процессоров различных архитектур [Электронный ресурс]. Режим доступа: https://habr.com/ru/company/icl_services/blog/501588/
8. *Приходько Д.И.* Экспертный анализ недостатков и методы их преодоления для архитектуры VLIW // Научные достижения и открытия современной молодёжи. 2018. С. 14–15.
9. *Мельников М.О.*, *Игонина Е.В.* Аппаратные уязвимости микропроцессоров архитектуры X86/X86_64 ИARM // Фундаментально-прикладные проблемы безопасности, живучести, надежности, устойчивости и эффективности систем. 2019. С. 128 –133.
10. *Ким А.К., Перекатов В.И., Ермаков С.Г.* Микропроцессоры и Вычислительные комплексы семейства «Эльбрус»: учебное пособие. СПб: Питер, 2013. 272 с.
11. *Яковлева Е.С.* Модели управления доступом в операционной системе Astra Linux: Сила привилегий // Состояние и перспективы развития современной науки по направлению «Информационная безопасность». 2020. С. 204–216.
12. *Гнутов М.С., Титаренко В.Н., Сизоненко А.Б.* Оценка эффективности функционирования автоматизированных систем с различными процессорными архитектурами // Информатика: Проблемы, Методы, Технологии (ISMP). Материалы XXI Международной научно-технической конференции. Воронеж: ВЭЛБОРН, 2021. С. 1582–1591.
13. Виды популярных архитектур процессоров [Электронный ресурс]. Режим доступа: https://tproger.ru/articles/processors-architectures-review.

Статья поступила в редакцию 03 марта 2024 г.

EVALUATION OF THE EFFICIENCY OF AUTOMATED SYSTEM BASED ON PROCESSORS WITH VLIW AND X86-64 ARCHITECTURES THROUGH TESTING*

M.S. Gnutov[1], S.K. Gnutov[2], V.N. Titarenko[1]

[1]Krasnodar Higher Military Order of Zhukov
and the October Revolution Red Banner School
named after Army General S.M. Shtemenko
4, Krasina st., Krasnodar, 350063, Russian Federation

[2]Military Educational and Scientific Center of the Air Force
"Air Force Academy named after Professor N.E. Zhukovsky and Yu.A. Gagarin"
1, Marshala Zhukova st., Syzran-7, 446006, Russian Federation

E-mail: monk666@bk.ru, votung@bk.ru, peresl77@mail.ru

Abstract. *The typical architectural concepts of popular processor families, their features, advantages and disadvantages are considered. Examples of structures of known processors are given. Technical limitations for the further development of popular RISC (CISC) architectures in accordance with the von Neumann machine paradigm are outlined. Attention is drawn to the increase in the size of the hardware scheduler and the resulting shortage of space on the chip with an increase in the number of functional modules on cores with a "sequential" structure. The advantages of the VLIW concept and the compiler working with this architecture are described. To assess the effectiveness of the functioning of automated systems with various architectures, a number of synthetic tests were carried out and test programs were used. Performance measurements were carried out to compare electronic computers based on VLIW and x86-64 cores. A number of algorithms have been proposed to accelerate computational tasks of an automated system with a parallel structure. The need to apply the concept of parallel programming when refining existing and developing new software for modern automated systems is emphasized. An example of converting program code into a parallel structure is given to describe an algorithm for a sequential task, illustrating complex information connections.*

Keywords: *Automated system, parallel programming, central processor architecture, Astra Linux*

REFERENCES

1. Concept of RISC architectures [Electronic resource]. Access mode: https://life-prog.ru/2_16623_concept_risk_architectyri.html
2. Postanovlenie pravitelstva RF ot 16.11. 2015 N 1236 «Ob ustanovlenii zapreta na dopusk programmnogo obespecheniya, proiskhodyashchego iz inostrannyh gosudarstv, dlya celej osushchestvleniya zakupok dlya obespecheniya gosudarstvennyh i municipal'nyh nuzhd» [On establishing a ban on the admission of software originating from foreign countries for the purposes of procurement to meet state and municipal needs] // Sobranie zakonodatel'stva RF. 2015. №. 47. (In Russian).
3. *Sohi G.S., Breach S.E., Vijaykumar T.N.* Multiscalar processors // In 25 Years ISCA: Retrospectives and Reprints. 1998. Pp. 521–532.
4. *Mishin A.A.* Issledovanie i sravnitelnaya ocenka effektivnosti arhitektury «Elbrus» [Research and comparative assessment of the effectiveness of the Elbrus architecture] // Innovacionnye, informacionnye i kommunikacionnye tekhnologii. 2019. №. 1. Pp. 91–95. (In Russian).

*Maksim S. Gnutov, (PhD. Sci. (Techn.)), Associate Professor.
Sergey K. Gnutov, (PhD. Sci. (Techn.)), Associate Professor.
Viktor N. Titarenko, (PhD. Sci. (Techn.)), Associate Professor.

5. *Plaksin M.A.* «Superkompyutery» vs «parallelnoe programmirovanie» [“Supercomputers” vs “parallel programming”] // Sovremennye informacionnye tekhnologii i IT-obrazovanie. 2015. № 11. Pp. 302–309. (In Russian).
6. FSTEC signed a decree on the use of domestic software at critical information infrastructure facilities [Electronic resource]. Access mode: https://www.securitylab.ru/news/512136.php.
7. Large testing of processors of various architectures [Electronic resource]. Access mode: https://habr.com/ru/company/icl_services/blog/501588/
8. *Prikhodko D.I.* Ekspertnyj analiz nedostatkov i metody ih preodoleniya dlya arhitektury VLIW [Expert analysis of shortcomings and methods for overcoming them for VLIW architecture] // Nauchnye dostizheniya i otkrytiya sovremennoj molodyozhi. 2018. Pp. 14–15. (In Russian).
9. *Melnikov M.O., Igonina E.V.* Apparatnye uyazvimosti mikroprocessorov arhitektury X86/X86_64 IARM [Hardware vulnerabilities of microprocessors of the X86/X86_64 IARM architecture] // Fundamentalno-prikladnye problemy bezopasnosti, zhivuchesti, nadezhnosti, ustojchivosti i effektivnosti sistem. 2019. Pp. 128 –133. (In Russian).
10. *Kim A.K., Perekatov V.I., Ermakov S.G.* Mikroprocessory i vychislitelnye kompleksy semejstva «Elbrus» [Microprocessors and Computing Complexes of the Elbrus Family]. SPb: Piter, 2013. 272 p. (In Russian).
11. *Yakovleva E.S.* Modeli upravleniya dostupom v operacionnoj sisteme Astra Linux: Sila privilegij [Access control models in the Astra Linux operating system: The power of privileges] // Sostoyanie i perspektivy razvitiya sovremennoj nauki po napravleniyu «Informacionnaya bezopasnost». 2020. Pp. 204–216. (In Russian).
12. *Gnutov M.S., Titarenko V.N., Sizonenko A.B.* Ocenka effektivnosti funkcionirovaniya avtomatizirovannyh sistem s razlichnymi processornymi arhitekturami [Assessing the efficiency of functioning of automated systems with different processor architectures] // Informatika: Problemy, Metody, Tekhnologii (ISMP). Materialy XXI Mezhdunarodnoj nauchno-tekhnicheskoj konferencii. Voronezh: VELBORN, 2021. Pp. 1582–1591. (In Russian).
13. Types of popular processor architectures [Electronic resource]. Access mode: https://tproger.ru/articles/processors-architectures-review.