Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа Группа 24M.71-мм

Разработка образовательного комплекса виртуального устройства и минимальной ОС на архитектуре RISC-V

***Кисельков Денис Андреевич***

Отчёт по учебной практике в

форме «Производственное задание»

Научный руководитель:

доцент кафедры системного программирования, к. ф.-м. н., Луцив Д. В.

Санкт-Петербург 2025

Оглавление

[Введение 3](#_Toc210598414)

[1. Постановка задачи 3](#_Toc210598415)

[2. Обзор 4](#_Toc210598416)

[2.1. Операционные системы 4](#_Toc210598417)

[2.2. Учебные компьютеры 8](#_Toc210598418)

[3. Описание решения 12](#_Toc210598419)

[3.1 Технологии и инструменты разработки 12](#_Toc210598420)

[3.2 Моделирование и эволюция систем 13](#_Toc210598421)

[3.3 Структура и архитектура виртуальных вычислительных устройств 13](#_Toc210598422)

[3.4 Реализация и интеграция программного обеспечения 14](#_Toc210598423)

[4. Тестирование 15](#_Toc210598424)

[4.1 Аппаратно-программная конфигурация 15](#_Toc210598425)

[4.2 Методика тестирования 15](#_Toc210598426)

[4.3 Результаты тестирования 15](#_Toc210598427)

[5. Заключение 16](#_Toc210598428)

Введение

В современных российских IT-образовательных программах по архитектуре вычислительных систем и операционных систем наблюдается дефицит практических материалов, которые позволяют студентам перейти от теории к глубокому пониманию полного цикла работы вычислительных устройств и минимальных операционных систем. Несмотря на значительный рост IT-отрасли, включая увеличение доли программного обеспечения до 44,3% в 2024 году на фоне сокращения аппаратной части, существует нехватка системных программистов, обусловленная сложностью освоения архитектур процессоров и операционных систем.

Целью данной работы является создание учебного комплекса, который объединит реализацию минимальных процессоров и операционных систем для виртуальных устройств, обеспечивая наглядное и последовательное изучение устройства компьютера с аппаратной и программной сторон. В основе исследования лежит использование таких материалов, как Little Man Computer, ранние версии Linux и GNU/Mes [4], а также разработка виртуальных устройств на Verilog. Постепенное расширение функциональности (включая графический интерфейс, поддержку USB и сетевых адаптеров) позволит сформировать полноценный образовательный инструмент для системного изучения компьютерных наук. Анализ будет опираться на существующие проекты и курсы, например, xv6, nandgame и nand2tetris, обеспечивающие системный подход к обучению от логических элементов до программируемых систем.

1. Постановка задачи

Целью проекта является создание и документирование учебного комплекса, включающего виртуальные процессоры и минимальные операционные системы для них, обеспечивающих полноценное изучение принципов работы вычислительных систем и ОС. Основные задачи включают:

1. Разработку простых виртуальных процессоров (одноцикловых и многоцикловых, включая RISC-V и nand\_cpu на Verilog) с возможностью моделирования и тестирования.
2. Сборку и запуск минимальных операционных систем (минимальный Linux, Embox, упрощённый xv6) на созданных процессорах.
3. Подготовку подробной документации и учебных материалов для преподавателей и студентов.
4. Организацию тестирования, анализа результатов и сравнительный обзор существующих средств обучения архитектуре вычислительных систем и системному программированию.

2. Обзор

2.1. Операционные системы

Операционная система (ОС) — это комплекс программного обеспечения, который управляет аппаратными ресурсами компьютера и предоставляет интерфейс для взаимодействия между пользователем и устройством [12]. Она выполняет множество ключевых функций, обеспечивая эффективное выполнение прикладных программ и управление системными ресурсами.

Основные функции операционной системы [13]:

1. Компиляция и исполнение программ:

ОС отвечает за загрузку приложений в оперативную память и их исполнение, обеспечивая необходимую среду для выполнения программ. Это включает в себя управление процессами, которые представляют собой запущенные экземпляры программ.

2. Управление процессором:

ОС распределяет вычислительные задачи между процессорными ядрами, контролируя их загрузку и обеспечивая эффективное выполнение задач. Это может включать как однозадачность, так и многозадачность, где несколько процессов могут выполняться одновременно.

