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 AMDRYZEN 5 3500U И AMD RYZEN 7 7700 НА ПРИМЕРЕАЛГОРИТМОВ ПОИСКА В ШИРИНУ (BFS) И В ГЛУБИНУ (DFS)**

БГУИР КП 6-05 0612 02 020 ПЗ

Студент М. М. Петроченко

Руководитель А. А. Калиновская

Нормоконтролёр А. А. Калиновская

Минск 2025

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc212412318)

[1 Архитектура вычислительной системы 6](#_Toc212412319)

[1.1 Выбор вычислительной системы 6](#_Toc212412320)

[1.2 История, версии и достоинства выбранной архитектуры 7](#_Toc212412321)

[2 Платформа программного обеспечения 14](#_Toc212412322)

[2.1 Структура и архитектура платформы 14](#_Toc212412323)

[Заключение 17](#_Toc212412324)

[Список литературных источников 18](#_Toc212412325)

Введение

В условиях стремительного развития вычислительных технологий эффективность аппаратного обеспечения остается одним из ключевых факторов, определяющих возможности современных программных решений. Особую актуальность приобретает сравнительный анализ производительности процессоров, поскольку это позволяет оптимально подобрать аппаратную платформу под конкретные задачи. В рамках данного курсового проекта рассматривается сравнительная оценка производительности двух моделей процессоров компании AMD – Ryzen 5 3500U и Ryzen 7 7700 – на основе реализации классических алгоритмов поиска в ширину (BFS) и поиска в глубину (DFS).

Цель данной работы – провести сравнительный анализ производительности процессоров AMD Ryzen 5 3500U и AMD Ryzen 7 7700 при выполнении алгоритмов поиска в ширину (BFS) и поиска в глубину (DFS). Это позволит выявить влияние технических характеристик процессоров на эффективность выполнения классических алгоритмических задач.

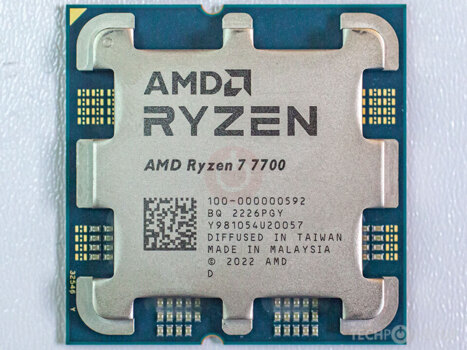
Для достижения цели планируется решить следующие задачи:

1. Разработать условия проведения измерений, позволяющие максимально объективно провести сравнительную оценку производительности.
2. Подготовить реализации алгоритмов BFS и DFS для тестирования.
3. Провести измерения времени их выполнения на обоих процессорах в разработанных условиях и выполнить сравнительный анализ результатов с учетом особенностей архитектуры и технических параметров исследуемых моделей.

Пояснительная записка оформлена в соответствии с  
СТП 01-2024 [1].

1. Архитектура вычислительной системы
   1. Выбор вычислительной системы

Для сравнения производительности был проведён анализ ряда доступных процессоров. В результате анализа были выбраны модели AMD Ryzen 5 3500U и AMD Ryzen 7 7700. Данные модели показаны на рисунке 1.1.

a б

а – процессор AMD Ryzen 5 3500U;

б – процессор AMD Ryzen 7 7700

Рисунок 1.1 – Изображения выбранных процессоров

Основные характеристики процессоров представлены в таблице 1.1 [2].

Таблица 1.1 – Основные технические характеристики выбранных процессоров

|  |  |  |
| --- | --- | --- |
|  | AMD Ryzen 5 3500U | AMD Ryzen 7 7700 |
| Год выхода | 2019 | 2023 |
| Категория | Мобильный | Десктопный |
| Кодовое имя архитектуры | Picasso-U (Zen+) | Raphael (Zen 4) |
| Количество физических ядер | 4 | 8 |
| Количество логических ядер | 8 | 16 |
| Базовая частота, ГГц | 2,1 | 3,8 |
| Максимальная частота, ГГц | 3,7 | 5,3 |
| Кэш L1, Кб | 96 | 64 |
| Кэш L2, Кб | 512 | 1024 |
| Кэш L3, Мб | 4 | 32 |
| Технологический процесс, нм | 12 | 5 |
| TDP, Вт | 15 | 65 |
| Встроенная графика | Radeon Vega 8 | Radeon Graphics |
| Примечание – Данные о размере кэшей L1 и L2 указаны на одно ядро, данные о размере кэша L3 – на все ядра. | | |

Выбор данных моделей был обусловлен следующими факторами:

1. Наличие оборудования. Обладая обеими моделями процессоров, есть возможность удобно провести сравнительный анализ и оценить производительность каждого из устройств в выбранном классе задач.
2. Наличие встроенной графики. За счёт наличия встроенной графики в обеих моделях процессоров появляется возможность оценить и сравнить производительность не только вычислительных ядер, но и графических.
   1. История, версии и достоинства выбранной архитектуры

В качестве процессоров для оценки производительности в ходе данного курсового проекта были использованы процессоры Ryzen 5 3500U и Ryzen 7 7700 производства компании AMD. Оба процессора относятся к архитектуре x86-64, иначе известной как AMD-64. Данная архитектура, представленная в 2000 году компанией AMD, является версией архитектуры x86, хотя изначально разрабатывалась как расширение. Впервые данная архитектура была реализована в микропроцессоре AMD Opteron, выпущенном в апреле 2003 года.

В процессоре AMD Ryzen 5 3500U архитектура x86-64 реализована в микроархитектуре Zen+. Данная микроархитектура является улучшением микроархитектуры Zen. На рисунке 1.2 представлена упрощённая схема ядра процессора на основе микроархитектуры Zen.

