

Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

К защите допустить:

И. о. заведующего кафедрой  
информатики

\_\_\_\_\_ С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту  
на тему

**«СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ПРОЦЕССОРОВ INTEL  
CORE I5-12450H И AMD RYZEN 7 5800H НА ОСНОВЕ  
ВЫПОЛНЕНИЯ ПРЕОБРАЗОВАНИЙ ФУРЬЕ»**

БГУИР КП 6-05-0612-02 23 ПЗ

Студент

Д. Ю. Себелев

Руководитель

А. А. Калиновская

Нормоконтролёр

А. А. Калиновская

Минск 2025

# СОДЕРЖАНИЕ

Введение .....	5
1 Архитектура вычислительной системы .....	6
1.1 Структура и архитектура вычислительной системы .....	6
1.2 История, версии и достоинства .....	7
1.3 Обоснование выбора вычислительной системы .....	8
1.4 Анализ выбранной вычислительной системы для написания программы .....	9
2 Платформа программного обеспечения .....	10
2.1 Структура и архитектура платформы .....	10
2.2 История, версии и достоинства .....	11
2.3 Обоснование выбора платформы .....	13
2.4 Анализ операционной системы для написания программы .....	14
3 Теоретическое обоснование разработки программного продукта .....	15
3.1 Обоснование необходимости разработки .....	15
3.2 Технологии программирования, используемые для решения поставленных задач .....	15
3.3 Связь архитектуры вычислительной системы с разрабатываемым программным обеспечением .....	15
4 Проектирование функциональных возможностей программы .....	16
Список использованных источников .....	17

## ВВЕДЕНИЕ

В эпоху цифровой трансформации производительность центральных процессоров становится ключевым фактором эффективности вычислительных систем. Развитие технологий приводит к постоянному появлению новых архитектурных решений, направленных на повышение производительности и энергоэффективности.

Вследствие этого возникает многообразие архитектур, использующих различные подходы к построению многоядерных систем. Понимание того, как эти подходы влияют на производительность при решении конкретных вычислительных задач, представляет значительный научный и практический интерес.

Актуальность темы данной курсового проекта обусловлена несколькими ключевыми факторами. Выбранные для сравнения процессоры являются популярными представителями своих линеек в сегменте производительных ноутбуков и построены на принципиально разных архитектурных подходах. Их прямое сравнение позволяет на конкретном примере оценить преимущества и недостатки современных тенденций в проектировании центральных процессоров.

В качестве тестовой нагрузки выбрано быстрое преобразование Фурье (БПФ). Это вычислительно интенсивная задача, эффективность выполнения которой напрямую зависит от архитектурных особенностей процессора. Поэтому анализ производительности БПФ является репрезентативным тестом для широкого класса научных и инженерных приложений.

Целью данной курсового проекта является проведение сравнительного анализа производительности процессоров *Intel Core i5-12450H* и *AMD Ryzen 7 5800H* при выполнении алгоритмов преобразования Фурье.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1 Изучение технических характеристик и архитектурных особенностей рассматриваемых процессоров.
- 2 Разработка программы для выполнения преобразования Фурье.
- 3 Проведение вычислительных экспериментов с различными размерами данных, числом ядер и потоков.
- 4 Анализ полученных данных для оценки производительности процессоров.

В результате исследования будут построены графики, демонстрирующие зависимость времени выполнения преобразования Фурье от размера данных, что позволит провести детальный анализ производительности рассматриваемых процессоров.

# 1 АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

## 1.1 Структура и архитектура вычислительной системы

Процессоры *AMD Ryzen 7 5800H* и *Intel Core i5-12450H* основаны на принципиально разных архитектурных подходах к построению вычислительных систем и ориентированы мобильные вычислительные системы.