3. Управление памятью:

Операционная система управляет оперативной памятью, выделяя её для различных процессов и предотвращая конфликты при доступе к памяти. Она также реализует механизмы виртуальной памяти, которые позволяют использовать дисковое пространство как расширение оперативной памяти.

4. Управление файлами:

ОС обеспечивает организацию и управление файловыми системами, позволяя пользователям создавать, удалять и изменять файлы.

Она также управляет доступом к данным на различных носителях (жёсткие диски, SSD и т.д.).

5. Управление устройствами ввода-вывода:

ОС взаимодействует с периферийными устройствами через драйверы, обеспечивая корректный ввод и вывод данных. Это включает в себя работу с клавиатурами, мышами, принтерами и сетевыми устройствами.

6. Интерфейс пользователя:

Операционная система предоставляет графический или текстовый интерфейс для удобного взаимодействия пользователя с компьютером. Это позволяет пользователям запускать приложения, управлять файлами и выполнять другие задачи без необходимости глубоких технических знаний.

**2.1.1. Первая версия ядра Linux**

Первая версия ядра Linux, выпущенная 17 сентября 1991 года Линусом Торвальдсом и обозначенная как 0.01, стала важной вехой в истории операционных систем. Однако это была не полноценная ОС, а лишь прототип ядра, который обеспечивал базовые функции. Для создания полноценной операционной системы требовалось интегрировать это ядро с другими компонентами, что произошло позже с появлением первых дистрибутивов Linux.

Документация для первой версии была ограниченной по нескольким причинам. Линус Торвальдс разрабатывал ядро в свободное время и не уделял достаточного внимания документированию кода. Сообщество разработчиков только начинало формироваться, и многие пользователи были более заинтересованы в экспериментировании с кодом, чем в его документировании.

Ядро 0.01 включало в себя функции: управление памятью, многозадачность и элементарная поддержка файловой системы. Также была реализована возможность взаимодействия с пользователем через терминал. Эта первая версия стала основой для дальнейшего развития проекта, который со временем превратился в одно из самых популярных и мощных ядер операционных систем в мире.

**2.1.2. GNU/Linux**

Объединение ядра Linux и системы GNU, например с использованием BusyBox, позволяет создать полную систему GNU/Linux [14]. GNU предоставляет набор утилит и библиотек, среди которых компилятор GCC и оболочка Bash. Ядро Linux и BusyBox, хотя и являются основными компонентами многих дистрибутивов Linux, не подходят для изучения операционных систем в качестве учебных материалов. Основная причина заключается в том, что код этих компонентов очень объемный и сложный. Ядро Linux содержит миллионы строк кода, что делает его трудным для понимания и анализа, особенно для начинающих. Это может привести к тому, что студенты будут перегружены информацией и не смогут сосредоточиться на ключевых концепциях операционных систем, таких как управление памятью, процессы и файловые системы.

BusyBox представляет собой компактный набор утилит [1]; однако он также имеет свои сложности. Он объединяет множество утилит в одном исполняемом файле, что делает его сложным для изучения в контексте отдельных команд и их функциональности. Для понимания работы операционных систем важно изучать отдельные компоненты и их взаимодействие, а не сводить все к одному универсальному инструменту. Это может затруднить понимание того, как различные части системы работают вместе.

**2.1.3. GNU/Mes**

GNU Mes — это инструмент для самодостаточной сборки компиляторов, который обеспечивает процесс бутстрэппинга (bootstrap) для компилятора GCC. Основная цель GNU Mes заключается в создании верифицированной начальной сборки компилятора, что позволяет избежать зависимости от уже собранных бинарных файлов, которые могут содержать скрытые закладки и другие уязвимости.

GNU Mes в первую очередь предназначен для бутстрэппинга компиляторов; это делает его более специализированным инструментом по сравнению с обычной операционной системой. Он предоставляет самодостаточный интерпретатор языка Scheme и простой компилятор для языка C, но не включает множество компонентов полноценной операционной системы, таких как управление процессами или сетевые функции.

Для работы с GNU Mes требуется знание языка Scheme и C, а также понимание процесса компиляции и бутстрэппинга. Несмотря на то, что цель GNU Mes — создать компактную и документированную систему, процесс бутстрэппинга сильно осложняет её использование. Также проект все еще находится в разработке и пока не полностью задокументирован.

