DOI: 10.24892/RIJIE/20250403

Сравнение временных характеристик операционных систем реального времени на микроконтроллерах с архитектурами ядер ARM и RISC-V*

Дубленский Д.С., Махмудов Т.Н., Канунников Е.И.
НИУ ВШЭ,
г. Москва, Российская Федерация
dubds@mail.ru, mahmudov.timur701@gmail.com, kanunnikovEugene1@yandex.ru

Аннотация. **Операционные системы реального времени (ОСРВ - RTOS) занимают важное место в современных технологических процессах, становясь неотъемлемой частью множества отраслей и производственных циклов. Эти системы обладают уникальными характеристиками, позволяющими эффективно управлять ресурсами и обеспечивать высокую скорость отклика на внешние события. Такая особенность базируется на способности ОСРВ перераспределять ресурсы между различными задачами с разным приоритетом, гарантируя выполнение операций в строго определенные временные рамки. Вместе с тем, микроконтроллеры, в частности те, у которых ядра основаны на архитектурах ARM и RISC-V, стали подходящими платформами для реализации ОСРВ благодаря их компактности и наличию соответствующих аппаратных условий. Однако, несмотря на достижения в области технологий, ОСРВ и устройства не лишены ошибок и ограничений, которые оказывают влияние на реальное время реакции на события. Данная проблема имеет долгую историю исследований для архитектуры ARM, так как она получила широкое распространение. Микроконтроллеры с ядром RISC-V постепенно появляются и набирают популярность. Разработчики адаптируют для них ОСРВ. Данная статья призвана исследовать зависимость качества временных характеристик, связанных с событиями в ОСРВ в зависимости от архитектуры ядра микроконтроллера: ARM и RISC-V.**

Ключевые слова: **RTOS, RISC-V, ARM, STM32, MIK32**

Введение

RISC-V, являясь открытым стандартом и модульной архитектурой команд, представляет собой революционное решение в области проектирования процессоров. Она предлагает гибкость, возможность настройки и низкие затраты на разработку, что делает её привлекательной для широкого спектра приложений: от встроенных систем до высокопроизводительных вычислений и искусственного интеллекта. Особенность RISC-V заключается в её открытом исходном коде, который способствует инновациям и сотрудничеству среди разработчиков, а также в возможности добавлять дополнительные расширения для удовлетворения специфических потребностей. Быстрое принятие этой архитектуры в индустрии и её адаптация к новым технологиям подчёркивают её потенциальное влияние на будущее вычислительных решений, конкурируя с такой устоявшейся архитектурой, как ARM [1]. Экосистема RISC-V продолжает развиваться, и её особенности делают её всё более привлекательной для области встраиваемых систем [2]. Проект RISC-V International был запущен в 2010 году в Калифорнийском университете в Беркли командой исследователей во главе с Дэвидом Паттерсоном и Крстэ Асановичем [3]. Ядро RISC-V использовано в отечественном микроконтроллере MIK32 Амур [4], который станет одной из двух платформ для исследования и сравнения.

Процессоры ARM принадлежат к семейству RISC и известны своей высокой энергоэффективностью и простой архитектурой, что делает их идеальными для мобильных устройств, таких как смартфоны, планшеты и другие IoT-устройства. Они обеспечивают высокую производительность при низком потреблении энергии, благодаря чему активно используются в встраиваемых системах [5]. Начало своей истории архитектура берёт в 1985 году, когда Софи Уилсон и Стив Фёрбер создали оригинальную архитектуру Acorn RISC Machine и микроархитектуру ARM1. С тех пор, технология прошла множество этапов, среди которых особо следует отметить разработку ядра Cortex-M, укрепившее позиции на рынке встраиваемых систем и интернета вещей [6]. В связи с популярностью и известностью данного типа ядер в области встраиваемых систем, для сравнения рационально использовать устройство основанная именно на нём: в экспериментах будет использован микроконтроллер STM32F411RET6 [7].

Для данного исследования выбрана FreeRTOS, так как она долго оставалась ведущей операционной системой реального времени (ОСРВ) для устройств с ограниченными ресурсами. Согласно опросу разработчиков, Eclipse IoT и Edge 2021 года, около 30% участников подтвердили, что используют FreeRTOS [8]. Исследования, проведённые EETimes.com и Embedded.com в 2019 году, показали, что 18% респондентов используют FreeRTOS, при этом оно уступает только более общим решениям, таким как Embedded Linux и собственные разработки [9]. Одни из причин её популярности являются длительный срок существования системы и открытый исходный код.