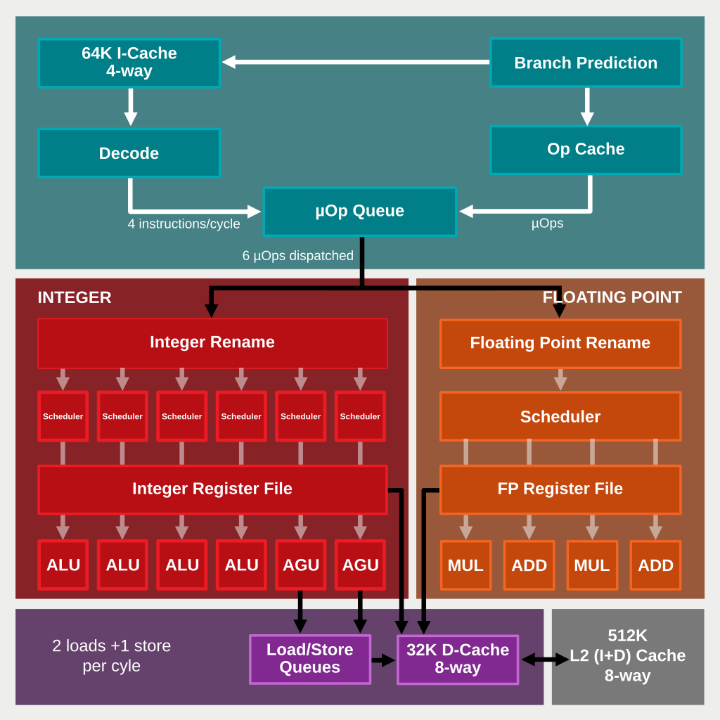


Рисунок 1.2 – Упрощённая иллюстрация ядра на основе микроархитектуры Zen.

В процессоре AMD Ryzen 7 7700 архитектура x86-64 реализована в микроархитектуре Zen 4. Для более наглядного сравнения перечисленных микроархитектур в таблице 1.2 представлены ключевые особенности и отличия микроархитектур от Zen до Zen 5. Стоит отметить, что для микроархитектуры Zen будут представлены отличия от семейства предыдущих микроархитектур компании AMD – Bulldozer.

Таблица 1.2 – Ключевые особенности и отличия микроархитектур семейства Zen

|  |  |
| --- | --- |
| Микроархитектура | Ключевые особенности и отличия |
| Zen | Два потока на ядро (SMT), добавлен кэш декодированных микроопераций [3], увеличен размер кэша L1, увеличена пропускная способность кэш-памяти, оптимизация задержек доступа к кэш-памяти [4], переход на 14-нм технологический процесс. |
| Zen+ | Улучшение регулировки тактовой частоты в зависимости от нагрузки [5], улучшения латентности кэша L2 и памяти [6], переход на 12-нм технологический процесс. |
| Zen 2 | Расширения набора инструкций: WBNOINVD, CLWB, RDPID, RDPRU, MCOMMIT (каждой из инструкций присвоен свой CPUID бит) [7], аппаратная защита от уязвимости Spectre V4 [8], переход на 7-нм технологический процесс. |
| Zen 3 | Изменения в расположении компонентов на чипе (рисунок 1.3), переход от двух четырёхядерных комплексов на чиплете к одному восьмиядерному, увеличение количества исполняемых за цикл инструкций на 19% [9]. |
| Zen 4 | Увеличение кэша микроопераций на 69%, удвоение размера кэша L2, увеличение максимальной тактовой частоты до 5,7 ГГц, расширены векторные (вещественные) регистры для работы с AVX-512, улучшение предсказания для прямых и косвенных ветвлений [10]. |
| Zen 5 | Добавление двух дополнительных АЛУ, улучшение блока предсказания ветвлений (предсказание двух ветвлений за один тактовый цикл), увеличение размера кэша L1, увеличение ассоциативности кэша L2 с 8 до 16. |

Также для справки приведена хронология выпуска микроархитектур, описанных выше:

– Zen – 2017;

– Zen+ – 2018;

– Zen 2 – 2019;

– Zen 3 – 2020;

– Zen 4 – 2022;

– Zen 5 – 2024.

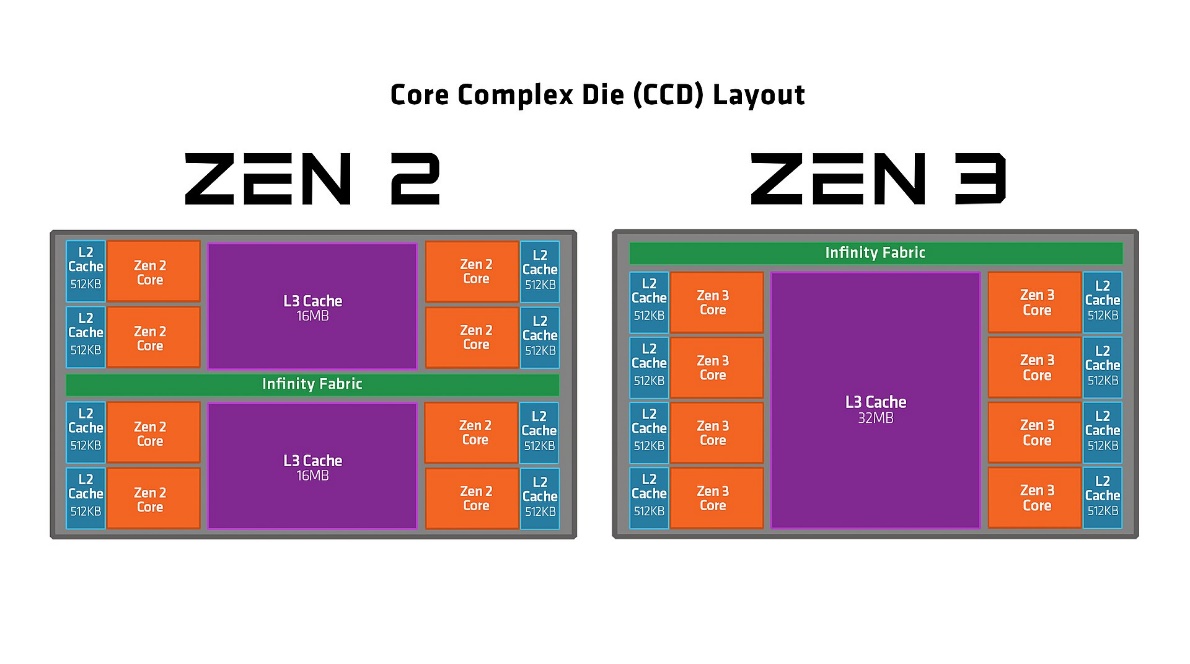


Рисунок 1.3 Сравнение расположения компонентов в микроархитектурах  
 Zen 2 и Zen 3