**2.1.4. Embox**

Embox — это операционная система реального времени, специально разработанная для использования во встроенных системах с ограниченными ресурсами [3]. Она является кросс-платформенной и поддерживает множество архитектур, таких как x86, ARM, MIPS и другие. Основная цель Embox заключается в создании среды, которая позволяет запускать приложения, разработанные для Linux, в более предсказуемом и безопасном окружении. Это особенно важно для задач реального времени.

Embox разработан для встроенных систем, где аппаратные ресурсы часто ограничены; это приводит к необходимости глубокой настройки системы под конкретные задачи. Задачи, решаемые с помощью Embox, как правило, очень специфичны и не отражают более широкие концепции, присущие традиционным ОС. Например, встроенные системы часто требуют управления аппаратными ресурсами на низком уровне.

В дальнейшем можно попробовать собрать минимальную сборку подходящую для обучения.

**2.1.5. CP/M**

CP/M (Control Program/Monitor) — это 8-битная операционная система, разработанная в 1973 году Гэри Килдаллом на языке программирования PL/M [2]. Она была создана для работы на микропроцессорах Intel 8080 и Zilog Z80 и быстро стала стандартом для многих микрокомпьютеров того времени.

Хотя CP/M была одной из первых операционных систем, она не подходит для изучения современных операционных систем по нескольким причинам. CP/M была разработана в 1970-х годах для работы на ограниченном аппаратном обеспечении, что делает её архитектуру устаревшей. Также CP/M страдает от недостатка документации и поддержки. С течением времени оригинальные материалы по CP/M стали труднодоступными.

**2.1.6. MicroDOS**

MicroDOS — одна из первых операционных систем для советских персональных компьютеров БК-0010 [5]. Данная ОС больше не поддерживается с 1992 года, что ставит под сомнение её актуальность для образовательных целей. Также MicroDOS является закрытым программным обеспечением; это ограничивает доступ к исходному коду и возможность его модификации, препятствуя глубокому анализу системы.

**2.1.7. Unix System**

Unix System V6 (Шестая версия Unix) была выпущена Bell Labs в 1975 году. Она стала одной из первых версий операционной системы Unix, получивших широкое распространение. Эта версия была разработана для архитектуры PDP-11 и использовала ранний диалект языка C [9]. Unix V6 отличалась простотой архитектуры и небольшим объемом кода, что делало её подходящей для образовательных целей и изучения основ проектирования операционных систем.

Однако системы на основе Unix V6 не обладают современными функциями. Отсутствует поддержка многопоточности и улучшенные механизмы управления памятью. Также старая архитектура усложняет обучение.

**2.1.8. Xv6**

Xv6 — учебная операционная система, основанная на шестой версии

UNIX, разработанная в Массачусетском технологическом институте (MIT). Она предназначена для обучения основам проектирования операционных систем [6]. Одним из главных достоинств Xv6 является простота: она написана на языке ANSI C и содержит около 9000 строк кода, что делает её легкой для понимания и анализа. Это позволяет быстро ознакомиться с основными концепциями операционных систем (управление процессами, памятью). Весь код структурирован; содержит множество комментариев о работе фрагментов кода.

Xv6 адаптирована для современных архитектур x86 и RISC-V; это помогает лучше понять принципы работы современных вычислительных систем. Несмотря на свои преимущества, Xv6 имеет некоторые ограничения связанные с отсутствием поддержки реального оборудования. Операционная система обычно запускается в эмуляторах (например QEMU), что может снизить уровень взаимодействия студентов с физическими системами.

2.2. Учебные компьютеры

Учебные компьютеры представляют собой специализированные устройства, предназначенные для образовательных целей. Эти компьютеры могут использоваться в различных учебных заведениях для поддержки учебного процесса.

**2.2.1. LMC**

Компьютер маленького человечка (Little Man Computer, LMC) — это модель компьютера, предназначенная для обучения тому, как устроен и работает компьютер. Эта модель была предложена профессором Стюартом Мэдником в 1965 году и успешно используется для обучения студентов начальных курсов как в области программирования, так и конструирования компьютеров [10].