Таблица 1.1 – Сравнение характеристик

Характеристики	<i>AMD Ryzen 7 5800H</i>	<i>Intel Core i5-12450H</i>
Кодовое имя архитектуры	<i>Cezanne-H (Zen 3)</i>	<i>Alder Lake-H</i>
Физические ядра	8	8
Количество потоков	16	12
<i>L1</i> кэш на ядро	64 КБ	80 КБ + 96 КБ
<i>L2</i> кэш на ядро	512 КБ	1.25 МБ + 2 МБ <sup>1)</sup>
<i>L3</i> кэш	16 МБ	12 МБ
Тактовая частота	3.2 - 4.4 ГГц	2 - 4.4 ГГц
Техпроцесс	7 нм	10 нм
Встроенная графика	<i>AMD Radeon RX Vega 8</i>	<i>Intel UHD Graphics Xe G4</i>
Поддерживаемая память	DDR4	DDR4, DDR5
<i>TDP</i>	45 Вт	45 Вт
<i>Hyper-Threading/SMT</i>	+	+

Процессор *AMD Ryzen 7 5800H*, использует монолитную архитектуру *Zen 3*, произведенную по техпроцессу 7 нм [1]. Данная архитектура предполагает использование 8 ядер, которые с помощью технологии *SMT (Simultaneous Multithreading)* обеспечивают обработку 16 потоков. Максимальная тактовая частота составляет 4.4 ГГц. Процессор оснащен 16 МБ кэш-памяти третьего уровня (*L3*), а его тепловой пакет (*TDP*) составляет 45 Вт. За обработку графики отвечает встроенный видеоадаптер *AMD Radeon RX Vega 8*. Он построен на проверенной временем архитектуре *Vega*, имеет в своем составе 8 вычислительных блоков и способен работать на частоте до 2.0 ГГц.

Процессор *Intel Core i5-12450H* базируется на гибридной архитектуре *Alder Lake*, изготовленной по техпроцессу 10 нм [2]. В его состав входят 4 производительных *P*-ядра и 4 энергоэффективных *E*-ядра, что обеспечивает поддержку 12 потоков. Распределение нагрузки между ядрами осуществляется аппаратным планировщиком *Intel Thread Director*. Пиковая частота производительных ядер достигает 4.4 ГГц, объем кэш-памяти *L3* составляет 12 МБ, а тепловой пакет (*TDP*) — 45 Вт. В него

<sup>1)</sup>Е-ядра используют общий 2 МБ кластер

интегрировано графическое ядро *Intel UHD Graphics Xe G4*, принадлежащее к более современной архитектуре *Xe-LP*. Оно включает 48 исполнительных блоков, но работает на более низкой пиковой частоте в 1.2 ГГц.

Таким образом, ключевое различие между процессорами заключается в подходе к многоядерности. Архитектура *Zen 3*, позволяющая процессору оперировать 16 потоками, потенциально лучше подходит для тяжелых параллельных вычислений, где важна максимальная производительность каждого потока, а увеличенный объем кэш-памяти *L3* способствует ускорению работы с большими наборами данных. В свою очередь, архитектура *Alder Lake* использует более современную модель с разнородными ядрами, которая обеспечивает гибкость и энергоэффективность за счет распределения задач между производительными и эффективными ядрами с помощью аппаратного планировщика *Intel Thread Director*. Однако эффективность такого подхода сильно зависит от оптимизации операционной системы и программного обеспечения под гибридную архитектуру. Эти фундаментальные различия в дизайне и станут основой для последующего сравнительного анализа их производительности в реальных задачах.

## 1.2 История, версии и достоинства

Архитектура *Zen 3*, на которой базируется *Ryzen 7 5800H*, является кульминацией многолетних усовершенствований. Её история начинается с революционной архитектуры *Zen* (2017), которая ознаменовала возвращение *AMD* в сегмент высокопроизводительных процессоров. Последующая *Zen 2* привнесла 7-нм техпроцесс и чиплетный дизайн, значительно увеличив количество ядер. *Zen 3* стала её логическим развитием, сфокусированным на повышении показателя *IPC* (числа инструкций за такт) и снижении задержек. Ключевым нововведением стало объединение ядер и кэш-памяти *L3* в единый комплекс, что улучшило межъядерное взаимодействие. Основные достоинства этой архитектуры заключаются в её монолитной структуре с восемью одинаково производительными ядрами, что обеспечивает предсказуемую производительность. Благодаря технологии *SMT* процессор обрабатывает 16 потоков, что делает его отлично подходящим для параллельных вычислений, а большой объединенный *L3* кэш ускоряет работу с крупными наборами данных.