* Статья публикуется по рекомендации программного комитета Всероссийской научно-практической конференции Индустрия 4.0", https://smartindustrycon.ru

ОСРВ должна быть спроектирована так, чтобы обеспечивать баланс между поддержкой широкого набора функций для разработки и эксплуатации приложений реального времени и при этом не нарушать требования по соблюдению временных ограничений и предсказуемости работы системы. Для этого важно, чтобы ОСРВ обеспечивала механизм приоритетов задач, который позволяет эффективно распределять ресурсы между задачами в зависимости от их важности [10]. Кроме того, система должна предоставлять надежные и быстрые механизмы для обмена данными и синхронизации между задачами, чтобы гарантировать целостность данных и своевременность их обработки. Кроме того, среди свойств ОСРВ отмечают совместимость с различными аппаратными платформами [11].

Результатом этого исследования должен стать вывод о зависимости свойств ОСРВ от ядра аппаратной платформы.

Обзор исследований в рассматриваемой области

Прежде чем начинать работу, требуется понять какие идеи, инструменты и метрики используются в аналогичных исследованиях. Необходимо выявить характеристики, которые используются в качестве метрик при сравнении работы ОСРВ. А также инструменты и способы, которые используются для анализа этих характеристик. Кроме того, нужно рассмотреть результаты исследований, посвящённых сравнению или совместному применению RISC-V, ARM и FreeRTOS.

Авторы [12] фокусируются на сравнении производительности платформ RISC-V и ARM в области векторизации вычислений. Автор отмечает значительное ускорение работы алгоритмов на RISC-V благодаря векторизации, что подтверждается улучшением производительности в несколько раз. Особенно впечатляют результаты векторизованных операций по сравнению с их не векторизованными аналогами, что демонстрирует потенциал платформы RISC-V в оптимизации вычислительных процессов. Когда дело доходит до сравнения с ARM, автор подчёркивает, что, несмотря на зрелость и распространённость архитектуры ARM, результаты оказались сопоставимыми. Это особенно интересно, учитывая открытость и относительно недолгую историю RISC-V. Сравнительный анализ проводился с использованием платформы ARM SVE, и итоговый вывод автора "ничья" по производительности свидетельствует о том, что RISC-V уже способен конкурировать с ведущими решениями на рынке. При этом автор подчёркивает преимущества RISC-V в области экспериментальных и открытых решений. В отличие от ARM, закрытая природа которой ограничивает возможности модификаций, открытая архитектура RISC-V предоставляет гибкость для дальнейших улучшений. Это делает её привлекательной для создания новых, доступных и масштабируемых вычислительных систем.

В [13] предоставляется сравнение трёх архитектур, подчёркивая их сильные и слабые стороны в контексте выполнения ресурсоёмких приложений. Особо стоит отметить высокую оценку архитектуры ARM, которая уверенно лидирует как в производительности, так и в энергоэффективности. RISC-V несмотря на то, что находится на более ранней стадии развития, демонстрирует хорошие результаты, что подтверждает её потенциал как открытой и гибкой архитектуры. Авторы работы справедливо делают акцент на её конкурентоспособности, особенно в условиях, где открытость системы является важным фактором. Однако RISC-V пока уступает ARM в ряде аспектов, что ожидаемо для более молодой технологии. Архитектура x86 в данном исследовании показала существенное отставание производительности и энергоэффективности от ARM и RISC-V.

Общее сравнение ядер ARM и RISC-V говорит об интересе и перспективности этой области. На следующем шаге будут рассмотрены существующие сравнения ОСРВ, чтобы узнать об инструментах и характеристиках, которые можно использовать для исследования.

В [14] содержится подробное описание сравнения операционных систем реального времени FreeRTOS, rt-thread, Keil RTX, uC/OS-II, and uC/OS-III. Свои эксперименты автор рассматриваемой статьи ставил на микроконтроллерах STM32F407IG ARM Cortex™-M4 MCU и STM32L053R8 ARM Cortex™-M0+ MCU. Свой список операционных систем автор выбрал на основании результатов ежегодных исследований популярности операционных систем реального времени, представленных в старых исследованиях EETimes.com и Embedded.com. Основная концепция предложенного метода сравнения состоит в синхронизации двух задач посредством реакций на определённые события. Автор объясняет, что для измерения времени реакций служат значения на выводе GPIO, снятые осциллографом, при этом время обработки порта одинаково для всех операционных систем реального времени. Каждая система имеет одинаковое время реакции при доступе к этому порту, который подключен к периферии микроконтроллера. Встроенные системы обычно должны реагировать на внешние события, которые приходят через различные устройства, и отправлять ответы обратно. Поэтому, по мнению автора, использование вывода GPIO для измерения времени реакции помогает лучше понять, как работает реальная система, основанная на операционной системе реального времени.

Автор предоставил четыре сценария сравнения: событие смены задачи, ожидание события переключения задачи, событие без смены задачи и ожидание доступного события.

В качестве механизма синхронизации автор статьи использует основные инструменты операционных систем реального времени [14].

