УДК 004.2; 004.31 DOI: 10.17587/it.26.67-79

А. Ю. Попов, канд. техн. наук, доц., e-mail: alexpopov@bmstu.ru,
Московский государственный технический университет им. Н. Э. Баумана, г. Москва

Принципы организации гетерогенной вычислительной системы с набором команд дискретной математики

В МГТУ им Н. Э. Баумана разработана и продолжает совершенствоваться вычислительная система с набором команд дискретной математики. Система состоит из вычислительных устройств с различной архитектурой и позволяет пользовательским программам обращаться к аппаратным функциям хранения и обработки больших множеств, структур данных, графов и других математических абстракций, используемых в задачах дискретной оптимизации. Основу системы составляет принципиально новое вычислительное устройство — процессор обработки структур Leonhard x64, обладающий высоким быстродействием и функциональностью.

В предыдущих работах продемонстрированы результаты проектирования вычислительной системы, раскрыты особенности выполнения программ и обработки данных, представлена структура нового процессора, приведены результаты экспериментов по измерению производительности.

В данной статье мы обобщаем и систематизируем ряд ключевых принципов построения системы с набором команд дискретной математики, которые в первую очередь обеспечивают высокий уровень производительности. Мы последовательно излагаем причину введения каждого из принципов и следствия его внедрения, демонстрируем примеры их реализации. Представленная статья имеет важный теоретический смысл для всего проекта, так как многие из представленных результатов сформулированы впервые.

Ключевые слова: *структура данных, вычислительная система с набором команд дискретной математики, система со многими потоками команд и одним потоком данных, процессор обработки структур, B + дерево*

Введение

Наборы команд современных микропроцессоров достигают нескольких сотен машинных инструкций, однако число вычислительных команд в них ограничено: это команды выполнения основных арифметических действий (сложение, вычитание, умножение и деление), команды математической логики (конъюнкция, дизъюнкция, отрицание, эквивалентность), сдвиги операндов, команды выполнения прямых и обратных тригонометрических функций, логарифмирование, извлечение корня и некоторое число команд обработки векторов. С точки зрения поддержки математических операций эти команды покрывают лишь часть востребованных в вычислениях действий. Непосредственно поддерживаются лишь такие разделы математики, как арифметика, математическая логика и тригонометрия. Остальные же разделы математики, в том числе и дискретная математика и принятые в них операции, обеспечиваются лишь посредством трансляции в существующий набор арифметических и логических команд. При этом большинство исполняемых в вычислительном потоке команд не связаны с вычислениями напрямую, а носят сопутствующий характер. Это такие типы команд, как пересылка данных, ветвления и команды управления (переходы, передача управления, возвраты и т. д.).

Анализ кода программ показывает [1—3], что команды пересылки составляют около 50 % потока инструкций, обрабатываемых процессором. Этот факт свидетельствует о проявлении проблемы семантического разрыва, когда большинство команд не связаны напрямую с вычислениями, либо обеспечивают аппаратную поддержку лишь части математических действий. Другие команды, такие как поиск минимума/максимума в блоке векторной обработки (SSE Unit, Streaming SIMD Extensions), закрывают лишь незначительную часть потребностей в аппаратной поддержке вычисли-

тельных алгоритмов. Вместе с тем при решении практических задач на ЭВМ используется существенно большее число математических операций дискретной математики, математической логики, дифференциального и интегрального исчислений, линейной алгебры, аналитической геометрии, математической статистики и других. Однако очевидно, что их реализация требует существенного изменения в структуре электронных вычислительных машин и является крайне трудоемкой. В данной работе мы более подробно остановимся на вопросах аппаратной поддержки операций дискретной математики, которая проводится в рамках экспериментальных научно-исследовательских работ в МГТУ им. Н. Э. Баумана.

Обзор существующих подходов к ускорению алгоритмов дискретной оптимизации

Наиболее перспективным способом повышения производительности компьютерных систем является применение специализированных аппаратных ускорителей (акселераторов). В многочисленных работах по данной тематике описаны примеры устройств, которые позволяют существенно снизить время выполнения как отдельных вычислительных операций, так и целых алгоритмов. Наиболее выдающиеся примеры ускорения для задач дискретно-оптимизационного характера достигаются благодаря применению графических ускорителей GPGPU (General-purpose computing on graphics processing units), а также специализированных ускорителей на основе СБИС и программируемых логических интегральных схем (ПЛИС типа FPGA). Так, благодаря применению платформы GPGPU NVidia Tesla удается ускорить алгоритм поиска кратчайшего пути на графе (Single-source shortest path, SSSP) до 400 раз [4]. Благодаря применению специализированного устройства на основе FPGA достигается ускорение преобразования Хафа в 370 раз [5]. Также в работе [5] приведены примеры ускорения вычислений статистических функций, алгоритмов биоинформатики и алгоритмов финансового анализа от 50 до 100 раз в сравнении с универсальными компьютерами. Неплохих результатов удается достичь и в области криптографии [6].

При этом стоит отметить ряд ограничений и недостатков как технологии ускорения на основе GPGPU, так и специализированных ускорителей на основе СБИС и FPGA. Было выявлено, что эффективность решения задач на GPGPU существенно зависит как от формата представления данных в локальной памяти графического процессора, так и от возможности представления алгоритма в виде параллельных потоков вычислений. Например, для алгоритма SSSP традиционный вариант представления графа в виде матрица смежности [7, 8] оказывается малопригодным для реализации на GPGPU в связи с сильной разреженностью исходных данных. Производительность графических ускорителей при таком представлении графа оказывается низкой (в сравнении с универсальными микропроцессорами). Другими проблемами применения GPGPU являются физические ограничения на объемы исходных данных алгоритма, размещаемых во внутренней кэш-памяти, а также необходимость их копирования из оперативной памяти центрального процессора в пространство памяти ускорителя GPGPU.

Вместе с тем существенным недостатком специализированных ускорителей на основе СБИС и FPGA является их низкая универсальность и высокая трудоемкость их разработки: такое устройство изначально создается для решения конкретной алгоритмической задачи и оказывается непригодным для решения других подобных проблем. Временные затраты на организацию системной логики (высокоскоростных шин передачи данных, системного программного обеспечения, средств компиляции кода, программных интерфейсов и пр.) оказываются настолько существенными, что не позволяют в разумные сроки представить такой ускоритель заказчику. Как следствие, разработка высокопроизводительных алгоритмов была и во многом остается трудоемкой научно-исследовательской, а не чисто инженерной задачей.