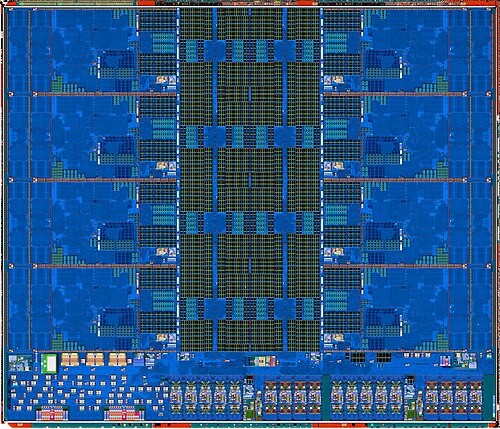


Рисунок 1.4 – Die-shot (снимок кристалла) процессора AMD Ryzen 5 9600X с микроархитектурой Zen 5.

Сама же архитектура x86 берёт своё начало в 1976 году, когда была начата разработка процессора Intel 8086, выпущенного в 1979 году. Архитектура набора команд, реализованная в этом процессоре, стала основой архитектуры x86. По этой причине все современные процессоры этой архитектуры могут выполнять все команды этого набора. Название архитектуры пошло от 2 последних цифр названий ранних моделей процессоров Intel – 8086, 80186, 80286 (i286), 80386 (i386), 80486 (i486).

Архитектура развивалась одновременно с развитием технологий разработки и производства микропроцессоров, в таблице 1.3 представлены избранные вехи в развитии архитектуры и процессоров, разработанных в соответствии с этой архитектурой.

Таблица 1.3 – Избранные вехи в развитии архитектуры x86

|  |  |
| --- | --- |
| Событие | Важность этапа для развития архитектуры |
| Выпуск Intel 8086 (1978) | Первый коммерческий процессор с реализованной в нём архитектурой x86. |
| Выпуск Intel 80286 (1982) | Появление такого понятия как защищённый режим и виртуальная память. |
| Выпуск Intel 80386 (19xx) | Первый 32-х разрядный процессор, появление режима виртуального 8086, аппаратной отладки и страничного преобразования. |
| Выпуск Intel Pentium (i586) (1993) | Первый суперскалярный и суперковейерный процессор. Номерные названия ушли по причине невозможности запатентовать число. |
| Выпуск Intel Pentium Pro (i686) (1995) | Появление блоков предсказания ветвлений, переименования регистров, RISC-ядра, интеграция L2 кэша в один корпус с ядром. |
| Выпуск Intel Pentium MMX (1997) | Появление поддержки технологии MMX. |
| Выпуск Intel LV-Xeon DP (2002) | Появление технологии Hyper-Threading. |
| Выпуск AMD Opteron (2003) | Первый процессор с архитектурой x86-64 |

К началу 2000-х годов стало очевидно, что 32-битное адресное пространство архитектуры x86 ограничивает производительность приложений, работающих с большими объёмами данных. 32-разрядное адресное пространство позволяет процессору осуществлять непосредственную адресацию лишь 4 ГБ данных. Этого может оказаться недостаточным для некоторых приложений, связанных, например, с обработкой видео или обслуживанием баз данных.

Для решения этой проблемы Intel разработала новую архитектуру IA-64 – основу семейства процессоров Itanium. Для обеспечения обратной совместимости со старыми приложениями, использующими 32-разрядный код, в IA-64 был предусмотрен режим эмуляции. Однако на практике данный режим работы оказался чрезвычайно медленным. Архитектура процессоров семейства Itanium представлена на рисунке 1.5.

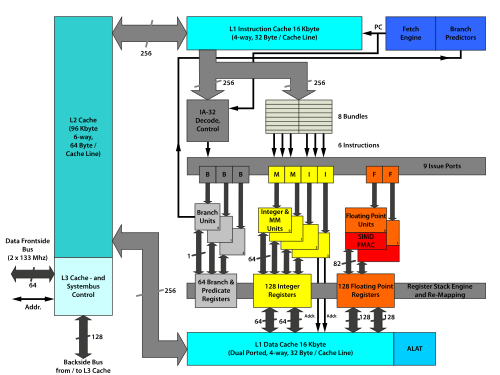


Рисунок 1.5 – Схема архитектуры процессоров семейства Itanium

Компания AMD предложила альтернативное решение проблемы увеличения разрядности процессора. Вместо того, чтобы изобретать совершенно новую систему команд, было предложено ввести 64-разрядное расширение к уже существующей 32-разрядной архитектуре x86. Первоначально новая архитектура называлась x86-64, позже она была переименована в AMD64. Первоначально новый набор инструкций поддерживался процессорами семейств Opteron, Athlon 64 и Turion 64 компании AMD. Успех процессоров, использующих технологию AMD64, наряду с вялым интересом к архитектуре IA-64, побудили Intel приобрести лицензию на набор инструкций AMD64. При этом был добавлен ряд специфических инструкций, не присутствовавших в изначальном наборе AMD64. Новая версия архитектуры получила название EM64T.

В литературе и названиях версий своих программных продуктов компании Microsoft и Sun используют объединённое именование AMD64/EM64T, когда речь заходит о 64-разрядных версиях их операционных систем Windows и Solaris соответственно. В то же время, поставщики программ для операционных систем семейства Linux, BSD используют метки «x86-64» или «amd64», а в Mac OS X используется метка «x86\_64», если необходимо подчеркнуть, что данное ПО использует 64-разрядные инструкции.