В свою очередь, архитектура *Alder Lake*, лежащая в основе *Core i5-12450H*, знаменует собой самый радикальный сдвиг для *Intel* за последнее десятилетие. Ей предшествовал долгий период доминирования архитектуры *Skylake* и её многочисленных итераций (от *Kaby Lake* до *Comet Lake*), производившихся по 14-нм техпроцессу. Столкнувшись с растущей конкуренцией, *Intel* разработала *Alder Lake* как комплексный ответ. Эта архитектура не только перешла на новый техпроцесс *Intel 7* (10 нм),

но и впервые в настольном сегменте представила гибридную модель. Она сочетает в себе производительные ядра (*P-cores*) для выполнения сложных задач и энергоэффективные ядра (*E-cores*) для фоновых процессов. Управление распределением задач между ними осуществляет аппаратный планировщик *Intel Thread Director*. Преимущества такого подхода заключаются в гибкости и энергоэффективности, поскольку система адаптируется к нагрузке, оптимизируя энергопотребление. *P-ядра* обеспечивают высокую однопоточную производительность, а поддержка современных технологий, таких как память *DDR5*, гарантирует высокую пропускную способность.

### 1.3 Обоснование выбора вычислительной системы

Выбор процессоров *AMD Ryzen 7 5800H* и *Intel Core i5-12450H* для сравнительного анализа обусловлен тем, что они, принадлежа к одному классу мобильных решений с одинаковым тепловым пакетом, основываются на принципиально разных архитектурных философиях. Их сопоставление позволяет наглядно оценить сильные и слабые стороны двух доминирующих подходов в современном процессоростроении: «классического» монолитного с однородными ядрами и нового гибридного с разнородными. Ключевым аспектом для сравнения является реализация технологий виртуальной многопоточности: *SMT* (*Simultaneous Multithreading*) у *AMD* и *Hyper-Threading* у *Intel*. В процессоре *Ryzen 7 5800H* технология *SMT* применяется ко всем восьми однородным ядрам, удваивая количество потоков до 16. Это создает симметричную и предсказуемую среду для параллельных вычислений. В свою очередь, в *Intel Core i5-12450H* технология *Hyper-Threading* активна только на четырех производительных *P-ядрах*, в то время как четыре энергоэффективных *E-ядра* не поддерживают ее. В результате общее число потоков составляет 12 (8 от *P-ядер* и 4 от *E-ядер*), что создает асимметричную структуру. Именно анализ того, как операционная система и приложение для тестов будут распределять нагрузку в этих двух принципиально разных моделях многопоточности, и является одной из центральных задач исследования. Наконец, принадлежность процессоров к одному сегменту и схожий уровень энергопотребления создают равные условия для анализа именно архитектурных различий, что делает сравнение объективным.

Выбор данных моделей также обусловлен их широкой распространенностью на рынке, что делает исследование актуальным для многих пользователей. В качестве тестовой нагрузки было выбрано быстрое преобразование Фурье (БПФ) — фундаментальный алгоритм, чувствительный к производительности ядер и подсистемы памяти. Его параллельная природа позволяет эффективно оценить, как каждая архитектура справляется с масштабированием вычислительной нагрузки.

## 1.4 Анализ выбранной вычислительной системы для написания программы

Анализ производительности выбранных процессоров целесообразно начать с рассмотрения их типичных результатов в популярных синтетических бенчмарках, таких как *Cinebench R24*, *Geekbench 5* и *Geekbench 6*. Эти тесты являются индустриальным стандартом для оценки производительности *CPU* и позволяют сделать предварительные выводы о поведении процессоров в различных сценариях.