LMC представляет собой аналогию почтового отделения, где ”маленький человек” выполняет инструкции, находясь в закрытой комнате. В этой модели:

* память: состоит из 100 почтовых ящиков (адреса от 0 до 99), каждый из которых может хранить трехзначные числа (от 000 до
* 999);
* вход и выход: данные поступают в INBOX и выводятся из OUTBOX;
* аккумулятор: простое устройство для выполнения арифметических операций (сложение и вычитание);
* программный счетчик: указывает адрес следующей инструкции, которую необходимо выполнить.

Работа LMC основана на простом наборе команд, который позволяет выполнять базовые операции. Каждая команда представлена трехзначным числом, где первая цифра определяет тип команды.

**2.2.2. SIC/XE**

SIC/XE (Simplified Instructional Computer Extra Equipment) — усовершенствованная версия архитектуры SIC (Simplified Instructional Computer). Эта архитектура используется для изучения основ компьютерной науки и программирования [8].

Одним из ключевых отличий от SIC является значительно увеличенный объем памяти, который составляет 1 мегабайт, что позволяет хранить больше информации и выполнять более сложные задачи. Память организована в 8-битные байты, а слово состоит из 3 последовательных байтов (24 бита).

Кроме того, SIC/XE имеет расширенное количество регистров — всего 9, включая базовый регистр и регистры с плавающей запятой.

Архитектура также поддерживает четыре формата инструкций, различающихся по длине и способу адресации.

Важным аспектом SIC/XE является поддержка различных режимов адресации, включая прямую, индексированную и базовую адресацию. Набор инструкций также был расширен: добавлены операции для работы с числами с плавающей запятой и системные вызовы для обработки прерываний.

**2.2.3. Е14**

Модель ”Е14” разработана для изучения многоядерной архитектуры и параллельного программирования. Она представляет собой симулятор, который позволяет осваивать сложные концепции в удобной форме. Вот ключевые особенности и возможности этой модели:

* в ”Е14” реализована несимметричная архитектура, где один из процессоров выполняет функции главного;
* система команд ”Е14” полностью совместима с предшествующей моделью ”Е97”, что обеспечивает плавный переход от изучения однопроцессорной к многопроцессорной архитектуре;
* для межпроцессорного обмена используются те же методы, что и в классической архитектуре: обращение к внешним устройствам через порты, обмен данными по шине и алгоритмы прямого доступа к памяти;
* ”Е14” является программным обеспечением, которое может работать на любом компьютере под управлением Windows [11].

**2.2.4. Модель ЭВМ**

Учебная модель ЭВМ представляет собой программную модель, предназначенную для обучения основам работы компьютеров и кэшпамяти [15]. Структура модели включает в себя:

* процессор: основной вычислительный элемент, который управляет выполнением команд;
* оперативная память (ОЗУ): хранит данные и программы во время выполнения;
* сверхоперативная память: включает регистры общего назначения (РОН) и кэш-память для ускорения доступа к часто используемым данным;
* устройства ввода/вывода: позволяют взаимодействовать с внешними устройствами.

**2.2.5. RISC-V CPU Core (RV32IM) UltraEmbedded**

Это 32-битный синтезируемый процессор RISC-V, написанный на Verilog, поддерживающий набор инструкций RV32IM с расширениями для умножения, деления, управляющих регистров и прерываний. Процессор реализует пользовательский, супервизорный и машинный режимы, оснащён базовой системой управления памятью (MMU) и эмуляцией атомарных операций, что позволяет запускать на нём Linux. Ядро протестировано с помощью co-simulation моделей и проверено средствами Google RISCV-DV для гарантии корректности. Поддерживаются инструкции кэширования, интерфейсы AXI и конфигурация стадии конвейера. Реализация ориентирована на FPGA и моделирование со скоростью до 2.94 CoreMark/MHz, что делает её подходящей для обучения аппаратным архитектурам RISC-V и разработке системных ПО.

**2.2.6. Nand2Tetris**

Nand2Tetris — это комплексный двенадцатичастный курс по основам компьютерной инженерии, нацеленный на последовательное построение компьютера с нуля. Студенты начинают с реализации булевой логики и NAND-вентилей и доходят до создания полноценной операционной системы, способной запускать сложные программы, например, Тетрис. Курс создан профессорами Еврейского университета в Иерусалиме и содержит лекции, проекты и инструменты разработки. Особенность курса — системный практический подход, позволяющий наглядно разобраться в архитектуре компьютера и операционных системах, что делает его популярным в образовательных учреждениях по всему миру.