В [15] авторы подробно рассматривает различные аспекты и цели сравнений, выделяя ключевые критерии оценки. Одним из них является поддерживаемый операционной системой набор функций и интерфейсов, или API, который обеспечивает взаимодействие с приложениями. Также важен объём памяти, занимаемый операционной системой и её компонентами, что критично для устройств с ограниченными ресурсами. Как не менее значимый фактор автор выделяет общее время выполнения программы, представленное периодом, необходимым для завершения её работы от начала до конца. Помимо этого, учитывается время переключения контекста, отражающее, сколько времени требуется системе для сохранения состояния текущей задачи и загрузки состояния следующей. Ещё одним критерием выступает джиттер, то есть отклонение фактического времени выполнения задачи от ожидаемого, характеризующее вариации в её завершении.

Авторы работы [16] применяют методологию бенчмаркинга для сравнительного анализа различных ОСРВ, акцентируя внимание на ряде ключевых критериев, которые позволяют глубже понять их производительность. Одним из наиболее значимых по мнению авторов показателей является время переключения задач. Этот параметр характеризует среднее время, затрачиваемое ядром операционной системы на процесс переключения между задачами, с учётом приоритета задач. Такой параметр является критически важным для оценки эффективности ОСРВ, так как он отражает способность системы оперативно реагировать на изменения текущих процессов.

Другим важным параметром, который также выделяется авторами, является время обработки прерываний. Этот показатель демонстрирует, сколько времени требуется задаче с более высоким приоритетом для того, чтобы захватить управление процессором у задачи, имеющей более низкий приоритет. Такой временной интервал имеет особое значение в приложениях, где задачи с высоким приоритетом должны обеспечивать практически мгновенную реакцию на внешние или внутренние события.

Кроме того, в исследовании рассматривается время перемещения семафоров между задачами одного уровня приоритета. Этот параметр показывает, сколько времени требуется операционной системе для передачи семафора, служащего для синхронизации задач, что является важным аспектом межзадачного взаимодействия. Эффективность обработки семафоров непосредственно влияет на производительность и устойчивость ОСРВ в условиях многозадачности.

Ещё одним ключевым аспектом, который анализируется авторами, является задержка межзадачного обмена сообщениями. Этот показатель определяет временной промежуток, который проходит от момента отправки сообщения одной задачей до момента его получения другой задачей. Данный параметр играет ключевую роль в системах, где межзадачное взаимодействие должно быть быстрым и стабильным.

Помимо работ [14], [15] и [16], временные задержки, возникающие в ОСРВ, а также их сравнительный анализ рассматриваются в исследованиях, представленных в работах [17] и [18]. Эти исследования предоставляют дополнительные данные и выводы, которые дополняют научное понимание различных параметров, характеризующих производительность операционных систем реального времени.

Таким образом, существует достаточно богатая научная база, охватывающая ключевые параметры, которые могут быть использованы для детального анализа и оценки эффективности работы ОСРВ. Эти параметры, описанные в упомянутых исследованиях, обеспечивают основу для проведения систематических сравнений различных архитектур и платформ, что позволяет делать выводы об их применимости в условиях реального времени.

В [19] авторы исследуют работу FreeRTOS на ядре RISC-V с помощью SPIKE симулятора, который представляет собой инструмент для эмуляции архитектуры RISC-V и позволяет выполнять отладку программ на этом ядре. SPIKE является точным симулятором, который позволяет разработчикам тестировать и оптимизировать свои программы в среде, приближенной к реальному аппаратному обеспечению, без необходимости иметь физическое устройство.

В качестве метрик в исследовании были выбраны уже знакомые временные характеристики работы ОСРВ, такие как переключение задач и время срабатывания некоторых алгоритмов FreeRTOS.

Рассмотренные исследования из источников [14], [15], [16], [17], [18] и [19] дают возможность построить более полное представление о различных метриках, используемых для сравнения ОСРВ. Во всех этих исследованиях присутствуют общие характеристики ОСРВ, которые авторы анализируют с разных точек зрения. Однако, несмотря на это, важно подчеркнуть, что в данной статье основное внимание уделено именно влиянию аппаратной платформы на различные параметры операционных систем, а не самим операционным системам реального времени. Это ключевое отличие обусловливает тот факт, что некоторые метрики могут не иметь практического смысла в рамках данного исследования. Например, нецелесообразно рассматривать поддержку API FreeRTOS в контексте предстоящих экспериментов, так как этот критерий напрямую зависит от программной части встраиваемой системы. В свою очередь, в рамках данной работы основной акцент делается на анализ аппаратной составляющей системы, что исключает влияние программных факторов, таких как поддержка API, на результаты экспериментов.

Таким образом, данная работа предполагает проведение экспериментального анализа временных периодов, которые могут оказывать значительное влияние на скорость работы ОСРВ: время переключения задач, джиттер, время обработки алгоритмов передачи и принятия семафора, создание и удаление задач.