Процессоры данной архитектуры имеют два режима работы: Long mode и Legacy mode (режим совместимости с 32-битным процессорами на основе x86). На рисунке 1.6 представлена диаграмма, содержащая данные о режимах работы и процессорах, для которых эти режимы работы были введены и которые их поддерживают.

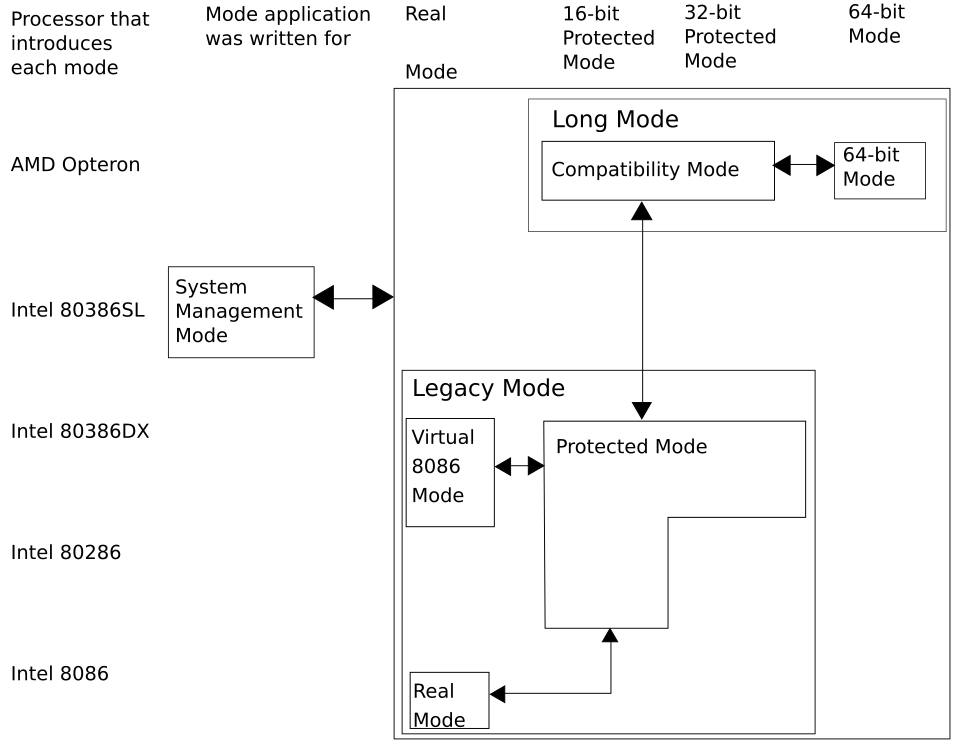


Рисунок 1.6 – Диаграмма режимов работы микропроцессоров

«Длинный» режим является основным для процессоров AMD64. Этот режим даёт возможность воспользоваться всеми преимуществами архитектуры x86-64. Для использования этого режима необходима любая 64-битная операционная система.

Этот режим позволяет выполнять 64-битные программы. Также, для обратной совместимости, предусмотрена поддержка выполнения 32-битного кода, включая 32-битные приложения. Однако при запуске 32-битных программ в 64-битной системе, они не смогут использовать 64-битные системные библиотеки, и наоборот. Для решения этой задачи большинство 64-разрядных операционных систем предоставляет два набора системных API: один для нативных 64-битных приложений и другой для 32-битных программ. Этот подход аналогичен методике, использовавшейся в ранних 32-битных системах, таких как Windows 95 и Windows NT, для выполнения 16-битных программ.

В режиме «Long Mode» были ликвидированы некоторые особенности архитектуры x86-32, такие как режим виртуального 8086 и сегментная модель памяти. Однако, осталась возможность использования сегментов FS и GS, что полезно для быстрого нахождения важных данных потока при переключении задач. Также аппаратная многозадачность и некоторые команды, связанные с устаревшими возможностями и работой с BCD-числами, которые редко используются в новых программах, были исключены. «Длинный» режим активируется установкой флага CR0.PG, который используется для включения страничного MMU. Таким образом, исполнение 64-битного кода с запрещённым страничным преобразованием невозможно. Это может вызвать определённые сложности в программировании, поскольку при переключении между «Long Mode» и «Legacy Mode» (например, для вызова функций BIOS или DOS, монитором виртуальной машины, и т. д.) требуется двойной сброс MMU, для чего код переключения должен располагаться в одинаково отображенной странице.

«Унаследованный» режим позволяет процессору x86-64 выполнять команды, предназначенные для процессоров x86, обеспечивая таким образом полную совместимость с 32-битным кодом и операционными системами для x86. В этом режиме процессор ведёт себя точно так же, как x86-процессор (например, как Athlon или Pentium III). Функции и возможности, предоставляемые архитектурой x86-64 (например, 64-битные регистры), в этом режиме, естественно, недоступны. В этом режиме 64-битные программы и операционные системы работать не будут.

Разрабатывая архитектуру AMD64 (AMD-реализации x86-64), инженеры корпорации AMD решили навсегда покончить с главным «рудиментом» архитектуры x86 – сегментной моделью памяти, которая поддерживалась ещё со времён 8086. Однако из-за этого при разработке первой x86-64-версии своего продукта для виртуализации программисты компании VMware столкнулись с непреодолимыми трудностями при реализации виртуальной машины для 64-битных гостевых систем [11]: поскольку для отделения кода монитора от кода «гостя» программой использовался механизм сегментации, эта задача стала практически неразрешимой.

1. Платформа программного обеспечения
   1. Структура и архитектура платформы

Для выполнения данного курсового проекта была выбрана операционная система Linux (GNU/Linux). Ниже рассмотрены основные компоненты данной платформы, её аспекты и характеристики, а также роль платформы в разработке и выполнении прикладных программ.