Понимание указанных недостатков позволило сформулировать ряд первичных требований к разрабатываемой вычислительной системе:

- вычислительная система с аппаратной поддержкой операций дискретной математики должна обладать достаточной универсальностью для ускорения широкого круга задач дискретной оптимизации;
- требуется пересмотреть существующие и разработать ряд новых базовых принципов функционирования вычислительной системы в целях повышения ее эффективности при решении задач дискретной оптимизации;
- архитектура вычислительной системы должна объединять различные вычислительные средства (универсальные микропроцессоры, ускорители вычислений, специализи-

рованные микропроцессоры, контроллеры памяти и системных шин, специализированную память) и поддерживать простое и эффективное программное управление ими;
- необходимо использовать доступную элементную базу ПЛИС FPGA и СБИС для скорейшей реализации системы.

Исходя из сказанного создание действующего образца такой системы в обозримом будущем может показаться маловероятным в связи с противоречивостью требований и масштабностью постановки задачи. Тем не менее, на данный момент уже разработана и прошла цикл испытаний третья версия системы Leonhard x64, она успешно функционирует в составе Центра обработки данных МГТУ им. Н. Э. Баумана, а для заинтересованных разработчиков к ней открыт удаленный доступ. По результатам исследований опубликован ряд работ, в которых отражены как базовые аппаратные принципы [9—12], так и результаты решенных с ее помощью прикладных задач [15, 16].

В предыдущих работах были проанализированы принципы организации существующих ЭВМ и систем, влияющие на эффективность обработки структур данных, разработана общая концепция функционирования вычислительных систем с аппаратной поддержкой операций над структурами данных [9, 10], исследованы способы ускорения вычислительных операций в такой системе [10, 11], разработан и реализован специализированный процессор обработки структур данных. На основе этого устройства создан прототип вычислительной системы [12], разработан набор команд и средства компиляции программ, выполнено проектирование ряда алгоритмов оптимизации [10, 11], проведено тестирование и сравнение полученного решения с существующими универсальными вычислительными системами [12].

По мере дальнейших исследований многие подходы потребовали уточнения и более последовательного изложения. Далее будут представлены шесть основных принципов функционирования системы.

Принципы построения ЭВМ с набором команд дискретной математики

Структуры данных являются ключевым и фундаментальным понятием в программировании и широко применяются в таких классах задач дискретной математики, как задачи оптимизации на сетях и графах, общие задачи комбинаторной оптимизации, задачи управления. Исследованиям новых структур, алгоритмам выполнения операций и принципам их применения посвящен ряд фундаментальных трудов [7, 8]. Однако в применяемых в настоящий момент вычислительных системах вся обработка алгоритмов и структур данных проводится на универсальных микропроцессорах, в которых аппаратно реализованы лишь арифметическая и логическая обработки числовых значений.

Следует отметить, что несмотря на большой интерес научного сообщества к созданию и применению различных ускорителей примеров построения эффективной платформы ускорения задач дискретно-оптимизационного характера и структур данных крайне мало. Наиболее близкими по технической сущности можно считать ассоциативные процессоры CAPP (*Content Addressable Parallel Processors*) [13], проекты STARAN и ASPRO, построенные на основе процессорных элементов (ПЭ), объединенных сложной сетью передачи информации [14]. Каждый 8-разрядный ПЭ в таких системах обрабатывает только свою локальную память, выполняя над ее содержимым простейшие арифметические и логические операции. Объединение большого числа простых процессорных элементов позволяет достичь высокого уровня параллельности. В каждый момент времени все процессорные элементы CAPP обрабатывают одну и ту же команду, в связи с чем такая архитектура относится к классу SIMD. В работе [14] исследуется возможность реализации ассоциативных операций над таблицей реляционной базы данных, хранимой в таком процессоре (рассмотрены такие операции, как Поиск, Следующий, Первый, Максимум, Минимум, И-ИЛИ-НЕ операции, Сумма, Количество). Помимо достоинств, ассоциативные процессоры CAPP имеют ряд недостатков:
- требуется реализация сложной сети передачи информации, необходимой для выполнения ряда ассоциативных операций;
- процессор обладает высокой удельной стоимостью единицы обрабатываемой информации (т. е. число логических элементов процессора в расчете на одно обработанное информационное слово достаточно велико);
- реализация операций над несколькими множествами (объединение, пересечение множеств и т. д.) сильно затруднено;
- в связи с ассоциативным характером памяти затрудняется возможность хранения и обработки больших массивов информации.

В связи с вышеизложенными недостатками известных универсальных и специализированных систем был сформулирован первый принцип, определяющий гетерогенный характер дискретно-оптимизационного вычислительного процесса.

Принцип 1 гетерогенности вычислений. *Для разработки вычислительной системы с аппаратной поддержкой операций дискретной математики необходимо реализовать специальные аппаратные средства обработки множеств и структур данных. Это позволит построить архитектуру вычислительной системы таким образом, чтобы множества отношений и множества данных обрабатывались параллельно и независимо.*

Архитектурные принципы современных вычислительных машин и систем, как показано в ряде работ [9, 12], не способствуют ускорению обработки ссылочных структур данных. Расслоение оперативной памяти ведет к замедлению обращений по произвольным адресам. Пакетный режим передачи информации между процессором и памятью делает неэффективным загрузку небольших информационных слов, а сегментно-страничная организация виртуальной памяти приводит к двойным обращениям к памяти. Отказ же от этих принципов может ускорить обработку ссылочных структур (деревьев, списков), но негативно отразится на обработке векторных структур данных (массивов, хэш-таблиц) и функциональности ЭВМ. Таким образом, дальнейшее совершенствование архитектуры вычислительных машин в рамках существующих принципов (например, увеличение размеров пакетов и страниц, увеличение объемов памяти) будет по-прежнему приводить к противоречивым результатам: при попытке ускорить работу одних алгоритмов будет происходить замедление при обработке других. Программная модель вычислительной машины представляет архитектуру ЭВМ в упрощенном виде, что приводит к разрыву между принципами проектирования программного обеспечения и реальной практикой программирования. Например, алгоритм с лучшей оценкой вычислительной сложности может работать дольше, чем алгоритм, который в большей степени учитывает особенности аппаратного обеспечения, но обладает худшей теоретической оценкой вычислительной сложности. Это противоречие может быть частично устранено благодаря применению аппаратных средств, реализующих более совершенные механизмы для поддержки операций над структурами данных, которые работают параллельно и одновременно с существующей подсистемой памяти. Также важно, чтобы наличие таких средств не приводило к сокращению универсальности системы, но позволяло ускорить решение ряда задач, основанных на использовании ссылочных структур данных.

