Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа Группа 24М.71-мм

Разработка минимальных операционных систем для виртуальных устройств с различным функционалом

Киселькoв 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

В последние годы российская IT-отрасль демонстрирует значительное развитие. По оценкам, к концу 2024 года объем рынка достигнет примерно $52 миллиардов, что соответствует росту на 44% по сравнению с предыдущим годом. Основным фактором роста является программное обеспечение, доля которого увеличилась с 41% в 2023 году до 44,3% в 2024 году. В то же время наблюдается сокращение доли аппаратного обеспечения и IT-услуг. Существует множество дистрибутивов Linux и множество процессоров разных архитектур, но полностью разобраться в их работе чрезвычайно трудоемкая задача. Существуют материалы для изучения устройства компьютеров и дистрибутивов ОС, такие как Little Man Computer (LMC) и ранние версии Linux.

Но их недостаточно, что приводит к нехватке системных програмистов. В настоящее время отсутствует возможность постепенно рассмотреть реализацию операционных систем на компьютере сразу с аппаратной и цифровой стороны.

Это создает необходимость в разработке новых образовательных и исследовательских инструментов, которые бы позволили пользователям глубже понять принципы работы компьютера.

Исследование начнется с использования LMC и первой версии Linux, а также GNU/Mes. Виртуальные устройства будут разрабатываться с использованием Verilog. Постепенно будут добавляться новые задачи и функции (графический интерфейс вместо командной строки, добавление usb, сетевого адаптера...), что позволит создать полноценные образовательные инструменты для изучения основ компьютерных наук.

1. Постановка задачи

Целью данной работы является создание серии виртуальных процессоров и операционных систем на их основе, которые будут документированы и доступны для изучения. Эти системы должны соотвествовать определению ОС (Управление процессором, процессами, памятью, файлами, устройствами ввода-вывода и интерфейсом). Задачи исследования включают:

1. Разработка виртуальных процессоров и ОС для них.
2. Создание документации процессоров и ОС.
3. Реализация базовых функций в каждой из операционных систем.

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. Она выполняет множество ключевых функций, обеспечивая эффективное выполнение прикладных программ и управление системными ресурсами [12][13].

Основные функции операционной системы:

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 [14]. 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 тер-

минал. 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. 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 [4]. 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 ме-

ханизмы 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. E14

Модель "E14" 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 "E14" реализована несимметричная архитектура, где один из процессоров выполняет функции главного;
- система команд "E14" полностью совместима с предшествующей моделью "E97", 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;
- "E14" 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 кэш-памяти [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. 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.

2.2.6. Выводы

Среди рассмотренных операционных систем больше всего подходит Xv6. Данная система открыта, написана на C, полностью документирована и размер её кода составляет 9000 строк. Рассмотренные учебные компьютеры демонстрируют те или иные функции компьютеров. Однако все они обладают разным набором команд, что осложняет изучение.

К тому же ни на одном из приведенных компьютеров нет возможности запустить реальную операционную систему.

3. Описание решения

В рамках планируемой серии будет реализована система, поддерживающая команды предыдущих версий, что позволит четко выделить изменения, отвечающие за тот или иной функционал. Для разработки виртуальных устройств будет использоваться язык Verilog, который позволяет создавать модули для цифровых логических схем. Для симуляции Verilog-модулей будет использоваться Verilator — это мощный и быстрый симулятор, который преобразует код Verilog в C++ и позволяет выполнять высокопроизводительные симуляции. Verilator подходит для больших проектов, где критично важно быстродействие симуляции.

Реализация операционной системы будет выполнена на языке C без использования ассемблерных вставок. Операционная система должна поддерживать компиляцию и исполнение программного обеспечения, поэтому помимо самой ОС на виртуальном устройстве необходим компилятор. В качестве компилятора для C планируется использовать GCC или TCC.

3.1. Ключевые версии

3.1.1. Минимальный учебный компьютер

Необходимо реализовать виртуальное устройство подобное LMC, обладающее хорошей аналогией и простой реализацией.

3.1.2. Минимальная ОС

Упрощенная версия Xv6 на виртуальном устройстве. Следующая ключевая версия — это виртуальное устройство, поддерживающее операционную систему. Несмотря на компактность и простоту, Xv6 реализует множество функций достаточно гибко и продвинуто. Опираясь на определение операционной системы, можно создать упрощенную версию Xv6. Это увеличит вероятность её запуска на виртуальном устройстве и позволит оценить, какие доработки потребуются для запуска полноценной версии Xv6.

3.1.3. Xv6 на минимальном виртуальном устройстве

Последняя версия должна поддерживать Xv6.

3.2. Дополнительные версии

Помимо ключевых версий стоит реализовать версии, в которых к минимальному учебному компьютеру добавлены кэш, многоядерность, большее количество периферийных устройств и увеличенный список ассемблерных команд. Это позволит постепенно перейти от первого учебного компьютера до устройства с урезанной операционной системой.