Платформа Linux имеет модульную структуру, что позволяет ей быть конкурентноспособной во множестве различных сценариев применения и обеспечивать высокую надёжность и производительностъ. ОС Linux состоит из следующих компонентов:

1. Ядро Linux (Linux Kernel) – центральная часть операционной системы, обеспечивающая приложениям координированный доступ к ресурсам компьютера, таким как процессорное время, память, внешнее аппаратное обеспечение, внешнее устройство ввода и устройства вывода. Также обычно ядро предоставляет сервисы файловой системы и сетевых протоколов. Как основополагающий элемент ядро представляет собой наиболее низкий уровень абстракции для доступа приложений к ресурсам системы, необходимым для своей работы. Как правило, ядро предоставляет такой доступ исполняемым процессам соответствующих приложений за счёт использования механизмов межпроцессного взаимодействия и обращения приложений к системным вызовам ОС.
2. Стандартная библиотека языка Си (libc) – библиотека Си, которая обеспечивает системные вызовы и основные функции, такие как open, malloc и т. д. Стандартная библиотека Cи используется для всех динамически скомпонованных программ. По сути стандартная библиотека является обёрткой над системными вызовами ядра Linux выполненной на языке Си, как показано на рисунке 2.1. Стоит также отметить, что libc является лишь частью стандарта ANSI C, то есть описанием программного интерфейса, а не настоящей библиотекой, используемой при компиляции. Существуют различные реализации стандартной библиотеки. В большинстве дистрибутивов Linux glibc (GNU C Library) выступает в качестве стандартной библиотеки. Данная реализация стремится предоставить пользователям наиболее оптимизированный относительно скорости выполнения вариант. Предназначена преимущественно для динамической компановки, поэтому оптимизация размера библиотеки на диске и в оперативной памяти не была наиболее приоритетной задачей при её разработке. Наиболее популярный конкуррент glibc – musl. musl – реализация стандартной библиотеки языка Си, нацеленная на эффективность в том числе и при статической компановке. Данная реализация стандартной библиотеки была разработана с нуля чтобы обеспечить надежность в реальном времени, избегая состояний гонки, внутренних сбоев при исчерпании ресурсов и различных других плохих случаев поведения, присутствующих в существующих реализациях [12].
3. Системные компоненты – программы, обеспечивающие работу системы или определённых подсистем. К системным компонентам относят: подсистемы инициализации (init daemon), системные демоны, оконные системы, библиотеки для работы с графикой и т.д. Примером подсистемы инициализации может послужить используемая в большинстве дистрибутивов systemd или OpenRC, используемая в Alpine Linux и других дистрибутивах.

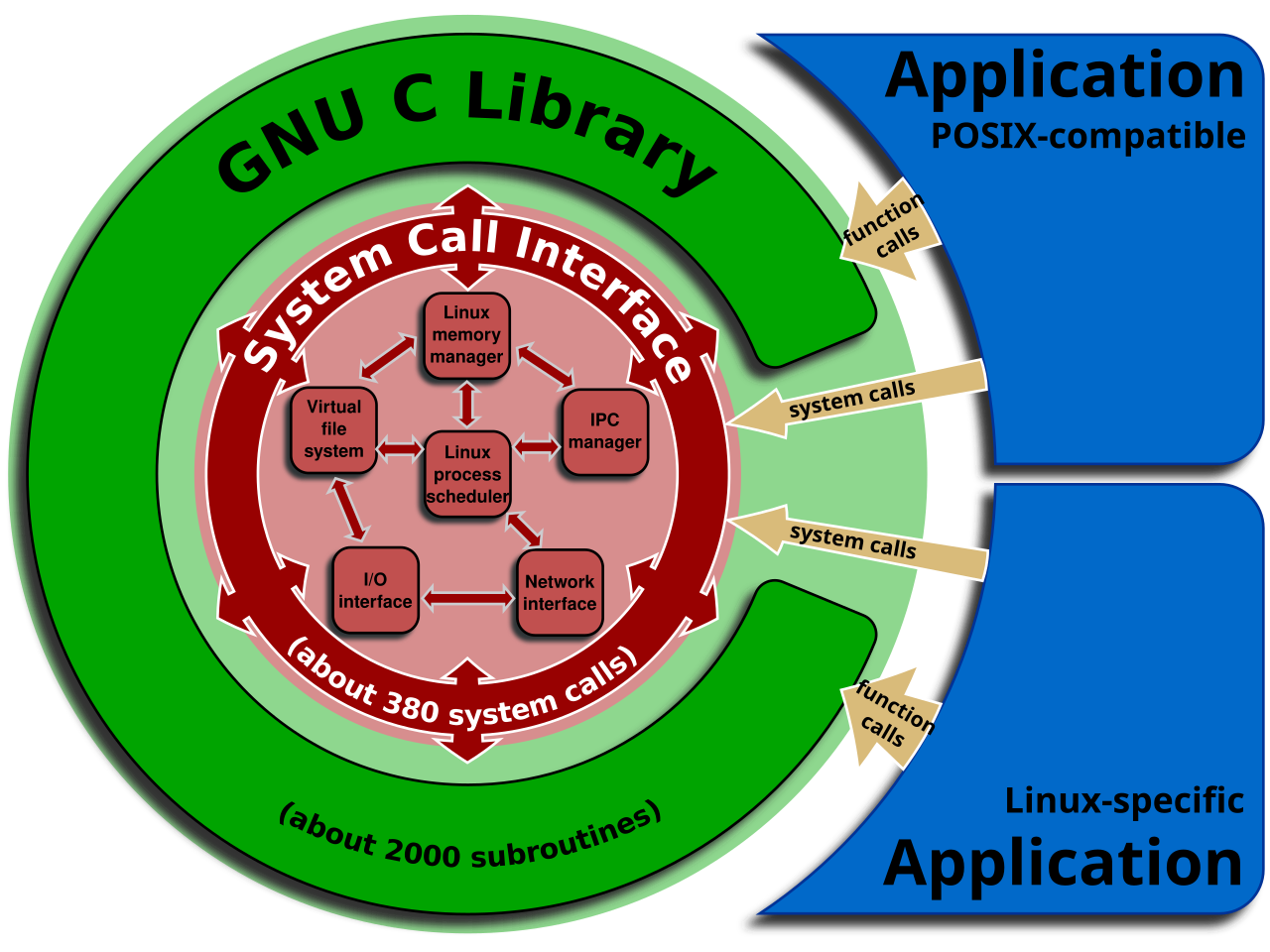


Рисунок 2.1 – Схематичное изображение отношения между стандартной библиотекой, приложениями и ядром Linux.