Исследования [12] показали необходимость и перспективы глубокого пересмотра основополагающих принципов представления информации в ЭВМ, в том числе форм представления структурированных и неструктурированных данных, способов обработки множеств, механизмов доступа к данным. В связи с этим исследования, направленные на определение новых парадигм обработки данных, в настоящее время чрезвычайно актуальны.

Безусловным преимуществом аппаратной реализации действий над множествами является возможность существенно более быстрой и параллельной обработки тех операций, которые требуется выполнять одновременно над несколькими элементами. В таком алгоритме сами множества или порождающие их процедуры должны быть записаны в память ЭВМ: на внешний носитель, в оперативную память, в постоянное запоминающее устройство или регистры процессора. Доступ к этой памяти может быть выполнен через независимые шины, обработка полученной из памяти информации может вестись независимо от других видов обработки, аппаратные устройства могут учитывать специфические форматы хранимых данных и выполнять основные алгоритмы с большей степенью параллельности. Таким образом, первый принцип определяет цели исследования: разработку новых архитектурных принципов и устройств аппаратной обработки множеств и структур данных.

Принцип 2 двойственности структур данных. *Для возможности параллельной обработки структур данных в состав вычислительной системы вносится аппаратное устройство (процессор обработки структур), который обрабатывает лишь ту часть информации, которая определяет взаимные отношения хранимых данных, т. е. структурную часть структур данных. Универсальный центральный процессор обрабатывает информационную составляющую структур данных.*

Отметим, что в вычислительных системах, не обладающих свойством гетерогенности, алгоритмы и скалярные данные, а также структуры данных и алгоритмы их обработки хранятся и обрабатываются одними и теми же универсальными микропроцессорами (рис. 1).

Рис. 1. Вычислительная система с аппаратной поддержкой операций над множествами и структурами данных

На основе гетерогенности вычислений выделяется часть аппаратного обеспечения для реализации операций дискретной математики. Принцип двойственности структур данных обосновывает состав и функциональное назначение частей такой системы.

Принцип опирается на понятие структуры данных, обладающей двойственностью:

$$S = (D, R), \quad (1)$$

где D — множество элементов данных; R — множество отношений между элементами данных. Таким образом, структура определяется как совокупность двух множеств: D и R. Их обработка во многом параллельна и происходит по различным алгоритмам: алгоритмы обработки отношений представляют собой действия по модификации или обходу структуры в соответствии с имеющейся структурной информацией. Например, при обработке древовидной структуры свойства самого дерева определяют алгоритм обхода, алгоритм добавления нового элемента и т. д. Алгоритмы же обработки информационной составляющей решают уже конкретную задачу оптимизации, используя структуры данных. Примером может служить реализация алгоритма Дейкстры для поиска кратчайшего пути на графе, т. е. конкретная задача оптимизации, для которой граф может быть представлен в виде списка смежных вершин. Классический вариант алгоритма оперирует такими понятиями, как множество вершин, множество ребер, инцидентность, и не оговариваются конкретные варианты структур данных, которые эти множества и операции реализуют. Поэтому алгоритм Дейкстры является алгоритмом обработки множества данных D (информационная составляющая), в то время как действия над списками определяют алгоритмы обработки отношений R структур данных.

Важной особенностью новой вычислительной системы является наличие специализированного процессора, выполняющего функции обработки отношений данных R, но не обрабатывающего сами данные. Это устройство носит название процессора обработки структур данных (СП). Действиями над информационной составляющей занимается центральный процессор (ЦП), являющийся универсальным процессорным устройством, реализующим арифметическую и логическую обработку данных.

Таким образом, второй принцип использует понятия структуры данных как совокупности двух множеств и обосновывает независимую их обработку двумя различными устройствами: центральным процессором и процессором обработки структур.

Набор команд дискретной математики DISC (Discrete Instruction Set Computing)

Ключевым вопросом при проектировании любого программно-управляемого устройства является выбор набора команд. Так как целями создания системы является аппаратная поддержка аппарата дискретной математики, были рассмотрены теоретические аспекты реализации алгоритмов дискретной оптимизации и совокупность элементарных преобразований, которые необходимо проводить над множествами. На основе таких понятий, как кванторы, отношения и операции, был определен базовый набор операций системы, представленный в табл. 1.

Исходя из вышесказанного можно отметить, что СП выполняет инструкции высокого уровня, реализующие операции, кванторы и функции дискретной математики. Набор операций СП представляет собой такие действия, как: добавление элемента, поиск элемента, удаление, поиск минимального или максимального элемента, поиск следующего, мощность множества ключей, объединение, дополнение, пересечение [12]. Дополнительно реализованы операции срезов и команды синхронизации потоков команд ЦП и СП. Последняя версия набора команд Leonhard x64 состоит из 20 высокоуровневых кодов операций, перечисленных ниже:

- **Search (SRCH)** выполняет поиск значения, связанного с ключом;
- **Insert (INS)** вставляет пару ключ-значение в структуру. SPU обновляет значение, если указанный ключ уже находится в структуре;

Таблица 1

Соответствие инструкций DISC функциям, кванторам и операциям дискретной математики

Функции, кванторы и операции дискретной математики	Инструкции набора команд DISC
Функция хранения кортежа	INS
Функция отношения элементов множества	NEXT, PREV, NSM, NGR, MIN, MAX
Мощность множества	CNT
Функция принадлежности элемента множеству	SRCH
Добавление элемента в множество	INS
Исключение элемента из множества	DEL, DELS
Исключение подмножества из кортежа	DELS
Включение подмножества в кортеж	LS, GR, LSEQ, GREQ
Отношение эквивалентности множеств	LS, GR, LSEQ, GREQ
Объединение множеств	OR
Пересечение множеств	AND
Разность множеств	NOT
Симметрическая разность множеств	Не реализовано
Декартово произведение множеств	Не реализовано
Булеан множества	Не реализовано

- операция **Delete (DEL)** выполняет поиск указанного ключа и удаляет его из структуры данных;
- последняя версия набора команд Leonhard x64 была расширена двумя новыми инструкциями (NSM и NGR) для обеспечения требований некоторых алгоритмов. Каждая инструкция набора включает до трех операндов: ключа, значения и номера структуры данных. Команды **NSM/NGR** выполняют поиск соседнего ключа, который меньше (или больше) заданного и возвращает его значение. Операции могут быть использованы для эвристических вычислений, где интерполяция данных используется вместо точных вычислений (например, кластеризация или агрегация);
- **Maximum/minimum (MAX, MIN)** ищут первый или последний ключи в структуре данных;
- операция **Cardinality (CNT)** определяет число ключей, хранящихся в структуре;
- команды **AND, OR, NOT** выполняют объединения, пересечения и дополнения в двух структурах данных;
- **срезы (LS, GR, LSEQ, GREQ)** извлекают подмножество одной структуры данных в другую;
- **переход к следующему или переход к предыдущему (NEXT, PREV)** находят соседний (следующий или предыдущий) ключ в структуре данных относительно переданного ключа. В связи с тем что исходный ключ должен обязательно присутствовать в структуре данных, операции NEXT/PREV отличаются от NSM/NGR;
- **удаление структуры (DELS)** очищает все ресурсы, используемые заданной структурой;
- команда **Squeeze (SQ)** дефрагментирует блоки памяти DSM, используемые структурой;
- команда **Jump (JT)** указывает SPU код ветвления, который должен быть синхронизирован с CPU (команда доступна только в режиме MISD).

