1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский политехнический университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации
5. **ЛАБОРАТОРНАЯ РАБОТА №1**
6. **Принципы разработки ОС**  
   по дисциплине «Операционные системы»
7. Выполнил
8. студент гр. 4851004/90002 Чеченев А. Д.
9. <*подпись*>
10. Руководитель
11. acc. Крундышев В. М.
12. <*подпись*>
13. Санкт-Петербург
14. 2021

**ОГЛАВЛЕНИЕ**

[1 Формулировка задания 3](#_Toc94659997)

[2 Результаты работы 4](#_Toc94659998)

[2.1 Описание загрузчика 4](#_Toc94659999)

[2.2 Описание ядра 6](#_Toc94660000)

[2.3 Результаты тестов 7](#_Toc94660001)

[3 ВЫВОДЫ ПО РАБОТЕ 10](#_Toc94660002)

# Формулировка задания

Цель работы – изучение основ разработки ОС, принципов низкоуровневого взаимодействия с аппаратным обеспечением, программирования системной функциональности и процесса загрузки системы.

В рамках данной работы было необходимо:

1. Скомпилировать загрузчик и минимальное ядро. Проверить работоспособность загрузчика и ядра на эмуляторе

2. Разработать функции загрузчика.

3. Расширить реализацию минимального ядра, добавив в него функции вывода информации, подсчёта и выключения системы

4. Предложить не менее 15 тестов для проверки работоспособности ОС

# Результаты работы

Вариант задания: 5.

Работа выполняется в ОС Linux

Транслятор ассемблера для загрузчика: GNU assembler, синтаксис: Intel

Компилятор ядра: gcc

Задание для реализации: CalcOS

Функционал: расчёт, вывод информации и выключение системы, загрузчик передаёт цвет вывода текста

## Описание загрузчика

Загрузчик в данной работе выполняет в работе 3 основные функции: позволяет выбрать параметры запуска ядра, переводит процессор в защищённый режим и передает управление ядру. Загрузчик загружается по адресу 0х7С00 и затем просит пользователя указать цвет текста консоли. Данная задача решается при помощи вывода приветственной строки, которая побайтовом выводится из регистра bx (она помогает пользователю понять, что от него требуется) и сохранения в видеобуфер введённого пользователем символа. Этот функционал обеспечивает следящий участок кода:

mov ah, 0

int 0x16

mov bh, al

mov al, bh

mov ah, 0x0e

int 0x10

Он сохранят введённый пользователем символ с помощью прерывания 16h и выводит его в консоль с помощью прерывания 10h

Согласно заданию, после ввода символа сразу же происходит запуск, его ввод подтверждать не надо.

Загрузчик показан на рисунке 1:

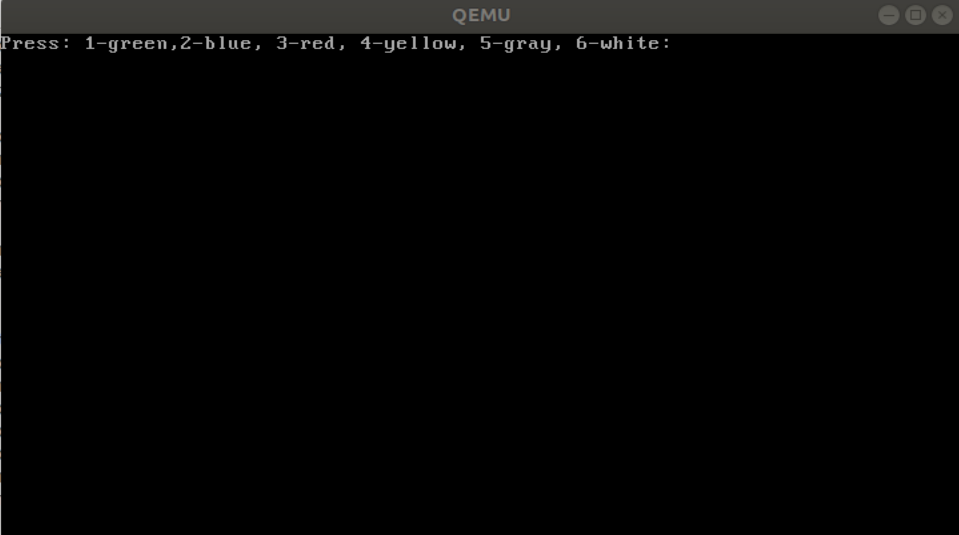


Рисунок 1 – Функционал загрузчика.