Методология

В качестве инструмента для временных отметок в экспериментах будут использованы выходы GPIO. Согласно исследованиям, представленным в источниках [14] и [20], такой метод позволяет достаточно точно определить и зафиксировать временные интервалы, что является необходимым для корректного проведения экспериментов и получения точных данных.

Для проведения эксперимента был подготовлен лабораторный стенд, включающий в себя: осциллограф OWON XDS3203A для измерения времени между сигналами, подаваемыми с GPIO, отладочную плату NUCLEO-STM32F411RE на базе микроконтроллера STM32 для генерации и обработки сигналов, а также плату ELBEAR ACE-UNO на базе микроконтроллера MIK32 Амур. Подключение к GPIO осуществлялось с помощью щупа осциллографа. Частота тактирования микроконтроллеров составляла 32 МГц. Блок-схема и фотография стенда представлены на рис. 1. и рис. 2 соответственно.

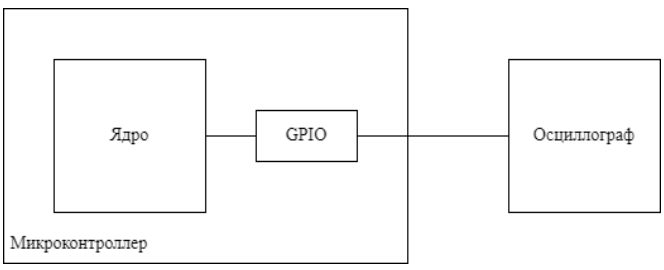

Рис. 1. Блок-схема эксперимента

Рис. 2. Аппаратная составляющая лабораторного стенда

Результаты экспериментов и их обсуждение

Первый эксперимент был направлен на исследование и получение наглядного представления о процессе переключения задач. На рис. 3 представлена столбчатая диаграмма, которая графически иллюстрирует сравнение времени, необходимого для выполнения функционала семафоров, отвечающих за переключение задачи между уровнями приоритетов. Микроконтроллер STM32F411RET6 с ядром на основе широко известной архитектуры ARM обработал алгоритм переключения задачи за 16,9 мкс. В то же время микроконтроллер MIK32 Амур, основанный на более новой архитектуре RISC-V, выполнил этот же алгоритм за 29,4 мкс. Таким образом, разница во времени выполнения составила 17,5 мкс в пользу устройства на базе ARM. В процентном соотношении время, затраченное первым устройством, составляет примерно 57% от времени, затраченного вторым устройством.

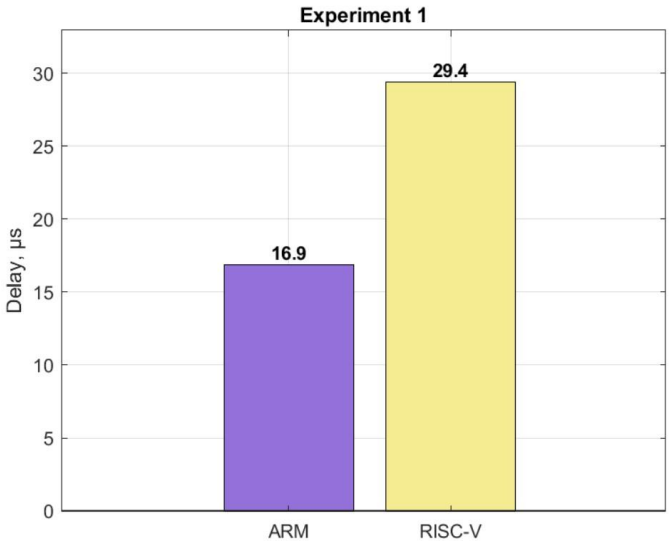

Рис. 3. Задержка переключения задач с использованием семафоров для каждой из архитектур

Следующий этап эксперимента предполагал изучение не только времени задержки выполнения задачи, но также анализ такого важного показателя, как джиттер. Джиттер характеризует возможные колебания или отклонения времени выполнения алгоритмов ОСРВ, что может быть вызвано несовершенствами планировщика задач. Эксперимент продемонстрировал, что один и тот же алгоритм, даже при одинаковых условиях, может быть выполнен за различные временные промежутки. Это связано с потенциальными ошибками и неточностями работы планировщика задач. Диаграмма, представленная на рис. 4, отображает четыре столбца: два столбца соответствуют возможным временным промежуткам выполнения для микроконтроллера STM32F411RET6, а другие два столбца иллюстрируют возможные временные промежутки для микроконтроллера MIK32 Амур. среднее.