*Cinebench R24* — это бенчмарк, который оценивает производительность процессора путем рендеринга сложной трехмерной сцены [3]. Тест доступен в двух режимах: однопоточном, который измеряет производительность одного ядра, и многопоточном, который задействует все доступные процессорные потоки. Этот бенчмарк хорошо показывает пиковую производительность *CPU* в задачах, которые эффективно распараллеливаются.

*Geekbench 6* — это кросс-платформенный бенчмарк, который измеряет производительность системы в задачах, имитирующих реальные сценарии использования, от просмотра веб-страниц до обработки изображений и машинного обучения [4]. Он также предоставляет результаты для однопоточного и многопоточного режимов, что позволяет составить комплексное представление о возможностях процессора.

Сравнивая результаты тестирования, можно отметить, что в многопоточном тесте *Cinebench R24* процессор *AMD Ryzen 7 5800H* опережает *Intel Core i5-12450H* примерно на 20%, что показывает его эффективность при работе в многопотоке, в то время как в однопоточном режиме разница составляет 10% в пользу *Intel Core i5-12450H*, что подчеркивает его высокую производительность при выполнении задач с использованием одного ядра.

Тест *Geekbench 6* показывает довольно схожие результаты. В многопоточном тесте *Ryzen 7 5800H* вырывается вперед на 16%, оставляя позади *Core i5-12450H*. Но в однопоточном режиме по-прежнему лидирует *Core i5-12450H* с отрывом в 12.5%, что доказывает его преимущество в задачах, не требующих использования нескольких ядер.

Экстраполируя эти результаты на задачу вычисления БПФ, следует ожидать аналогичной картины. В однопоточном режиме *Intel Core i5-12450H*, вероятнее всего, покажет лучшее время. В многопоточном же режиме, где алгоритм может быть эффективно распараллелен, преимущество будет за *AMD Ryzen 7 5800H* благодаря его превосходству в многопоточных вычислениях.

## 2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

### 2.1 Структура и архитектура платформы

В качестве операционной системы для разработки и тестирования программного обеспечения была выбрана ОС *Linux*. *Linux* — это семейство *Unix*-подобных операционных систем с открытым исходным кодом, основанных на одноименном ядре. Впервые выпущенное Линусом Торвалдсом в 1991 году, ядро *Linux* стало основой для множества операционных систем, известных как дистрибутивы.

Ключевыми особенностями *Linux*, определившими его выбор, являются высокая стабильность, безопасность, гибкость настройки и модульность. Эти качества сделали его доминирующей системой в серверном сегменте, суперкомпьютерах и встраиваемых системах, а также популярной платформой для разработчиков программного обеспечения.

Архитектура *Linux* строится на нескольких фундаментальных принципах, унаследованных от *Unix*, но получивших собственное развитие:

1 Монолитное ядро с модульной структурой. Ядро *Linux* является монолитным, то есть все основные сервисы работают в едином адресном пространстве (пространстве ядра), имея прямой доступ к аппаратному обеспечению. Однако оно поддерживает динамическую загрузку модулей (драйверов), что позволяет сохранять компактность, обеспечивая широкую поддержку оборудования.

2 Системные библиотеки. Они формируют ключевой промежуточный слой между пользовательскими приложениями и ядром системы. Программы, как правило, не используют системные вызовы напрямую. Вместо этого они обращаются к функциям из системных библиотек, самой известной из которых является *GNU C Library (glibc)*. Такой подход упрощает разработку и повышает переносимость программного обеспечения.

3 Многозадачность и многопользовательский режим. Система изначально проектировалась для одновременной работы нескольких пользователей и параллельного выполнения множества процессов в пространстве пользователя. Ядро использует механизм вытесняющей многозадачности для справедливого распределения процессорного времени.