Приведем пример того, как алгоритм дискретной оптимизации может быть реализован в гетерогенной вычислительной системе с различными типами микропроцессоров. Указанный выше алгоритм Дейкстры [7, 8] позволяет найти кратчайший путь от начальной вершины графа до всех остальных вершин и использует две структуры данных: структуру для представления графа, а также очередь для хранения упорядоченной последовательности вершин. Чтобы понять, какой микропроцессор должен выполнять действия алгоритма в гетерогенной системе, примем следующие обозначения в псевдокоде:

- (D) — это действие основано на операциях дискретной математики и должно выполняться на микропроцессоре Leonhard x64;
- (M) — это операция чтения/записи в память (или регистр), которая может быть инициирована любым процессором в системе;
- (A) — это арифметическая операция, которая может выполняться арифметическим микропроцессором.

Далее приведем псевдокод алгоритма Дейкстры. Пусть граф задан с помощью структуры ***Graph***, в котором для каждой вершины ***v*** указывается поле длины пути ***dist*** и номер предшествующей вершины графа в кратчайшем пути ***previous***. Тогда поиск кратчайшего пути от вершины ***source*** до каждой вершины графа ***Graph*** может быть выполнен следующим образом:

```
FOR ALL вершина v из Graph (D) DO
    dist[v] ← ¥ (M)
    previous[v] ← undefined (M)
END FOR
dist[source] ← 0 (M)
Q ← множество вершин из Graph (D)
WHILE Q (D) DO
    u ← вершина из Q с минимальным dist[] (D)
    Удалить u из Q (D)
    FOR ALL вершина v, смежная u DO (D)
        alt ← dist[u] + ребро(u, v) (A)
        IF alt < dist[v] (A) THEN
            dist[v] ← alt (M)
            previous[v] ← u (M)
        END IF
    END FOR
END WHILE
```

Как мы видим, наиболее трудоемкие и длительные по времени операции алгоритма связаны с выполнением операций дискретной математики (доступу и изменению множеств), а не с арифметической обработкой чисел. Следовательно, совершенствование аппаратного обеспечения для выполнения этих частей рабочей нагрузки способно значительно повысить общую производительность системы. Таким образом, мы используем некоторые новые аппаратные принципы, чтобы частично устранить ограничения, установленные выше:

- реализуем специально разработанный и высокопараллельный микропроцессор обработки структур данных (СП, версия Leonhard x64) для выполнения инструкций набора команд DISC. Этот микропроцессор не требует программного обеспечения для управления памятью, а доступ ко всем структурам данных был реализован исключительно аппаратно;
- используем чрезвычайно короткий конвейер в СП, состоящий из трех стадий, что значительно уменьшает задержку для операций с зависимыми данными. Такой конвейер неэффективен для общей арифметики, но хорошо подходит для обработки множеств;
- разделяем поток инструкций на два потока: арифметическую обработку и инструкции дискретной математики.

Далее рассмотрим более подробно архитектурные принципы построения системы.

Рис. 2. Потоки команд и данных в системе с аппаратной обработкой множеств и структур данных

Архитектура вычислительной системы с набором команд DISC

СП Leonhard x64 осуществляет хранение структур данных в оперативной памяти, обеспечивает их обработку и выдачу скалярных данных в центральный процессор для ее последующей обработки основным вычислительным алгоритмом. Оперативная память, доступная процессору обработки структур, делится на локальную память команд и локальную память структур данных (рис. 2). Вся необходимая для ЦП информация структур данных выдается из СП посредством выполнения команд обработки структур.

На основе представленной на рис. 2 системы был сформулирован принцип структурной декомпозиции системы.

Принцип 3 структурной декомпозиции вычислительной системы для решения задач дискретной оптимизации. *Процессор обработки структур имеет доступ к оперативной памяти, в которой хранятся структуры данных и команды. Результаты выполнения команд направляются в центральный процессор для дальнейшего использования в ходе основного вычислительного алгоритма. Способ представления структур данных в локальной памяти процессора обработки структур должен обеспечивать высокую производительность процессора для всего набора операций.*

Рассматривая возможные варианты организации структур данных в оперативной памяти, необходимо учитывать аппаратную и временную сложность выполнения операций. В связи с тем, что заранее не известны те из них, которые будут наиболее существенными для конкретных алгоритмов, следует выбирать такой вариант внутреннего представления структур, который обеспечивает хорошие результаты для всего спектра операций. Наиболее пригодными как с точки зрения аппаратной сложности, так и по временным характеристикам были признаны сбалансированные деревья. Среди них следует выделить так называемые сильно ветвящиеся B-деревья и B+-деревья. Использование хэш-таблиц не обеспечивает приемлемой вычислительной сложности при определении порядка на множестве, т. е. не позволяет осуществить обход значений. Многочисленные практические примеры, такие как поиск по индексам в базах данных и доступ к информации на flash-накопителях и внешних носителях, показывают, что этот вид деревьев является перспективным для реализации как в программном, так и в аппаратном обеспечении.

Таким образом, третий принцип определяет основы взаимодействия центрального процессора и процессора обработки структур, а также поясняет состав и назначение оперативной памяти СП.

Принцип 4 множественности потоков команд. *В вычислительной системе с аппаратной поддержкой операций над структурами данных один поток данных обрабатывается несколькими потоками команд.*

Для подтверждения четвертого принципа следует более подробно рассмотреть управление вычислительным процессом и форматов представления данных в вычислительной системе с набором команд дискретной математики.