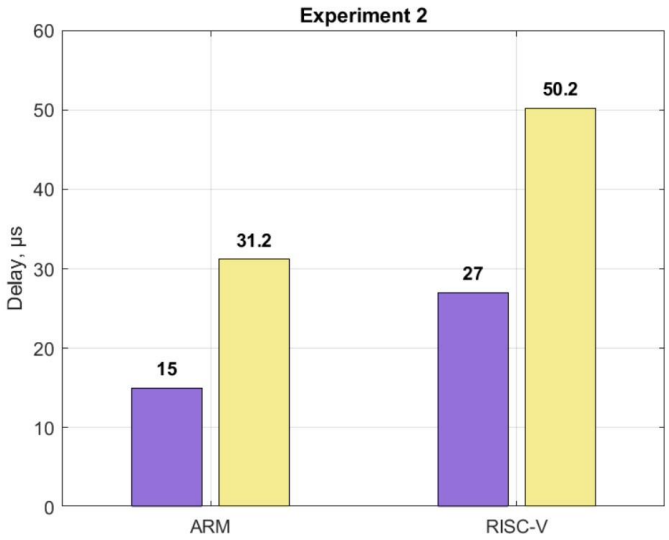

Рис. 4. Задержка переключения задач и возникающий джиттер для каждой из архитектур ядра

Результаты эксперимента показали, что ядро ARM, используемое в STM32F411RET6, может выполнить один и тот же алгоритм, основанный на использовании семафоров, за два временных промежутка: 15 мкс и 31,2 мкс. Таким образом, возможный временной разброс значений для данного устройства составляет 16,2 мкс. В случае использования ядра RISC-V, которое применено в MIK32 Амур, выполнение аналогичного алгоритма возможно либо за 27 мкс, либо за 50,2 мкс. Для данной архитектуры временной разброс значений оказался равным 23,2 мкс. Если рассматривать относительные значения увеличения времени, то для ядра ARM прирост составил 108%, тогда как для ядра RISC-V прирост составил приблизительно 86%.

На следующем этапе исследования, данные которого представлены на рис. 5, основное внимание было сосредоточено на изучении алгоритма захвата сообщений внутри ОСРВ. На данном этапе переключение задач больше не рассматривалось. Анализировался временной промежуток, вызванный выполнением алгоритма принятия семафора, который был выдан той же задачей ранее. Иными словами, исследовался процесс обработки сообщения, связанного с семафором, внутри операционной системы реального времени.

Согласно полученным результатам, микроконтроллер STM32F411RET6 на базе ядра ARM справился с задачей за 3,2 мкс. В то же время микроконтроллер MIK32 Амур с яд-

ром RISC-V выполнил тот же алгоритм за 5,2 мкс. Если рассматривать относительное сравнение, то время выполнения алгоритма микроконтроллером на базе ARM составило около 62% от времени выполнения тем же микроконтроллером на базе RISC-V.

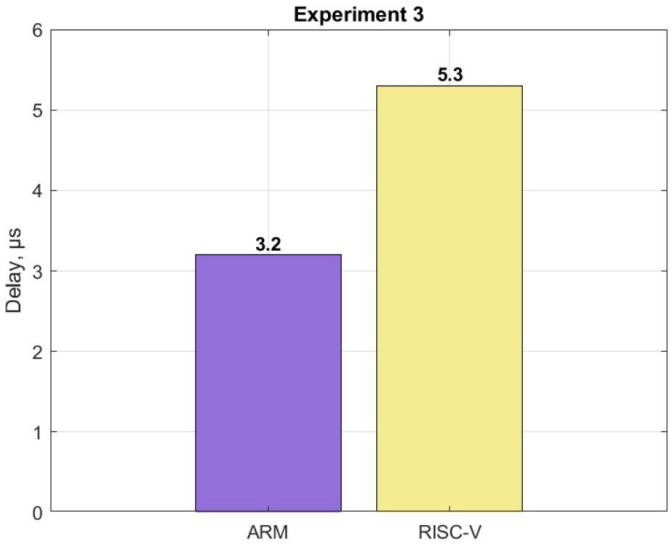

Рис. 5. Задержка, вызванная алгоритмом принятия семафора, для каждой из архитектур ядра

Некоторого времени может потребовать не только принятие сообщения, но и его отправка. Следующий эксперимент позволяет изучить длительности алгоритма выдачи семафора. На рис. 6 представлены результаты наблюдения. Для выдачи семафора микроконтроллеру на базе ядра с архитектурой ARM потребовалось 4,9 мкс. Микроконтроллер на базе ядра с архитектурой RISC-V обрабатывал алгоритм 7,2 мкс. Время для первого аппарата составило около 68% от требуемого времени для второго.