4 Стандарт иерархии файловой системы (*FHS*). Структура каталогов в *Linux* строго стандартизирована. *FHS* определяет назначение основных директорий, таких как */bin*, */etc*, */home* и */var*, что обеспечивает предсказуемость и совместимость программного обеспечения.

5 Разделение пространства. Система четко разделена на пространство ядра (*kernel space*) и пространство пользователя (*user space*), где выполняются все прикладные программы. Взаимодействие между ними

происходит через строго определенный интерфейс системных вызовов, что повышает стабильность и безопасность системы.

6 Широкая поддержка оборудования. Огромное количество драйверов для самого разного оборудования включено непосредственно в исходный код ядра *Linux*. Это, в сочетании с модульной структурой, позволяющей подгружать только необходимые драйверы, обеспечивает широкую поддержку устройств «из коробки». Активное участие сообщества и производителей оборудования постоянно расширяет этот список.

На рисунке 2.1 изображена архитектура операционной системы *Linux*.

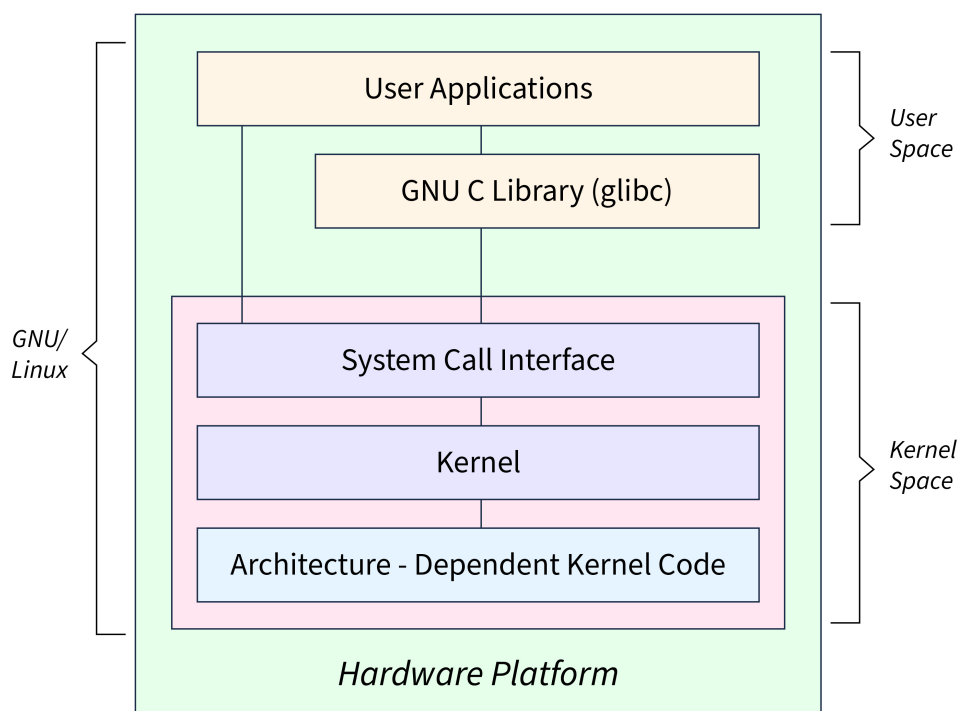


Рисунок 2.1 – Архитектура операционной системы Linux

## 2.2 История, версии и достоинства

Возникновение операционной системы *Linux* датируется 1991 годом и связано с инициативой финского исследователя Линуса Торвальдса. Изначальной целью разработки было создание *UNIX*-подобного ядра, которое было бы свободно от лицензионных ограничений, присущих существовавшим на тот момент системам, таким как *MINIX*. Первая официальная версия ядра, 0.01, была представлена в сентябре 1991 года.