**2.2.7. Katvus/my\_processor**

Это процессор реализованный в рамках бакалаврской ВКР Екатерины Васильевой [7]. Процессор использует набор из 16 команд, которые включают арифметические и логические операции, операции с памятью и переходы.

Длина машинного слова процессора составляет 16 бит. Из них 4 бита выделены под кодирование операции, что позволяет закодировать 16 различных инструкций. Остальные 12 бит используются для кодирования номера регистра.

Процессор спроектирован с учетом простоты использования, что позволяет писать программы без излишней сложности. Это делает его подходящим для учебных целей и для понимания основ работы процессоров. Процессор протестирован на вычислении факториала.

**2.2.8. Выводы**

Embox представляет собой полноценную, простую в настройке операционную систему реального времени с поддержкой многозадачности, POSIX-интерфейса и возможностью запуска готового Linux-программного обеспечения. Благодаря модульной архитектуре и поддержке различных аппаратных платформ, включая RISC-V (RV32IM), Embox подходит как для практического использования, так и для образовательных целей.

Xv6, с другой стороны, является компактной и хорошо документированной учебной Unix-подобной ОС на C (около 9000 строк кода), что делает её отличным инструментом для изучения основных концепций операционных систем в академической среде.

Процессор UltraEmbedded RISC-V Core (RV32IM) обладает необходимым функционалом для моделирования и запуска ОС, но пока остается менее доступным в части полноты документации и методических материалов, что ограничивает его широкое образовательное применение без дополнительной поддержки.

Образовательный проект Nand2Tetris славится своей полной и систематизированной документацией, последовательно обучая основам построения компьютеров и операционных систем от простейших логических элементов, однако из-за своей масштабности и теоретической глубины он требует значительного времени для освоения.

Таким образом, Embox и Xv6 предлагают комбинацию простоты и полноты функционала для изучения и практического применения ОС, в то время как UltraEmbedded RISC-V Core и Nand2Tetris представляют две противоположные образовательные крайности: мощный, функциональный, но требующий дополнительного учебного сопровождения аппаратный процессор и всеобъемлющий учебный курс с богатой документацией, но более длительным и абстрактным изучением.

3. Описание решения

3.1 Технологии и инструменты разработки

Для проектирования виртуальных вычислительных устройств выбран язык описания аппаратуры **Verilog HDL**. Verilog является широко распространённым языком, ориентированным на описание и моделирование цифровых систем различной сложности. Его синтаксис схож с языком Си, что облегчает изучение и применение при разработке аппаратных моделей. Кроме того, Verilog поддерживает структурное, поведенческое и потоковое описание, что позволяет гибко создавать как логические схемы, так и сложные архитектурные решения.

В качестве средства симуляции выбрана **Icarus Verilog** — свободный, открытый инструмент с возможностями компиляции и моделирования Verilog-моделей. Он удобен для интерактивной отладки и демонстрирует достаточную производительность и совместимость с современными средствами разработки.

С точки зрения программного обеспечения, используется язык **C** для системного кода и **Python** для вспомогательных скриптов автоматизации, подготовки и трансляции программ. Для компиляции системного и прикладного ПО применяются компиляторы **GCC** и **TCC**, что обеспечивает совместимость с архитектурой RISC-V и универсальность инструментария.

3.2 Моделирование и эволюция систем

Процесс создания виртуальных вычислительных устройств построен поэтапно:

1. Начальный этап — разработка минималистичных моделей с ограниченным набором команд и упрощённой архитектурой, подходящих для глубинного изучения базовых принципов (например, NAND\_CPU).

2. Далее реализуются более сложные процессоры с полным набором базовых команд, способные реализовывать типичные вычислительные потоки (RISC-V RV32I CPU).

3. Затем проекты эволюционируют в сторону поддержки многозадачности, прерываний, расширенных архитектурных режимов и интеграции периферийных устройств.

Такой поэтапный подход позволяет постепенно расширять функциональность и масштаб систем, обеспечивая понятность и управляемость разработки.