Из принципов 1—3 следует, что процессор обработки структур — это устройство, которое изменяет и анализирует лишь ту часть информации, которая определяет отношения хранимых данных [9, 12]. Основным форматом данных СП, показанным на рис. 3, является тройка чисел (*структура, ключ, значение*), где: поле СТРУКТУРА имеет три разряда; каждое из полей КЛЮЧ и ЗНАЧЕНИЕ имеют 64 разряда (справедливо для версий СП с микроархитектурой Leonhard x64). Используются также дополнительные форматы данных для обмена между ЦП и СП, однако они применяются для синхронизации вычислительных процессов и будут подробно рассмотрены в следующих статьях.

В качестве аппаратной формы представления множеств ключей и значений было выбрано двоичное дерево. В таком случае структурная часть содержит информацию о конфигурации дерева и взаимных отношениях всех ее элементов, а сами значения являются информационной частью структуры данных. В случае, когда необходимо выбрать элемент с заданным ключом, процессор обработки структур выполняет поиск и выборку элемента по структурной составляющей, после чего передает информационную составляющую (*значение*, ассоциированное с ключом) ЦП. Таким образом, действия, исполняемые двумя процессорами, могут выполняться параллельно и независимо, а общий поток команд обработки данных для такой ЭВМ разделяется на два потока: поток команд обработки информационной составляющей и поток обработки структурной составляющей структуры данных. Это позволяет классифицировать предложенную вычислительную систему как систему со многими потоками команд и одним потоком данных.

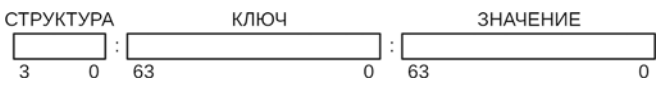

Рис. 3. Основной формат данных процессора обработки структур с микроархитектурой Leonhard x64

Примером решения вычислительной задачи в такой системе может служить некоторая программа обработки данных, на некотором шаге которой требуется найти минимальное значение ключа и увеличить его в *n* раз. Тогда последовательность команд, выполняемых процессорами, будет следующей:

1. Процессор обработки структур выполняет команду поиска минимального элемента в структуре: (*ключ, значение*) = *MIN* (*структура*).
2. Пара значений (*ключ, значение*) передаются в ЦП и удаляется из структуры: *DEL* (*структура, ключ, значение*).
3. ЦП меняет поле ключа: *ключ* = *ключ*·*n*.
4. ЦП передает новую пару значений в процессор обработки структур: *ADD* (*структура, ключ, значение*).

Первая команда приводит к выполнению алгоритма обхода структуры и сравнению ключей для поиска наименьшего. Ключ найденной вершины должен быть увеличен в *n* раз, что будет выполняться арифметико-логическим устройством (т. е. ЦП) и для внесения изменений может потребовать перестроение структуры процессором обработки структур. Очевидно, что действия по поиску вершины с минимальным ключом и последующее перестроение структуры должен выполнить СП, а за изменение значения ключа отвечает арифметически-логическое устройство ЦП.

В предложенной вычислительной системе в памяти процессора обработки структур содержится информация, характеризующая взаимное расположение элементов данных. Иными словами, это информация, указывающая на последовательность в соответствии с их числовыми значениями (структурная составляющая). Это позволяет процессору обработки структур начать выполнение очередной команды по поиску максимального ключа структуры сразу после завершения предыдущей операции удаления, не дожидаясь обработки ключа в АЛУ ЦП. Таким образом, процессор обработки структур выполняет повторяющиеся операции поиска и удаления максимального элемента в соответствии со своим потоком команд, а также выдает поток данных ЦП, который обрабатывает их вторым потоком команд, т. е. ЦП последовательно накапливает значение счетчика стоимости. Потоки команд независимы и определяются лишь характером

решаемой задачи. В результате оба процессора обрабатывают структуру данных одновременно и параллельно. Эта особенность подтверждает, что вычислительная система, соответствующая предложенным принципам, относится к классу "много потоков команд и один поток данных".

Для пояснения следующего принципа отметим, что классификация ЭВМ и вычислительных систем по числу потоков команд и данных была предложена М. Флинном. Согласно работе [17] наличие двух потоков команд и/или данных в вычислительной машине обязательно предполагает организацию архитектурного параллелизма вычислений, что существенно влияет на всю концепцию построения вычислительной машины. В работе [17] предлагаются четыре класса вычислительных машин: с одиночным потоком команд и одиночным потоком данных (ОКОД); с одиночным потоком команд и множественным потоком данных (ОКМД); со множественным потоком команд и одиночным потоком данных (МКОД); со множественным потоком команд и множественным потоком данных (МКМД).

Принцип 5 иерархической вложенности архитектур вычислительных систем. *Принцип иерархической вложенности вычислительных систем различных классов предполагает, что в одной МКМД-системе может быть реализовано несколько МКОД-систем, в МКОД-системе может быть реализовано несколько ОКМД-систем, в одной ОКМД-системе может быть реализовано несколько ОКОД-вычислительных устройств.*

Практические реализации вычислительных машин классов ОКОД, ОКМД и МКМД широко распространены: к классу ОКОД относятся все однопроцессорные ЭВМ; к классу ОКМД относятся устройства векторной и матричной обработки (SIMD SSE, 3DNow, Altivec и другие); к классу МКМД относятся многоядерные вычислительные системы. И только класс МКОД (рис. 4) остается невостребованным в настоящее время.

В вычислительной МКОД-системе взаимодействуют несколько устройств, которые ведут обработку единственного потока данных, причем возможна как последовательная обработка данных, передаваемых от устройства к устройству, так и обработка первоначального потока из локальной памяти всеми устройствами. Этот тип архитектуры соответствует описанной ранее архитектуре системы с аппаратной поддержкой операций дискретной математики, когда одно устройство выполняет обработку информационной составляющей, а второе обрабатывает структурную составляющую. Вся структура данных хранится в локальной памяти и доступна одному или обоим процессорам. В настоящий момент не предложены и не реализованы в промышленных масштабах системы, обладающие такой архитектурой. К классу МКОД иногда относят конвейерные ЭВМ, однако большинство исследователей отрицают это, так как ступени конвейера не являются независимыми и обрабатывают все же один поток команд. К МКОД также относят так называемые систолические матрицы, которые реализовывались в проектах в 60—70-е годы прошлого века. Результаты этих проектов повлияли на развитие векторно-конвейерных процессоров, однако имели ряд функциональных ограничений. В связи с этим считается, что систем данного класса в настоящий момент не существует. Причиной этого явилось отсутствие четко сформулированных принципов построения таких систем, которые были бы востребованы в вычислительных оптимизационных задачах и при обработке данных. Так, в качестве возможных примеров в работе [17] и других источниках называется цифровая фильтрация несколькими устройствами одного сигнала, а также использование нескольких устройств криптографического анализа информации в целях ее декодирования. Указанные примеры являются узкоспециализированными задачами в ограниченных областях, в то время как обработка множеств и структур данных имеет широкое распространение.