Фундаментальным этапом в развитии *Linux* стала его интеграция с набором системных утилит и библиотек проекта *GNU*, инициированного Ричардом Столлманом в 1983 году. Проект *GNU* ставил своей целью создание полностью свободной *UNIX*-совместимой операционной системы, и к началу 1990-х годов располагал практически всеми необходимыми компонентами (т.н. *userland*), за исключением низкоуровневого ядра.

Объединение ядра *Linux* и пользовательского окружения *GNU* привела к формированию полноценной операционной системы, которую корректно именовать *GNU/Linux*. Перевод проекта под юрисдикцию лицензии *GNU General Public License (GPL) v2* в 1992 году стал решающим фактором, обеспечившим правовую основу для свободного распространения, модификации и коллективной разработки.

Разработка *Linux* в значительной степени была вдохновлена его предшественниками, в первую очередь *UNIX* и *MINIX*. Система *UNIX*, созданная в лабораториях *Bell Labs* в конце 1960-х — начале 1970-х годов, заложила фундаментальные принципы проектирования, ставшие стандартом для современных операционных систем: иерархическая файловая система, концепция процессов и функциональный интерфейс командной строки. Её портируемость и многозадачность сделали её стандартом в академической и коммерческой среде.

Однако со временем лицензирование *UNIX* становилось всё более ограничительным. Это побудило Эндрю Таненбаума в конце 1980-х годов создать *MINIX* — *UNIX*-подобную операционную систему на основе микроядра, предназначенную для образовательных целей. Её исходный код был доступен, но лицензия всё ещё накладывала ограничения на свободное изменение и распространение. Именно опыт работы Линуса Торвалдса с *MINIX* и его желание создать по-настоящему свободную *UNIX*-подобную операционную систему с открытым исходным кодом, которую он мог бы адаптировать для собственного оборудования, и послужили прямой причиной для создания ядра *Linux*.

Ядро *Linux* само по себе не является готовой к использованию операционной системой. Оно служит основой для дистрибутивов — комплексных программных сборок, которые включают также системные утилиты, библиотеки и прикладное программное обеспечение. Дистрибутивы, такие как *Debian*, *Fedora* и *Ubuntu*, различаются системами управления пакетами, набором ПО и целевым назначением, предоставляя пользователям готовые к работе системы для различных задач.

Ключевые достоинства *Linux* как платформы для разработки и научных вычислений включают следующие аспекты:

1 Доступность исходного кода: Лицензия *GPL* гарантирует право на изучение, модификацию и распространение исходного кода. Это способствует проведению аудита безопасности, адаптации системы под специфические задачи и ускоряет цикл внедрения новых технологий.

2 Высокая стабильность и надежность: Архитектурные принципы, унаследованные от *UNIX*, обеспечивают высокую отказоустойчивость и способность к длительной непрерывной работе, что является критически важным для серверных систем и длительных вычислительных экспериментов.

3 Модель безопасности: Система разграничения прав доступа для пользователей, групп и прочих субъектов, а также механизм изоляции процессов, значительно снижают риски, связанные с вредоносным программным обеспечением.

4 Гибкость и модульность: Возможность детальной конфигурации всех компонентов системы, от параметров ядра до выбора графической среды, позволяет оптимизировать платформу для конкретных аппаратных ресурсов и прикладных задач.

5 Эффективное управление ресурсами: Планировщик задач ядра *Linux* и подсистема управления памятью обеспечивают эффективное использование вычислительных ресурсов, что часто выражается в более высокой производительности по сравнению с альтернативными ОС на идентичном оборудовании.

## 2.3 Обоснование выбора платформы

Для проведения исследования в качестве дистрибутива операционной системы *GNU/Linux* был выбран *Arch Linux*. Этот выбор обусловлен рядом ключевых преимуществ, которые делают его оптимальной платформой для проведения точного и воспроизводимого сравнительного анализа производительности.

Центральным принципом *Arch Linux* является минимализм. Система поставляется в виде базового набора компонентов, предоставляя пользователю полный контроль над устанавливаемым программным обеспечением. Такой подход позволяет создать «чистую» среду для тестирования, свободную от фоновых процессов и служб, которые могли бы вносить помехи в результаты измерений. Это гарантирует, что полученные данные о времени выполнения алгоритма БПФ будут максимально точно отражать производительность самих процессоров, а не влияние стороннего ПО.