3.3 Структура и архитектура виртуальных вычислительных устройств

NAND\_CPU построен из следующих основных компонентов:

* **Регистровый файл** — множество регистров общего назначения с поддержкой асинхронных считываний и синхронных записей.
* **Память** — модуль, осуществляющий хранение инструкций и данных.
* **Арифметико-логическое устройство (ALU)** — выполняет арифметические и логические операции, управляемые кодами команд.
* **Блок управления (Control Unit)** — декодирует инструкции и формирует сигналы управления для согласованной работы всех компонентов.

Команды декодируются по установленному формату, причем исполнение осуществляется пошагово с применением условных переходов, что обеспечивает универсальность и расширяемость архитектуры.

Рассмотренный RISC-V CPU включает следующие основные компоненты:

* **ALU (Arithmetic Logic Unit, ALU.v)** – выполняет арифметические операции (сложение, вычитание), логические операции (AND, OR, XOR, сдвиги). Принимает операнды и управляющие сигналы, выдаёт результат и флаги состояния.
* **ALUControl.v** – формирует управляющие сигналы ALU на основе опкода и полей инструкций (funct3, funct7), сопоставляя конкретные команды архитектуры с операциями ALU.
* **BranchComparator.v** – реализует условия переходов: проверку равенства, сравнения меньше, меньше или равно и т.п., выводит сигнал "взять ветвление".
* **ControlUnit.v** – основной блок декодирования инструкций, формирует сигналы управления для всех компонентов процессора: разрешение записи в регистровый файл, доступ к памяти, источник для ALU, включение ветвлений и др.
* **DataMemory.v** – блок памяти данных с синхронным доступом на чтение и запись, реализующий основную память RAM.
* **ImmediateGenerator.v** – выделяет и расширяет знаковым образом немедленные значения из форматов инструкций I, S, B, U, J, соответствуя спецификациям RISC-V.
* **InstructionMemory.v** – память команд, ROM, адресующаяся программным счётчиком, снабжающая команды на фазе выборки.
* **PCUpdate.v** и **ProgramCounter.v** – обеспечивают обновление и регистрацию программного счётчика с учётом ветвлений и переходов.
* **RegisterFile.v** – регистровый файл из 32 регистров общего назначения, поддерживающий асинхронное чтение и синхронное записывание.
* **RV32ICPU.v** – топ-уровневый модуль, интегрирующий все перечисленные компоненты для обеспечения полного процесса исполнения команды за один такт.

3.4 Реализация и интеграция программного обеспечения

Операционные системы для виртуальных устройств имеют модульную структуру, что упрощает адаптацию и расширение функционала под конкретные архитектуры. Компиляция ОС и прикладных программ выполняется специализированными трансляторами и компиляторами, а подготовка к запуску сопровождается скриптами для формирования исполняемых образов и загрузчиков.

Использование модульного подхода позволяет быстро интегрировать новые компоненты и оптимизировать существующий код, делая проект гибким и удобным для учебных целей.

4. Тестирование

4.1 Аппаратно-программная конфигурация

Тестирование и симуляция проводились на двух ноутбуках:

* Acer Aspire 5 с процессором Intel Core i3 1215U и графическим процессором Intel UHD Graphics;
* HP Pavilion 14 под управлением ОС Ubuntu 22 с процессором Intel Core i7 1355U и видео Intel Iris Xe.

4.2 Методика тестирования

Для проверки функциональности процессоров и программ поставлены следующие задачи:

* Выполнение команд арифметики (сложение, вычитание, логические операции, сдвиги);
* Проверка переходов по условию и безусловных переходов с метками;
* Проверка загруженных и сохранённых данных в памяти;
* Обработка системных вызовов (ECALL), в том числе ввод-вывод и завершение программы;
* Исполнение ассемблерных программ вычисления чисел Фибоначчи и умножения;
* Проверка корректного исполнения псевдоинструкций;
* Обработка ошибок и корректное завершение работы.

Для комплексного тестирования применялся следующий процесс:

1. Сборка тестовых ассемблерных программ и конвертация их в формат hex с помощью Python-скриптов;

2. Преобразование hex-файлов в бинарный формат, совместимый с эмулятором;

3. Запуск программ в симуляторе и сравнение результатов с ожидаемыми значениями;

4. Контроль работы регистров, памяти и системы прерываний;

5. Отладка исполнения инструкций на уровне машинного кода.

4.3 Результаты тестирования

На обоих процессорах (RISC-V CPU RV32I и NAND\_CPU) успешно выполнены программы вычисления чисел Фибоначчи и умножения. Эти тесты подтвердили корректность выполнения арифметики, логики, переходов и работы с памятью.