Ядро Linux относится к классу монолитных ядер операционных систем. Основными особенностями этого архитектурного класса являются работа всех частей ядра в одном адресном пространстве и богатый набор абстракций оборудования. Альтернативой данного класса ядер являются микроядра, которые реализуют минимальную функциональность (управление физической и виртуальной памятью компьютера, управление процессорным временем, управление доступом к устройствам ввода-вывода,коммуникация и синхронизация процессов) в пространстве ядра, а все другие функции (драйверы устройств, реализации файловых систем и др.) являются процессами в пользовательском пространстве и осуществляют взаимодействие с ядром с помощью системных вызовов, а взаимодействие между собой с помощью средств межпроцессного взаимодействия. Также существуют гибридные архитектуры ядер, к которым например относится ядро Windows NT. Микроядро NT слишком велико (более 1 Мбайт, кроме того, в ядре системы находится, например, ещё и модуль графического интерфейса), чтобы носить приставку «микро». Компоненты ядра Windows NT располагаются в вытесняемой памяти и взаимодействуют друг с другом путём передачи сообщений, как и положено в микроядерных операционных системах. В то же время все компоненты ядра работают в одном адресном пространстве и активно используют общие структуры данных, что свойственно операционным системам с монолитным ядром.

Старые монолитные ядра требовали перекомпиляции при любом изменении состава оборудования. Большинство современных ядер, такие как Linux, позволяет во время работы динамически подгружать и выгружать модули, выполняющие часть функций ядра. Модульность ядра осуществляется на уровне бинарного образа, а не на архитектурном уровне ядра, так как динамически подгружаемые модули загружаются в адресное пространство ядра и в дальнейшем работают как непосредственная часть ядра. Модульные монолитные ядра не следует путать с архитектурным уровнем модульности, присущим микроядрам и гибридным ядрам. Практически, динамическая загрузка модулей — это просто более гибкий способ изменения образа ядра во время выполнения — в отличие от перезагрузки с другим ядром. Модули позволяют легко расширить возможности ядра по мере необходимости. Динамическая подгрузка модулей помогает сократить размер кода, работающего в пространстве ядра, до минимума, например, свести к минимуму размер ядра для встраиваемых устройств с ограниченными аппаратными ресурсами.

* 1. История, версии и достоинства

Семейство операционных систем GNU/Linux является одной из наиболее значимых ступеней в развитии операционных систем и в компьютерной индустрии в целом. История Linux началась в 1991 году и уже к 2000-м годам системы Linux являются основными для серверов и суперкомпьютеров, распространение среди пользователей настольных компьютеров также растёт, хотя и значительно меньшими темпами, и по статистике на сентябрь 2025 года достигло 3,17% [13].

Предшественником и идейным вдохновителем GNU/Linux послужила операционная система Unix, являющаяся предком или вдохновителем большинства используемых операционных систем, что показано на рисунке 2.2.