Другим важным достоинством является модель непрерывных обновлений (*rolling release*). *Arch Linux* предоставляет доступ к самым последним стабильным версиям программного обеспечения, включая ядро *Linux*, компиляторы (*GCC*, *Clang*) и системные библиотеки. Использование новейшего ядра особенно актуально для данного исследования, поскольку оно включает последние оптимизации планировщика задач, что критически важно для корректной работы с гибридной архитектурой процессора *Intel Core i5-12450H* и его технологией *Thread Director*. Современные компиляторы, в свою очередь, могут предложить улучшенные возможности для оптимизации кода, что напрямую влияет на итоговую производительность.

Ключевым преимуществом *Arch Linux* является его философия, предполагающая полный контроль над конфигурацией системы.

Прозрачность и простота структуры дистрибутива позволяют точно документировать и, что более важно, воспроизводить тестовое окружение с высокой степенью достоверности. Это, в сочетании с минимализмом и доступом к новейшему программному обеспечению, обеспечивает создание изолированной и предсказуемой среды, необходимой для получения объективных и сопоставимых результатов производительности.

## 2.4 Анализ операционной системы для написания программы

Анализ операционной системы в контексте поставленной задачи заключается в определении конкретных инструментов и методологий, которые будут использованы для разработки, компиляции и выполнения программы для тестирования. Выбранная платформа, *Arch Linux*, предоставляет все необходимые средства для проведения глубокого и точного исследования.

Компиляция программы будет производиться с помощью компилятора *GCC* последней версии, доступной в дистрибутиве. Для достижения максимальной производительности компиляция будет выполняться с флагами, активирующими высокий уровень оптимизации, такие как *-O3* и *-march=native*. Флаг *-march=native* заставляет компилятор генерировать код, оптимизированный под специфический набор инструкций конкретного процессора, на котором производится сборка. Это позволит в полной мере задействовать архитектурные преимущества каждого из тестируемых *CPU*.

Такой подход позволит детально изучить вклад каждого типа ядер в общую производительность и оценить эффективность планировщика задач.

Для получения более глубоких данных о производительности, помимо простого измерения времени выполнения, планируется использование утилиты *perf*. Этот инструмент позволяет собирать данные с аппаратных счетчиков производительности процессора, такие как количество выполненных инструкций, промахи кэша и неверно предсказанные переходы. Анализ этих метрик даст возможность сделать более обоснованные выводы о причинах наблюдаемых различий в производительности.

Таким образом, *Arch Linux* предоставляет полный набор инструментов для разработки, низкоуровневой оптимизации и детального анализа производительности, что делает его идеальной средой для решения задач данного исследования.

### **3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА**

#### **3.1 Обоснование необходимости разработки**

#### **3.2 Технологии программирования, используемые для решения поставленных задач**

#### **3.3 Связь архитектуры вычислительной системы с разрабатываемым программным обеспечением**

## **4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ**

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] NotebookCheck. AMD Ryzen 7 5800H Specs [Электронный ресурс]. – Режим доступа : <https://www.notebookcheck.net/AMD-Ryzen-7-5800H-Processor-Benchmarks-and-Specs.512759.0.html>. – Дата доступа : 23.09.2025.

[2] TechnicalCity. Intel Core i5-12450H Specs [Электронный ресурс]. – Режим доступа : <https://technical.city/en/cpu/Core-i5-12450H>. – Дата доступа : 23.09.2025.

[3] Cinebench benchmark [Электронный ресурс]. – Режим доступа : <https://cinebench.net/>. – Дата доступа : 28.09.2025.

[4] Geekbench 6 [Электронный ресурс]. – Режим доступа : <https://www.geekbench.com/>. – Дата доступа : 28.09.2025.