3.2.1. RISC-V эмулятор и вспомогательные скрипты

Проект реализован на языке C и демонстрирует работу процессора RISC-V с помощью основных файлов, таких как cpu.c и memory.h. В этих файлах реализованы ядро процессора и система памяти, обеспечивающие выполнение инструкций, управление регистрами и доступ к памяти.

Для удобства работы с программами в ассемблерном виде используется отдельный Python-скрипт, который переводит текстовые ассемблерные команды в машинный код в формате hex. Этот hex-код затем с помощью другого Python-утилита конвертируется в бинарный файл, пригодный для загрузки и исполнения в эмуляторе процессора.

Таким образом, проект объединяет реализацию процессорного ядра на C и инструменты на Python для трансляции и подготовки программ к запуску, что позволяет изучать и тестировать работу процессора на уровне машинных инструкций.

1. **Эмулятор процессора (C)**

   - `cpu.c` — ядро эмулятора с:
     - Реализацией регистрового файла (32 регистра).
     - Системой памяти (RAM).
     - Циклом выборки-декодирования-исполнения.

- Поддерживает:
  - Целочисленные инструкции (R/I/S/B-форматы).
  - Системные вызовы через `ECALL`.
  - Базовые прерывания.

2. **Ассемблер (Python)**

   - `assembler.py` — транслятор ассемблерного кода в машинный код.
   - Принимает текстовый файл с инструкциями RISC-V.
   - Поддерживает:
     - Все базовые инструкции RV32I.
     - Псевдоинструкции (`li`, `mv`, `nop`, `j`).
     - Метки и вычисление смещений.
   - Выводит hex-коды в формате:

     ```
     02A00513
     00100893
     00000073
     ```

3. **Конвертер hex в bin (Python)**

   - `hex2bin.py` — утилита для преобразования hex-файла в бинарный формат.
   - Читает входной файл с hex-строками.
   - Преобразует каждую 8-символьную строку в 4 байта.
   - Записывает бинарный файл для загрузки в эмулятор.

4. Тестирование

4.1. Аппаратно-программная конфигурация

Тестирование проводилось на двух ноутбуках:

- Acer Aspire 5 с процессором Intel Core i3 1215U и видеокартой Intel UHD Graphics;
- HP Pavilion 14 с ОС Ubuntu 22. В ноутбуке установлены Intel Core i7 1355U и Intel Iris Xe Graphics.

4.2. Методика тестирования

Для тестирования минимального учебного компьютера необходимо проверить работу арифметической операции, перехода по условию и функцию переноса данных между ячейками памяти.

Задачи для тестирования минимальной ОС:

- выполнение команд во время параллельного расчета чисел Фибоначчи;
- создание, изменение, перемещение и удаление текстового файла;
- прием числа с клавиатуры и вывод его на экран;
- компиляция кода на C и исполнение полученной программы.

Для тестирования проекта эмулятора процессора RISC-V были поставлены следующие задачи:

- Проверка корректности выполнения арифметических операций (сложение, вычитание, логические операции, сдвиги).
- Тестирование переходов по условию и безусловных переходов с использованием меток.
- Проверка работы с памятью: загрузка и сохранение данных различных размеров (байт, слова).

- Тестирование системных вызовов через инструкцию `ECALL`, включая ввод-вывод и завершение работы.
- Проверка работы псевдоинструкций и правильности их трансляции в базовые инструкции.
- Запуск простых программ, например, вычисление чисел Фибоначчи, бесконечные циклы, ввод и вывод чисел.
- Проверка обработки ошибок и корректного завершения эмуляции.

Для комплексного тестирования использовался следующий подход:

1. Сборка тестовых программ в ассемблере с помощью Python-скрипта, переводящего их в hex-код.
2. Конвертация hex-файлов в бинарный формат для загрузки в эмулятор.
3. Запуск программ в эмуляторе и сравнение результатов с ожидаемыми.
4. Проверка корректности работы регистров, памяти и системы прерываний.
5. Отслеживание выполнения инструкций и отладка на уровне машинного кода.

4.3. Результат тестирования

На обоих устройствах my_processor успешно выполнил три поставленные задачи.

5. Заключение

В ходе работы за первый семестр была реализована первая ключевая версия минимальной ОС на Python. В следующем семестре планируется перенести её на Verilog и реализовать следующую ключевую версию.

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/ (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/ (дата обращения: 12 января 2025 г.).

[7] katvus/my-processor. — URL: 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 (дата обращения: 12 января 2025 г.).

[9] Исследовательский UNIX. — URL: 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/ (дата обращения: 12 января 2025 г.).

[11] Многопроцессорный учебный компьютер "Е14". — URL: https://emc.orgfree.com/e14/ (дата обращения: 12 января 2025 г.).

[12] Операционная система. — URL: 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%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.).

[14] Ядро Linux. — URL: https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_Linux (дата обращения: 12 января 2025 г.).

[15] модель учебной ЭВМ. — URL: https://emc.orgfree.com/model/ (дата обращения: 12 января 2025 г.).