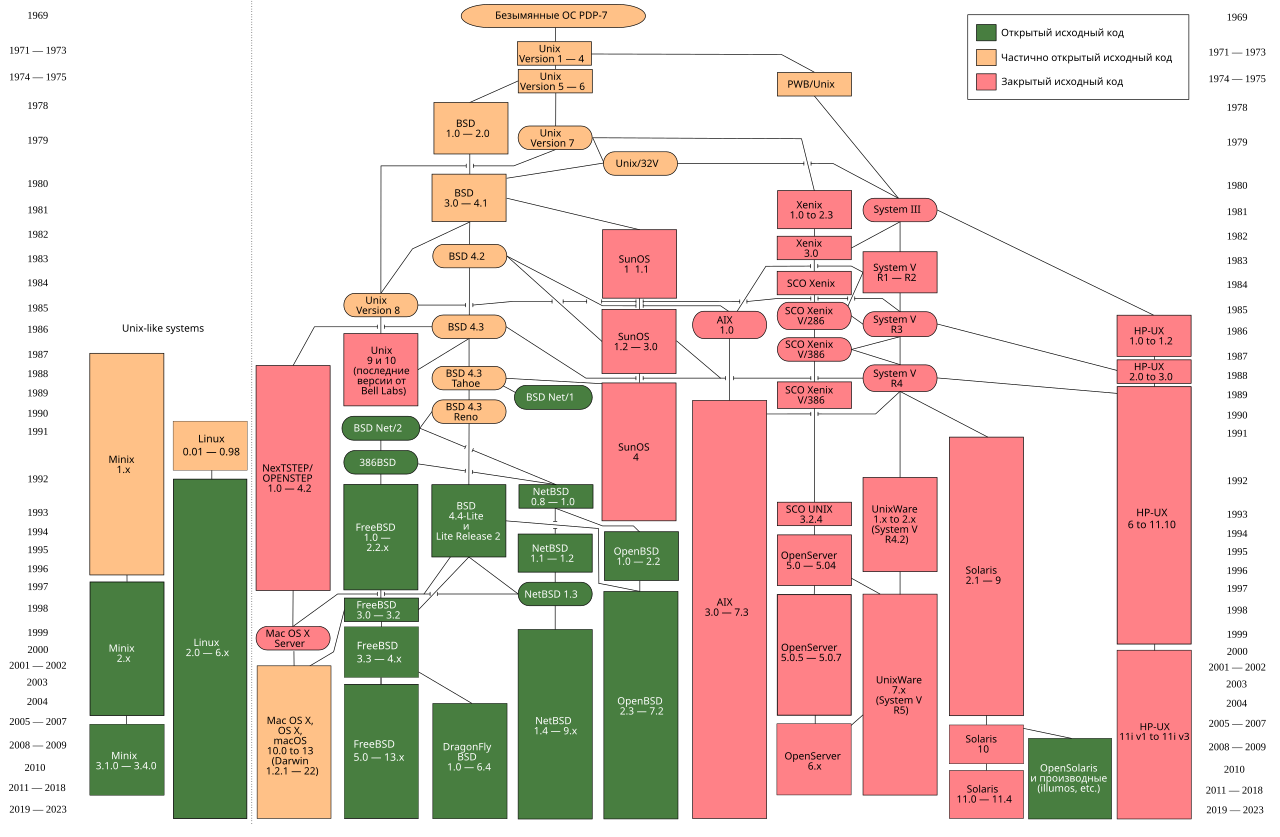


Рисунок 2.2 – Гениалогическое древо Unix-систем

В 1984 году разработавшая Unix компания Bell Labs, принадлежащая AT&T, начала продавать ОС как проприетарный продукт, модификация которого была юридически запрещена. В связи с закрытостью Unix в 1983 Ричард Столлман основал проект GNU, основной целью которого было создание свободной Unix-подобной операционной системы с открытым исходным кодом. Аббревиатура GNU это рекурсивный акроним от английского GNU’s Not Unix – «GNU – не Unix». К началу 1990-х годов был готов ряд программ, необходимых для операционной системы, таких как библиотеки, компиляторы, текстовые редакторы и оболочка коммандной строки, однако разработка низкоуровневых элементов таких как драйверы, демоны и ядро, вместе называемых GNU Hurd, была затянута и данные элементы были далеки от готовности.

В 1990 году студент Хельсинкского университета Линус Торвальдс записался на курс по Unix-системам. На данном курсе в качестве одного из обязательных материалов для изучения фигурировала книга «Операционные системы: разработка и реализация» за авторством Эндрю Таненбаума. В комплекте с книгой распространялся также исходный код операционной системы Minix, разработанной самим Таненбаумом в целях обучения студентов конструированию компьютеров и созданию операционных систем. Данная операционная система была лицензирована только для учебного использования.

Из-за проблем с лицензированием *Minix* в 1991 году Торвальдс начал работу над своим ядром операционной системы, которое впоследствии стало ядром Linux.

В феврале 1992 Линус Торвальдс перевёл свой проект на лицензию *GPLv2* (*GNU General Public License version 2*) [14], которая, в отличие от его собственной лицензии, позволяла свободно коммерчески распространять ядро. Это позволило с одной стороны использовать *Linux* в качестве ядра для операционной системы GNU, а с другой стороны заменить *Unix* и *Minix* приложения на свободные приложения GNU. По этой причине технически семейство операционных систем называемое Linux правильнее называть *GNU/Linux*, потому как оба компонента обеспечивают существование операционной системы как общего целого.

В мае 1992 была выпущена версия ядра 0.96, позволявшая запускать оконную систему *X11*.

Версия 1.0, первая версия *Linux*, подходящая для работы в окружении конечного пользователя, была выпущена в марте 1994.

В июне 1996 была выпущена версия 2.0, основными нововведениями которой были симметричная многопроцессорная обработка (*SMP*), поддержка большего количества типов процессоров и поддержка выбора конкретного аппаратного обеспечения для обеспечения специфических особенностей архитектуры и оптимизации, а также загружаемые модули ядра.

В версии 2.2, выпущенной в январе 1999 года была добавлена поддержка 64-разрядных архитектур, а также множества файловых систем, включая поддержку *NTFS* компании *Microsoft* в режиме только для чтения.

В версии 2.4.0 была добавлена поддержка микроархитектур *Pentium* 4 и *Itanium* а также поддержка *Bluetooth, RAID* и *LVM.*

В версии 2.6.0 в ядро была интегрирована реализация стандартной библиотеки языка Си *uClinux*, предназначенная для использования во встраиваемых системах, максимальное количество пользователей было увеличено с 216 до 232, а максимальное количество процессов – до 229, но только для 64-разрядных систем и ещё множество других нововведений.

Изменение нумерации с 2.6.39 на 3.0 и с 3.19 на 4.0 не имело существенных технических отличий; номер основной версии был просто увеличен, чтобы избежать большого числа во второстепенном номере [15].

В апреле 2015 было выпущено ядро версии 4.0, а в марте 2019 – ядро версии 5.0.

Актуальная на момент написания проекта стабильная версия ядра *Linux* – 6.17.5, именно она и используется в качестве ядра программной платформы.

По поводу наименования данного семейства операционных систем по сей день ведутся споры. Сторонники *FSF* (*Free Software Foundation*), авторы программ *GNU*, и Ричард Столлман в частности считают, что операционная система должна быть названа GNU/Linux в соответствии с тем, кто разработал какие части системы. Столлман писал: «Проект *GNU* не был и не является проектом по разработке определенных пакетов программ. […] Многие сделали серьезный вклад в свободные программы этой системы, и все они заслуживают признательности за свои программы. Но причина, по которой это является целостной системой – а не только набором полезных программ – то, что проект *GNU* постановил сделать это такой системой. Мы составили список программ, необходимых, чтобы свободная система была полной, и систематически отыскивали, писали или отыскивали людей для написания всего, что стояло в списке».

Сам Торвальдс относится к данным спорам иронично. В документальном фильме *Revolution OS* он говорит: «Ну, я думаю, что это справедливо, однако справедливо в том случае, если вы сделаете *GNU* дистрибутив *Linux* … так же как я считаю, что [название] “*Red Hat Linux*” это нормально, или “*SUSE Linux*”, или “*Debian Linux*”, потому что если вы реально сделаете свой дистрибутив *Linux* вы можете дать ему название, но называть *Linux* в общем “*GNU/Linux*” я считаю попросту смешным» [ССЫЛКА НА ИНТЕРВЬЮ НА ЮТУБЕ]. Однако общее его вовлечение в этот спор невелико.