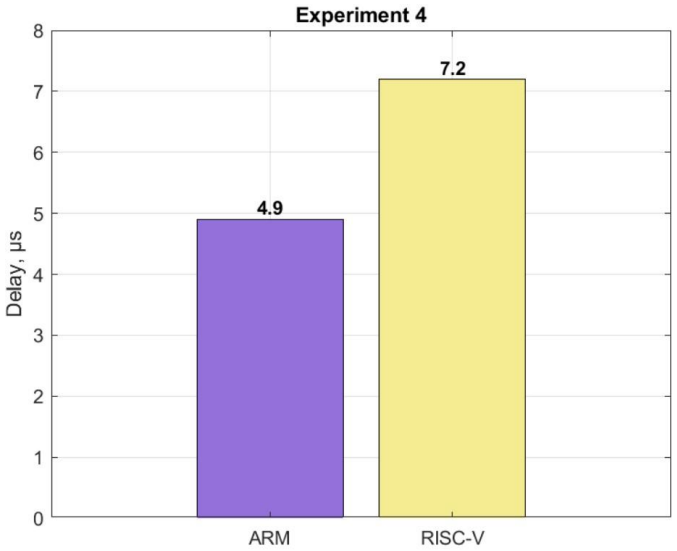

Рис. 6. Задержка, вызванная алгоритмом выдачи семафора, для каждой из архитектур ядра

Два заключающих эксперимента посвящены изучению времени задержки, которая происходит из-за регистрации или удаления новой задачи. По рис. 7 можно убедиться, что ARM затрачивает на обработку алгоритма создания новой задачи 48.4 микросекунд, а RISC-V обрабатывает алгоритм за 35.6 микросекунд. Таким образом, время работы второго аппарата составляет приблизительно 74% от работы второго. Эксперимент, результаты которого представлены на рис. 8, показал, что ядро микроконтроллера на базе архитектуры ARM удалило задачу за 21.5 микросекунд, что составляет примерно 60% от второго результата: ядро на базе RISC-V удалило за 35.6 микросекунд. Любопытно, что алгоритм удаления и алгоритм создания задачи требует одинаковое время задержки у ядра RISC-V.

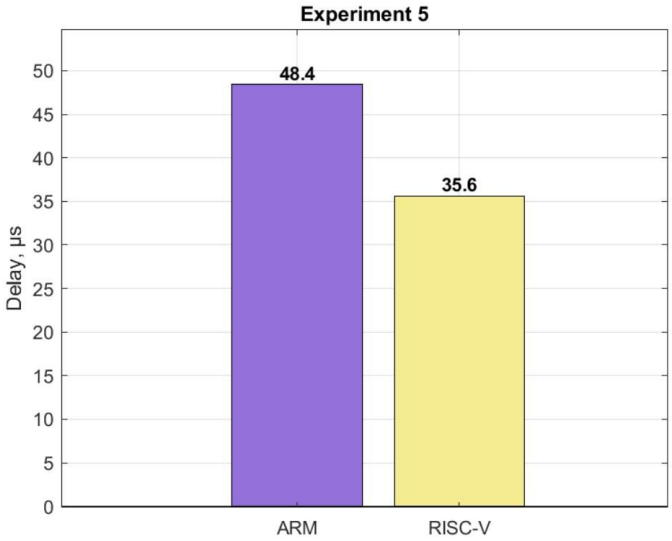

Рис. 7. Задержка создания новой задачи

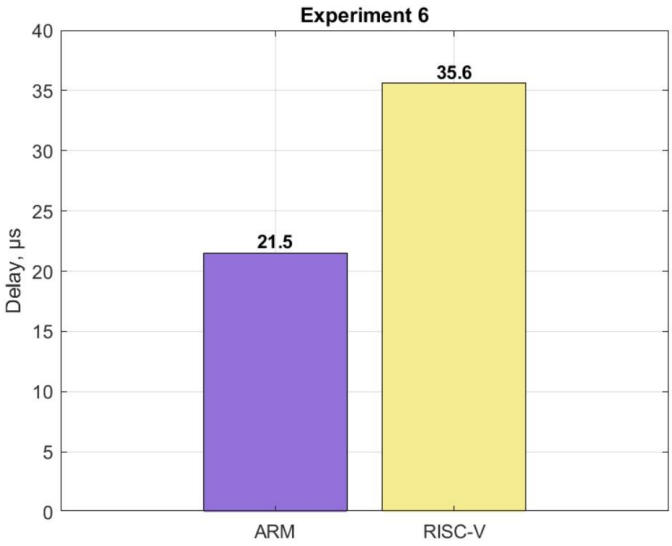

Рис. 8. Задержка удаления задачи

Результаты исследования показали, что в большинстве случаев архитектура RISC-V уступает архитектуре ARM по времени выполнения алгоритмов. Однако при этом разница в производительности, учитывая разное время развития и зрелость этих архитектур, не является критически значительной. Более того, RISC-V уже сейчас демонстрирует повышенную надёжность. Это можно увидеть на примере сложной для планировщика задач операции: время переключения задач для архитектуры ARM увеличилось более чем в два раза, в то время как для архитектуры RISC-V этот временной промежуток увеличился менее чем в два

раза. Кроме того, для RISC-V потребовалось одинаковое время на обработку алгоритма создания и удаления задачи, причём на первый алгоритм потребовалось меньше времени, чем ARM.