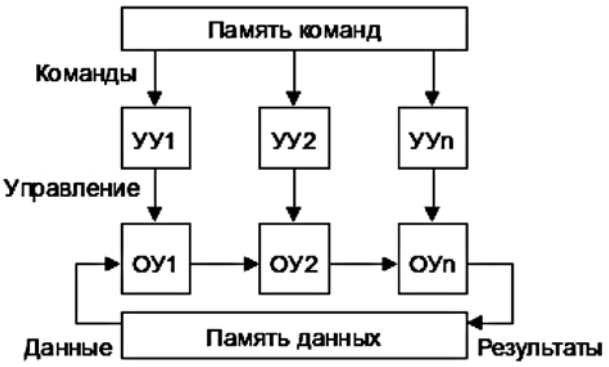

Рис. 4. Вычислительная система с многими потоками команд и одним потоком данных

Таким образом, до представления результатов проекта МГТУ им. Н. Э. Баумана не было предложено разумной и обоснованной концепции применения МКОД-парадигмы вычислений, востребованной при решении практических задач, не было получено ни одного примера реализации всех указанных четырех

Рис. 5. Принцип иерархической вложенности классов вычислительных систем при обработке структур данных и множеств

классов в виде единой вычислительной системы. Поэтому приведенный принцип иерархической вложенности, сформулированный на основе 13 лет кропотливой исследовательской и конструкторской работ по созданию вычислительной системы с набором команд дискретной математики, является важным достижением и сформулирован впервые.

Реализация четвертого класса вычислительных систем (много потоков команд и один поток данных), не реализованного до данного исследования, позволила определить иерархические отношения всех четырех классов (рис. 5):

- в МКОД-системе может быть реализовано несколько ОКМД (один поток команд и много потоков данных) вычислительных устройств, обрабатывающих параллельно элементы данных. Например, на одном уровне B+-дерева могут храниться несколько пар (*ключ, значение*). Все пары могут параллельно обрабатываться одним потоком команд в узлах ОКМД вычислителя;
- устройство с одним потоком команд и многими потоками данных состоит из нескольких примитивных устройств, обрабатывающих по одному потоку данных (ОКОД). Например, такой блок может обрабатывать один ключ в вершине. Таким образом, устройства с одним потоком команд и одним потоком данных при обработке структур данных могут быть иерархически вложены в устройства класса ОКМД;
- для параллельной обработки многих структур данных одновременно могут использоваться несколько взаимосвязанных МКОД-систем, что позволяет получить вычислительную систему с многими потоками команд и многими потоками данных (МКМД).

Таким образом, принцип иерархической вложенности определяет возможность включения структур ЭВМ одного класса в другие, что и было продемонстрировано в ходе проекта реализации вычислительной системы с набором команд дискретной математики.

Принцип 6 работы процессора обработки структур. *Процессор обработки структур может работать в режиме МКОД, режиме сопроцессора и в комбинированном режиме.*

ЦП выполняет функции основного управляющего устройства: загружает задачу из внешней памяти или сети, выделяет место в оперативной памяти ЦП для размещения исходных данных и результатов, выполняет инициализацию содержимого оперативной памяти, запускает задачу на исполнение, передает данные в СП для формирования структур данных или использует уже сформированные структуры, получает результаты обработки структур и использует их в арифметически-логической обработке, передает результаты во внешние устройства и освобождает память. Основной же вычислительный алгоритм разделяется на два потока команд: команды обработки данных, обрабатываемые ЦП, и команды обработки структур, обрабатываемые СП. Для функционирования системы процессоры должны получать следующую информацию (рис. 6):

- ключи, значения и номера структур (в случае хранения нескольких структур одновременно) передаются ЦП в качестве аргументов для выполнения команд в СП;
- результаты выполнения команд (ключи, значения, флаги) передаются из СП в ЦП;
- события синхронизации вычислительных потоков поступают из ЦП в СП и в обратном направлении для координации двух частей алгоритмов;
- команды управления поступают из ЦП в СП для настройки режимов работы и выполнения сервисных процедур;

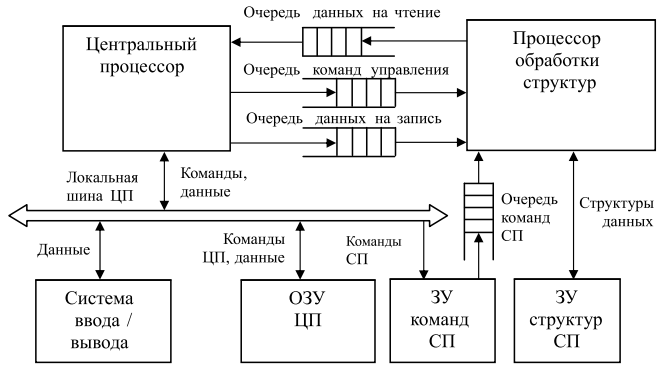

Рис. 6. Пример взаимодействие устройств в системе МКОД

- команды обработки структур данных поступают в СП из локальной памяти команд;
- команды обработки скалярных данных (значений) и сами значения поступают в ЦП из локальной памяти.

Для гетерогенной вычислительной системы, работающей под управлением ЦП, возможны несколько режимов работы других процессоров. Могут быть использованы следующие режимы:

- ***режим сопроцессора*** — управляющие команды для обработки множеств и структур данных формируются в ЦП системы и поступают через очередь управляющих команд;
- ***режим МКОД*** — команды выбираются процессором обработки структур из локального ОЗУ команд. СП использует независимую память для хранения структур, что обеспечивает параллельность и независимость арифметически-логической обработки ЦП и обработки множеств и структур данных в СП.
- ***комбинированный режим*** — команды поступают как от ЦП, так и из локального ОЗУ. Выбор очередной команды осуществляется в порядке их готовности в соответствии с приоритетами. Таким образом, память для хранения структур данных является единым ресурсом системы, предоставляемым условно в виде сервиса с помощью процессора обработки структур другим обрабатывающим устройствам.

Обеспечение работы системы в перечисленных режимах и передачу командной информации и данных можно организовать многими способами, используя шины передачи данных. Известны несколько вариантов топологии вычислительных машин [1], которые могут быть реализованы и для МКОД-систем. В зависимости от назначения, конструктивных особенностей и используемой элементной базы могут быть использованы такие топологии, как общая шина, независимые шины, иерархия шин.