Весь долгий путь пройденный *Linux* свидетельствует о постоянной эволюции и стремлении сообщества к улучшению производительности, безопасности и удобства использования для миллионов пользователей по всему миру. *Linux* остается одной из наиболее популярных и важных операционных систем на сегодняшний день.

* 1. Обоснование выбора платформы

В ходе курсового проекта выбор программной платформы играет ключевую роль. В качестве кандидатов на роль программной платформы при выполнении данного курсового проекта были выделены *Arch Linux, Windows 10* и *Windows 11.* В данном случае, было принято решение использовать дистрибутив из семейства операционных систем *Linux Arch Linux* в качестве платформы для проведения сравнения выбранных моделей процессоров. Существует несколько важных аргументов, которые обосновывают выбор *Arch Linux:*

1. Легковесность. Дистрибутив *Arch Linux* является одним из наиболее оптимизированных дистрибутивов в мире. Это означает, что множество инструментов и программ не установлены по умолчанию. Такой подход позволяет наиболее точно оценить производительность процессора в отсутствие задач заднего плана и даже графической оболочки, в чём и заключается основное преимущество над основным конкурентом в данном выборе – операционной системой *Windows 10*.
2. Поддержка многих процессоров. *Linux* обладает высокой совместимостью с различными аппаратными конфигурациями, включая многоядерные процессоры. Это предоставляет отличную возможность для исследования и оптимизации вычислений на различных системах.
3. Современность. *Arch Linux* выпускается в соответствии с моделью *Rolling Release*, то есть позволяет пользователю всегда иметь последние версии устанавливаемых программ, избавляя его от необходимости периодической переустановки системы. Данная модель также относится и к версии ядра *Linux*, поэтому при использовании данного дистрибутива у пользователя всегда есть доступ к самым новым нововведениям и особенностям ПО.
4. Доступность библиотек и фреймворков. Платформа *Linux* на данный момент де-факто является стандартом индустрии, поэтому большинство библиотек и фреймворков доступно и имеет наилучшую поддержку именно для *Linux*.

Таким образом, выбор операционной системы *Arch Linux* в качестве платформы для проведения сравнения производительности процессоров обоснован легковесностью, поддержкой множества процессоров и архитектур, наличием разнообразных инструментов разработки, доступом к наиболее современным версиям программ, богатстве библиотек и удобстве использования. Эта платформа позволяет провести обширное исследование в области оптимизации производительности вычислений.

Заключение

Текст заключения.

Список литературных источников

1. Доманов, А. Т. Стандарт предприятия / А. Т. Доманов,   
   Н. И. Сорока. – Минск: БГУИР, 2024. – 178 с.
2. Technical city [Электронный ресурс]. – Режим доступа: https://technical.city/ru/cpu/Ryzen-5-3500U-protiv-Ryzen-7-7700. – Дата доступа: 20.09.2025.
3. AnandTech [Электронный ресурс]. – Режим доступа: https://www.anandtech.com/print/10578/amd-zen-microarchitecture-dual-schedulers-micro-op-cache-memory-hierarchy-revealed. – Дата доступа: 17.12.2019.
4. AnandTech [Электронный ресурс]. – Режим доступа: https://www.anandtech.com/show/10578/amd-zen-microarchitecture-dual-schedulers-micro-op-cache-memory-hierarchy-revealed/2. – Дата доступа: 17.12.2019.
5. Forbes [Электронный ресурс]. – Режим доступа: https://www.forbes.com/sites/antonyleather/2018/01/07/amd-confirms-new-zen-ryzen-cpus-for-april-2018-x470-chipset-threadripper-and-apus-inbound-too/. – Дата доступа: 20.09.2025.
6. PC World [Электронный ресурс]. – Режим доступа: https://www.pcworld.com/article/3246211/computers/amd-reveals-ryzen-2-threadripper-2-7nm-navi-and-more-in-ces-blockbuster.html. – Дата доступа: 13.01.2018.
7. Phoronix [Электронный ресурс]. – Режим доступа: https://www.phoronix.com/news/AMD-Zen-2-New-Instructions. – Дата доступа: 20.09.2025.
8. TechPowerUp [Электронный ресурс]. – Режим доступа: https://www.techpowerup.com/256478/amd-zen-2-has-hardware-mitigation-for-spectre-v4. – Дата доступа: 20.09.2025.
9. AnandTech [Электронный ресурс]. – Режим доступа: https://www.anandtech.com/show/16214/amd-zen-3-ryzen-deep-dive-review-5950x-5900x-5800x-and-5700x-tested. – Дата доступа: 12.01.2020.
10. AnandTech [Электронный ресурс]. – Режим доступа: https://web.archive.org/web/20220926130934/https://www.anandtech.com/show/17585/amd-zen-4-ryzen-9-7950x-and-ryzen-5-7600x-review-retaking-the-high-end. – Дата доступа: 26.09.2022.
11. Pagetable [Электронный ресурс]. – Режим доступа: http://www.pagetable.com/?p=25. – Дата доступа: 18.07.2011.
12. musl – Introduction [Электронный ресурс]. – Режим доступа: https://www.musl-libc.org/intro.html. – Дата доступа: 26.10.2025.
13. StatCounter – Desktop Operating Systems Market Share Worldwide [Электронный ресурс]. – Режим доступа: https://gs.statcounter.com/os-market-share/desktop. – Дата доступа: 26.10.2025.
14. kernel.org [Электронный ресурс]. – Режим доступа: https://www.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.12. – Дата доступа: 27.10.2025.
15. Linux-Kernel Archive [Электронный ресурс]. – Режим доступа: https://lkml.indiana.edu/hypermail/linux/kernel/1107.2/01843.html. – Дата доступа: 27.10.2025.