Архитектура RISC-V продемонстрировала способность успешно обрабатывать задачи ОСРВ с показателями, незначительно уступающими первой из упомянутых архитектур. Тем не менее, на данный момент устройство на базе ARM обрабатывает алгоритмы ОСРВ с большей эффективностью, чем устройство на базе RISC-V.

Заключение

Исследование заключалось в рассмотрении влияния архитектуры ядра микроконтроллеров на выполнение основных и наиболее критичных функций ОСРВ, таких как переключение задач между различными уровнями приоритетов, передача сообщений между задачами, а также анализ возникающего джиттера в результате несовершенства работы планировщика задач. Для данного исследования в качестве ОСРВ была выбрана активно используемая система FreeRTOS, которая обеспечивает большой набор инструментов для анализа её поведения.

Платформами, которые использовались для детального и всестороннего сравнения работы архитектур, стали два микроконтроллера: STM32F411RET6 и МИК32 Амур. Выбор этих устройств обусловлен ключевыми особенностями архитектур их ядер. STM32F411RET6 базируется на популярной и уже зарекомендовавшей себя архитектуре Arm Cortex-M4, которая является широко используемой в современной промышленности. МИК32 Амур, в свою очередь, основан на более новой, перспективной и стремительно набирающей популярность архитектуре RISC-V, которая привлекает внимание благодаря своей открытости и модульности.

Выбор именно этих двух архитектур для исследования и сравнения обусловлен не только техническими, но и социальными аспектами. Архитектура ARM уже много лет занимает лидирующие позиции в инженерном сообществе благодаря своей стабильности, высокому уровню производительности и широкому применению в различных областях, начиная от мобильных устройств и заканчивая сложными промышленными системами. В то же время архитектура RISC-V только начала активно распространяться, причём её популярность растёт в основном благодаря открытости стандарта и возможности создания локальных решений.

Это исследование ещё раз подчёркивает актуальность изучения современных архитектур и их адаптации к специфическим задачам встраиваемых систем. Несмотря на то, что ARM уже является устоявшимся стандартом, перспективы, предлагаемые архитектурой RISC-V, в будущем могут сыграть важную роль в развитии технологий.

Литература

1. Liu Y. Performance evaluation of various RISC processor systems: A case study on ARM, MIPS and RISC-V / Y. Liu, K. Ye, C.Z. Xu // International Conference on Cloud Computing. – Cham: Springer International Publishing, 2021. – P. 61-74.

2. Saidova J. RISC-V Architecture and its Role in the Near Future // Journal of Advanced Scientific Research. – 2024. – vol. 5, no. 9.

3. Sharma S. Understanding RISC-V: The Open Standard Instruction Set Architecture [Электронный ресурс]. – URL: https://www.wevolver.com/article/risc-v-instruction-set (дата обращения: 10.01.2025).

4. Микрон. Информация о микроконтроллере МИК32 Амур: техническая документация [Электронный ресурс]. – URL:https://mikron.ru/upload/docs/mcu/Information_AMUR_MIK32.pdf (дата обращения: 13.01.2025).

5. Kaiser S. Container technologies for arm architecture: A comprehensive survey of the state-of-the-art / S. Kaiser et al. // IEEE Access. – 2022. – vol. 10. – P. 84853-84881.

6. Grisenthwaite R. The Milestones That Define Arm's Past, Present, and Future // IEEE Micro. – 2021. – vol. 41, no. 6. – P. 58-67.

7. STMicroelectronics. STM32F411xC/E datasheet: ARM Cortex-M4 32-bit MCU [Электронный ресурс]. – URL: https://www.st.com/resource/en/datasheet/stm32f411re.pdf (дата обращения: 15.01.2025).

8. Desbiens F. Operating Systems // Building Enterprise IoT Solutions with Eclipse IoT Technologies: An Open Source Approach to Edge Computing. – Berkeley, CA: Apress, 2022. – P. 243-267.

9. EETimes. Embedded Markets Study 2019. [Электронный ресурс]. – URL: https://www.embedded.com/wp-content/uploads/2019/11/EETimes_Embedded_2019_Embedded_Markets_Study.pdf (дата обращения: 15.12.2024).

10. Luna R. Security and reliability of safety-critical rtos / R. Luna, S.A. Islam // SN Computer Science. – 2021. – vol. 2, no. 5. – 356.