Конкретный вариант топологии может быть выбран по различным критериям, таким как производительность; сложность аппаратной реализации; степень готовности технология разработки программного обеспечения; возможность масштабирования системы. Под масштабируемостью будем понимать технические возможности по увеличению рабочей нагрузки системы посредством увеличения количественных характеристик (дублирования таких узлов, как ЦП и СП, памяти, числа шин и пр.).

Таким образом, шестой принцип определяет способы командного управления вычислительной системой и взаимодействия ЦП и процессора обработки структур.

Производительность микропроцессора Leonhard x64

В проведенных экспериментах был использован микропроцессор Leonhard x64, реализованный на основе ПЛИС FPGA Virtex 6 со встроенным локальным микропроцессором Microblaze. Мы использовали третью версию Leonhard x64 со следующими параметрами: 64-битные ключи и значения; максимальное число ключей в памяти структур 100 млн (100 663 296 записей); максимальное число структур данных: 7; 4 Гбайт объем локальной памяти структур; рабочая частота Leonhard x64 100 МГц. В качестве основы альтернативной системы был выбран ЦП Intel Core i5-4300U CPU, 2 ядра, 4 потока, тактовая частота 1.90GHz, 8 Гбайт ОЗУ. Результаты экспериментов показаны в табл. 2.

Таблица 2

Экспериментальное исследование производительности Leonhard x64

Эксперимент	Число тактов на операцию		Архитектурное ускорение
	Intel Core i5-4300U CPU, 2 ядра, 1.90GHz, 8 Гбайт ОЗУ	Leonhard x64, 100 MHz, 4 Гбайт память структур	
Последовательная вставка 1М элементов во множество, *Average sequential insert rate, ASIR*	3201	741	4,3
Случайная вставка 1М элементов во множество, *Average random insert rate, ARIR*	3298	1425	2,3
Последовательный поиск элементов в 1М множестве, *Average sequential search rate, ASSR*	1903	255	7,5
Случайный поиск элементов в 1М множестве, *Average random search rate, ARSR*	3870	985	3,9
Последовательное удаление элементов из 1М множества, *Average sequential delete rate, ASDR*	2921	360	8,1
Случайное удаление элементов из 1М множества, *Average random delete rate, ASDR*	6485	943	6,9
Обход множества 1М элементов, *LSM 1M keys traversing time, LSMTT*	2017	262	7,7
Поиск ближайшего в 1М множестве, *Average neighbours search rate, ANSR*	3729	1338	2,8

Приведенные эксперименты позволяют комплексно оценить производительность систем при выполнении базовых операций над множествами: вставка, поиск, удаление, обход множеств. Эксперименты построены на основе выполнении действий по формированию множеств или его изменению. В качестве основы выбрано множество, состоящее из 1 млн элементов. Так, эксперимент по последовательной вставке элементов во множество (Average sequential insert rate, ASIR) предполагает создание множества из последовательно возрастающих элементов. Эксперимент по вставке в множество случайных значений отличается тем, что последовательность значений формируется случайном образом.

Как видно из табл. 2, во всех проведенных экспериментах было достигнуто архитектурное ускорение при выполнении базовых операций дискретной математики. Это доказывает целесообразность реализации архитектурных принципов, описанных в данной статье. При этом аппаратная сложность микропроцессора Leonhard x64, выраженная в количестве ресурсов кристалла, составляет около 1 млн вентилей. Это более чем в 1300 раз меньше, чем у микропроцессора Intel Core i5, участвующего в сравнении.

Однако в связи с 19-кратной разницей в тактовых частотах универсальная система на основе микропроцессора Intel Core i5 имеет от 2- до 5-кратного преимущества в сравнении с микропроцессором Leonhard x64 по времени выполнения тестов. Для устранения этого отставания в настоящее время разрабатывается новая версия микропроцессора Leonhard x64 с существенно большим быстродействием.

Выводы

Вычислительная система с набором команд дискретной математики, соответствующая архитектурным принципам класса МКОД, разработана и успешно реализована в МГТУ им. Н. Э. Баумана. Вычислительная система использует параллельную обработку структур данных на основе специализированного аппаратного блока — процессора обработки структур. Принципы функционирования МКОД-системы основаны на параллельной обработке двух взаимосвязанных частей структур данных: информационной и структурной.

В работе приведены первые шесть базовых принципов функционирования вычислительной системы с аппаратной поддержкой операций над множествами и структурами данных:

- принцип гетерогенности вычислений;
- принцип двойственности структур данных;
- принцип структурной декомпозиции вычислительной системы для решения задач дискретной оптимизации;
- принцип множественности потоков команд;
- принцип иерархической вложенности архитектур вычислительных систем:
- принцип работы процессора обработки структур.

В следующих работах будут продемонстрированы принципы хранения и аппаратной обработки информации оперативной памяти *процессора обработки структур*, принципы построения такого процессора, принципы функционирования программного обеспечения в МКОД-системе.

Список литературы

1. **Цилькер Б. Я., Орлов С. А.** Организация ЭВМ и систем. СПб.: Питер, 2011. С. 688.

2. **Kankowski P.** x86 Machine Code Statistics. 2009. URL: http://www.strchr.com/x86_machine_code_statistics (дата обр. 27.03.2018).

3. **Huang I. J., Peng T. C.** Analysis of x86 instruction set usage for DOS/Windows applications and its implication on superscalar design // IEICE Transactions on Information and Systems. 2002. Vol. E85-D, N. 6. P. 929—939.

4. **Davidson A., Baxter S., Garland M., Owens J. D.** Work-Efficient Parallel GPU Methods for Single-Source Shortest Paths *// 2014 IEEE 28th International Parallel and Distributed Processing Symposium*. Phoenix, AZ. 2014. P. 349—359.

5. **Accelerating** High-Performance Computing With FPGAs. White Paper. Altera Corporation. WP-01029-1.1: tech. rep. October 2007. P. 8. URL: https://www.intel.ru/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01029.pdf (дата обр. 19.09.2019).

6. **Integrated** Cryptographic and Compression Accelerators on Intel Architecture Platforms. Intel Corporation: tech. rep. 2013. P. 5. URL: https://www.intel.ru/content/dam/www/public/us/en/documents/solution-briefs/integrated-cryptographic-compression-accelerators-brief.pdf (дата обр. 19.09.2019).

7. **Кнут Д.** Искусство программирования. Т. 3. Сортировка и поиск. М.: Вильямс, 2000. 832 с.

8. **Кормен Т., Лейзерсон Ч., Ривест Р.** Алгоритмы: построение и анализ. М.: МЦНМО, 2000. 960 с.