Кроме того, на UltraEmbedded RISC-V Core успешно запущен образ Linux ELF, что демонстрирует стабильность архитектуры и возможность запуска полноценной операционной системы.

5. Заключение

В ходе работы было принято решение о разработке образовательного комплекса, ориентированного на изучение архитектуры вычислительных машин на базе RISC-V. Существующие операционные системы и виртуальные устройства не всегда подходят для целей учебного курса по архитектуре ЭВМ из-за сложности реализации, громоздкости кода или ограниченного образовательного потенциала. Это препятствует глубокому и наглядному изучению ключевых концепций системного программирования и архитектуры процессоров.

Создание собственного комплекса позволит адаптировать учебный материал так, чтобы он максимально эффективно раскрывал основные принципы работы вычислительных систем и минимальных операционных систем. Такой комплекс обеспечит студентов наглядными примерами, удобными средствами тестирования и возможностью поэтапного углубления в архитектурные и системные детали.

Таким образом, предлагаемый образовательный комплекс будет способствовать более интересному и понятному изучению курса «Архитектура ЭВМ», облегчая восприятие сложных технических концепций и формируя прочную базу для будущей профессиональной деятельности студентов в области системного программирования и компьютерной инженерии.

Список литературы

1. BusyBox: швейцарский нож для встраиваемых Linux-систем. –– URL: <https://samag.ru/archive/article/1908> (дата обращения: 12 января 2025 г.).
2. CP/M. –– URL: <https://sysadminmosaic.ru/cp_m/cp_m> (дата обращения: 12 января 2025 г.).
3. Embox Documentation. –– URL: [https://non-descriptive.github. io/embox-mdbook/](https://non-descriptive.github.io/embox-mdbook/) (дата обращения: 12 января 2025 г.).
4. GNU Mes. –– URL: <https://www.gnu.org/software/mes/> (дата обращения: 12 января 2025 г.).
5. MicroDOS. –– URL: <https://ru.wikipedia.org/wiki/MicroDOS>(дата обращения: 12 января 2025 г.).
6. XV6 как ОС для обучения. –– URL: [https://habr.com/ru/ articles/597153/](https://habr.com/ru/articles/597153/) (дата обращения: 12 января 2025 г.).
7. katvus/my-processor. –– URL: [https://github.com/katvus/my\_ processor/tree/main](https://github.com/katvus/my_processor/tree/main) (дата обращения: 12 января 2025 г.).
8. Архитектура SIC / XE. –– URL: [https://progler.ru/blog/ arhitektura-sic-xe](https://progler.ru/blog/arhitektura-sic-xe) (дата обращения: 12 января 2025 г.).
9. Исследовательский UNIX. –– URL: [https://dit.isuct.ru/IVT/ BOOKS/OPERATING\_SYSTEMS/OPER7/GLAVA\_4.HTM](https://dit.isuct.ru/IVT/BOOKS/OPERATING_SYSTEMS/OPER7/GLAVA_4.HTM) (дата обращения: 12 января 2025 г.).
10. Компьютер маленького человечка. –– URL: [https://habr.com/p/ 257331/](https://habr.com/p/257331/) (дата обращения: 12 января 2025 г.).
11. Многопроцессорный учебный компьютер ”Е14”. –– URL: [https:// emc.orgfree.com/e14/](https://emc.orgfree.com/e14/) (дата обращения: 12 января 2025 г.).
12. Операционная система. –– URL: [https://blog.skillfactory.ru/ glossary/operaczionnaya-sistema/](https://blog.skillfactory.ru/glossary/operaczionnaya-sistema/) (дата обращения: 12 января 2025 г.).
13. Функции операционной системы. –– URL: [https://www.tadviser. ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F: %D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8\_%D0%BE%D0%](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B)

[BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B)

[BE%D0%B9\_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) (дата обращения: 12 января 2025 г.).

1. Ядро Linux. –– URL: [https://ru.wikipedia.org/wiki/%D0%AF%D0% B4%D1%80%D0%BE\_Linux](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_Linux) (дата обращения: 12 января 2025 г.).
2. модель учебной ЭВМ. –– URL: <https://emc.orgfree.com/model/>(дата обращения: 12 января 2025 г.).