11. Hambarde P. The survey of real time operating system: RTOS / P. Hambarde, R. Varma, S. Jha // 2014 International Conference on Electronic Systems, Signal Processing and Compu-ting Technologies. – IEEE, 2014. – P. 34-39.

12. Cococcioni M. Vectorizing posit operations on RISC-V for faster deep neural networks: experiments and comparison with ARM SVE / M. Cococcioni et al. // Neural Computing and Applications. – 2021. – vol. 33. – P. 10575-10585.

13. Ali W. Exploring Instruction Set Architectural Variations: x86, ARM, and RISC-V in Compute-Intensive Applications // Authorea Preprints. – 2023.

14. Ungurean I. Timing comparison of the real-time operating systems for small microcontrollers // Symmetry. – 2020. – vol. 12, no. 4. – P. 592.

15. Neuhard Y. A Comparison of Real-time Operating Systems for Embedded Computing [Электронный ресурс]. – URL: https://es.cs.rptu.de/publications/datarsg/Neuh22.pdf (дата обращения: 17.01.2025).

16. Mazzi Y. Benchmarking and comparison of two open-source RTOSs for embedded systems based on ARM Cortex-M4 MCU / Y. Mazzi, A. Gaga, F. Errahimi // Indian Journal of Science and Technology. – 2021. – vol. 14, no. 16. – P. 1261-1273.

17. Lejeune E., Bruyninckx H., Kulik A. Real-time Jitter Measurements under ROS2: The Inverted Pendulum Case [Электронный ресурс]. – URL: https://www.researchgate.net/profile/Erwin-Lejeune/publication/350353690_Real-time_Jitter_Measurements_under_ROS2_the_Inverted_Pendulum_case/links/605b4868a6fdccbfea0475fd/Real-time-Jitter-Measurements-under-ROS2-the-Inverted-Pendulum-case.pdf (дата обращения: 17.01.2025).

18. Peng L. Behaviour and performance compari-son between FreeRTOS and μC/OS-III / L. Peng et al. // International Journal of Embedded Systems. – 2016. – vol. 8, no. 4. – P. 300-312.

19. Abraham V. Exploration of FreeRTOS on a RISC-V Architecture / V. Abraham et al. // ICT Systems and Sustainability: Proceedings of ICT4SD 2022. – Singapore: Springer Nature Singapore, 2022. – P. 1-11.

20. Wu H. Two Designs of Automatic Embedded System Energy Consumption Measuring Platforms Using GPIO / H. Wu, C. Chen, K. Weng // Applied Sciences. – 2020. – vol. 10, no. 14. – 4866.

DOI: 10.24892/RIJIE/20250403

Comparison of Time Characteristics of Real-Time Operating Systems (RTOS) on Microcontrollers with Core Architectures ARM and RISC-V

Dublensky D.S., Makhmudov T.N., Kanunnikov E.I.

HSE Moscow Institute of Mathematics and Electronics named after A.N. Tikhonov
Moscow, Russian Federation
dubds@mail.ru, mahmudov.timur701@gmail.com, kanunnikovEugene1@yandex.ru

Abstract. **Real-time operating systems (RTOS) occupy an important place in modern technological processes, becoming an integral part of numerous industries and production cycles. These systems have unique characteristics that allow for effective resource management and ensure a high response speed to external events. This feature is based on the RTOS's ability to redistribute resources between different tasks with varying priorities, guaranteeing the completion of operations within strict time limits. At the same time, microcontrollers, particularly those with cores based on ARM and RISC-V architectures, have become suitable platforms for implementing RTOS due to their compactness and the presence of the necessary hardware conditions. However, despite advancements in technology, RTOS and devices are not without errors and limitations that affect the real-time response to events. This issue has a long history of research for the ARM architecture, as it has become widely used. Microcontrollers with RISC-V cores are gradually emerging and gaining popularity. Developers are adapting RTOS for them. This article aims to investigate the dependence of time-related characteristics, associated with events in RTOS, on the microcontroller core architecture: ARM and RISC-V.**

Keywords: RTOS, RISC-V, ARM, STM32, MIK32

Библиографическое описание статьи

Дубленский Д.С. Сравнение временных характеристик операционных систем реального времени на микроконтроллерах с архитектурами ядер ARM и RISC-V / Д.С. Дубленский, Т.Н. Махмудов, Е.И. Канунников // Машиностроение: сетевой электронный научный журнал. – 2025. – Т.12, №4. – С. 15-21. DOI: 10.24892/RIJIE/20250403

Reference to article

Dublensky D.S., Makhmudov T.N., Kanunnikov E.I. Comparison of time characteristics of real-time operating systems (RTOS) on microcontrollers with core architectures ARM and RISC-V, *Russian Internet Journal of Industrial Engineering*, 2025, vol.12, no.4, pp. 15-21. DOI: 10.24892/RIJIE/20250403