9. **Попов А. Ю.** Электронная вычислительная машина с многими потоками команд и одним потоком данных: патент 71016 Рос. Федерация. № 2006115810; заявл. 10.05.2006; опубл. 20.02.2008. Бюл. № 5. 1 с.

10. **Попов А. Ю.** Исследование вариантов реализации алгоритмов Крускала и Прима в вычислительной системе с многими потоками команд и одним потоком данных // Наука и образование. МГТУ им. Н. Э. Баумана. Электрон. журн. 2015. № 11. С. 505—527.

11. **Попов А. Ю., Подольский В. Э.** Методика декомпозиции информационного графа программы для организации параллельной обработки данных на ЭВМ МКОД // Вестник МГТУ им. Н. Э. Баумана. Сер. Приборостроение. 2016. Т. 1, № 106. С. 112—128.

12. **Popov A.** An Introduction to the MISD Technology // HICSS50. Proceedings of the 50th Hawaii International Conference on System Sciences, 2017. P. 1003—1012.

13. **Foster C. C.** *Content Addressable Parallel Processors. New York: Van Nostrand Reinhold Company,* 1976. P. 169.

14. **Batcher K. E.** *STARAN parallel processor system hardware // National Computer Conference.* 1974. P. 405—410.

15. **Popov A., Rasheed B.** Network Graph Datastore Using DISC Processor // 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). Saint Petersburg and Moscow, Russia. 2019. P. 1582-1587. doi: 10.1109/EIConRus.2019.8656749.

16. **Popov A., Abdymanapov Ch.,** Motion Planning Algorithms Using DISC // 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). Saint Petersburg and Moscow, Russia. 2019. P. 1844-1847. doi: 10.1109/EIConRus.2019.8657250.

17. **Flynn M. J.** Very High-Speed Computing Systems // Proceedings of the IEEE. 1966. Dec. Vol. 54, N. 12. P. 1901—1909.

A. Yu. Popov, Ph.D., Associate Professor of the Department "Computer Systems and Networks", Bauman Moscow State Technical University, Moscow, e-mail: alexpopov@bmstu.ru

Principles of the Organization of a Heterogeneous Computing System with a Set of Commands of Discrete Mathematics

The computing system with a discrete mathematics instruction set (DISC) is improving in the Bauman Moscow State Technical University. This system consists of microprocessors with different architectures which allow programs to call the hardware functions of large sets and data structures operations, graphs processing and other discrete mathematics functions useful for optimization process. The core of this system is a principally new computing device which called Structure processor Leonhard x64. In the previous works, the results of computer system design were shown, the features of programs execution and data processing were disclosed, the structure of the new processor was presented, and the results of experiments on measuring performance were presented. In this article, we summarize and systematize a number of key principles for constructing a system with a DISC command set, which primarily provide a high level of performance. We consistently state the reason for the introduction of each of the principles and the consequences of its implementation, and demonstrate some examples. The presented article has an important theoretical meaning for the entire project, since many of the results are formulated for the first time.

Keywords: *data structure, computer system with a set of discrete mathematics commands; system with many streams commands and one stream of data; structure processing processor; B + tree*

DOI: 10.17587/it.26.67-79

References

1. **Tsilker B. Ya., Orlov S. A.** Computer Organization and Systems: A Textbook for High Schools, SPb.: Peter, 2011, 688 p. (in Russian).

2. **Kankowski P.** x86 Machine Code Statistics, 2009, available at: http://www.strchr.com/x86_machine_code_statistics (date of access 03.27.2018).

3. **Huang J. J., Peng T. C.** *IE J. Trans.,* 2002, vol. E85-D, no. 6, pp. 929—939.

4. **Davidson A., Baxter S., Garland M., Owens J. D.** Work-Efficient Parallel GPU Methods for Single-Source Shortest Paths, *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, Phoenix, AZ, 2014, pp. 349—359.

5. **Accelerating** High-Performance Computing With FPGAs. White Paper. Altera Corporation. WP-01029-1.1: tech. rep. October 2007, pp. 8, available at: https://www.intel.ru/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01029.pdf (access: 19.09.2019).

6. **Integrated** Cryptographic and Compression Accelerators on Intel Architecture Platforms. Intel Corporation: tech. rep. 2013, pp. 5, available at: https://www.intel.ru/content/dam/www/public/us/en/documents/solution-briefs/integrated-cryptographic-compression-accelerators-brief.pdf (access: 19.09.2019).

7. **Knut D.** The Art of Programming, volume 3. Sorting and searching, Moscow, Williams, 2000, 832 p. (in Russian).

8. **Kormen T., Leyzerson Ch., Rivest R.** Algorithms: construction and analysis, Moscow, MTSNMO, 2000, 960 p. (in Russian).

9. **Popov A. Yu.** Electronic computer with many streams of commands and one data stream: patent 71016 Ros. Federation. No. 2006115810; declare 05/10/2006; publ. February 20, 2008 Bul № 5. 1 p. (in Russian).

10. **Popov A. Yu.** Investigation of options for the implementation of Kruskal and Prym algorithms in a computer system with many streams of commands and one data stream, *Science and Education. MGTU them. N. E. Bauman. Electron. Journals,* 2015, no. 11, pp. 505—527 (in Russian).

11. **Popov A. Yu.** Method of decomposition of the information graph of the program for organizing parallel data processing on an MKOD computer / A. Yu. Popov, V. E. Podolsky // Vestnik MGTU im. N. E. Bauman. Ser. Instrument making. 2016. T. 1, No. 106. pp. 112—128 (in Russian).

12. **Popov A. Yu.** An Introduction to the MISD Technology / A. Popov // HICSS50. Proceedings of the 50th Hawaii International Conference on System Sciences, 2017. pp. 1003—1012 (in Russian).

13. **Foster C. C.** *Content Addressable Parallel Processors, New York, Van Nostrand Reinhold Company,* 1976, 169 p.

14. **Batcher K. E.** *STARAN parallel processor system hardware, National Computer Conference,* 1974, pp. 405—410.

15. **Popov A. Yu., Rasheed B.** Network Graph Datastore Using DISC Processor, St. Petersburg and Moscow, Russia, 2019, pp. 1582—1587. doi: 10.1109 / EIConRus.2019.8656749.

16. **Popov A. Yu., Abdymanapov Ch.** Motion Planning Algorithms Using DISC, St. Petersburg and Moscow, Russia, 2019, pp. 1844—1847. doi: 10.1109 / EIConRus.2019.8657250.

17. **Flynn M. J.** Very High-Speed Computing Systems, *Proceedings of the IEEE,* 1966, vol. 54, no. 12, pp. 1901—1909.