Блок-схема алгоритма работы загрузчика представлена на рисунке 2:

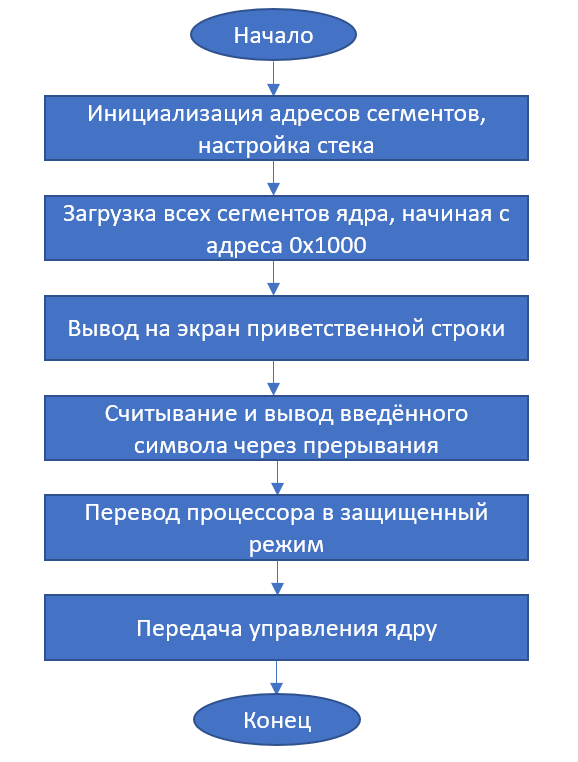


Рисунок 2 – Блок-схема алгоритма работы загрузчика.

## Описание ядра

Ядро взаимодействует с загрузчиком при помощи видеобуфера. Каждый символ в консоли – это 2 байта, первый отвечает за значение, второй – за цвет. Ядро начинает со считывания цвета из видеобуфера. Затем оно включает прерывания для того, чтобы после обработчика прерываний скан- коды всех введённых клавиш сохранялись в виде символов в видеобуфер на основе словаря. Нажатие enter запускает функцию, отвечающую за обработку основных функций программы: если советующая команда находится в видеобуфере происходит соответствующая операция: вывод информации подсчёт результата выражения или отключение системы.

Блок-схема алгоритма работы ядра представлена на рисунке 3:

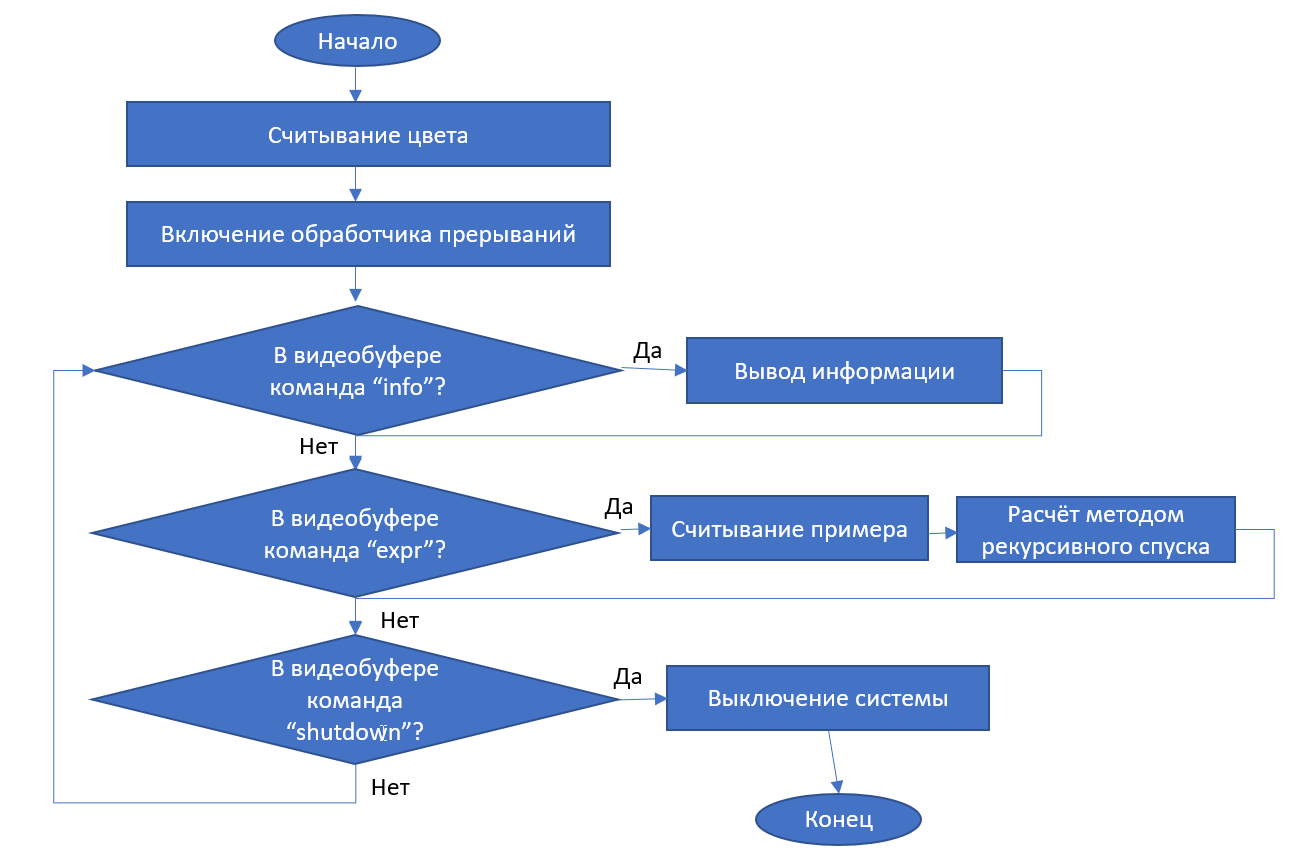


Рисунок 3 – Основные функции ядра.

Пример работы ядра представлен на рисунке 4:

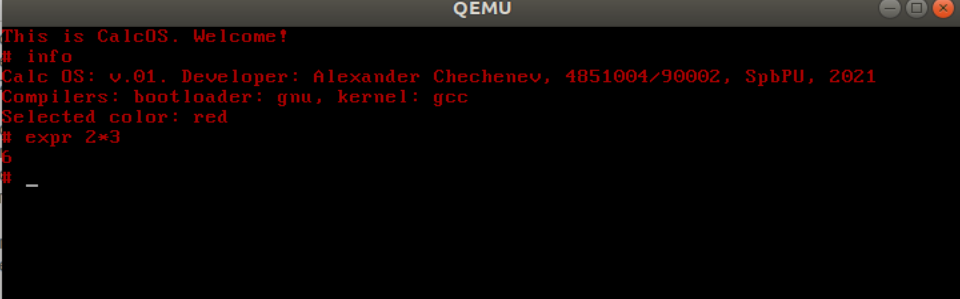


Рисунок 4 – пример работы ядра.

Калькулятор в данной программе реализован методом рекурсивного спуска: рекурсивным разбиением выражения на простые подвыражения и последующая рекурсивная замена их результата на результат. Данный алгоритм был выбран из-за простоты реализации и отсутствия необходимости работы с динамическими структурами данных.

## Результаты тестов

Команды для сборки загрузчика: as --32 -o bootsect.o bootsect.asm

ld -Ttext 0x7c00 --oformat binary -m elf\_i386 -o bootsect.bin bootsect.o

Команды для сборки ядра: g++ -fno-pie -ffreestanding -m32 -o kernel.o -c kernel.cpp

ld --oformat binary -Ttext 0x10000 -o kernel.bin --entry=kmain -m elf\_i386 kernel.o

Их запуск: qemu -fda bootsect.bin -fdb kernel.bin

Сначала проведем тестирование базового функционала (рисунок 5):

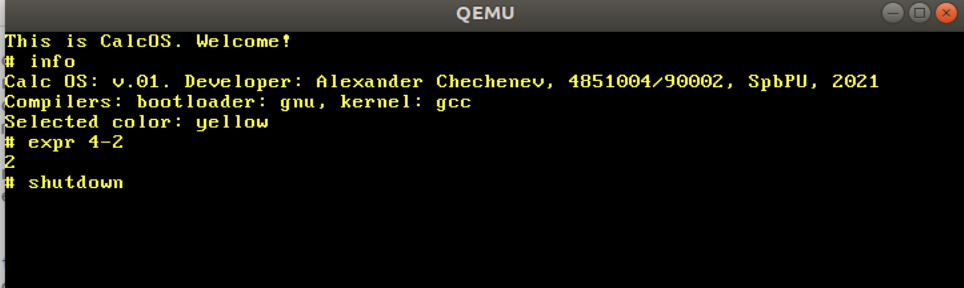


Рисунок 5 – Тесты 1-3.

Теперь проведем тестирование расчёта сложных выражений (Рисунок 6):

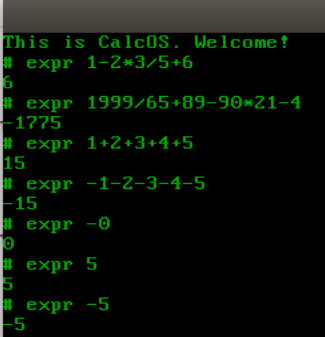


Рисунок 5 – Тесты 4-10.

Теперь проверим минусы и исключения (рисунок 6):

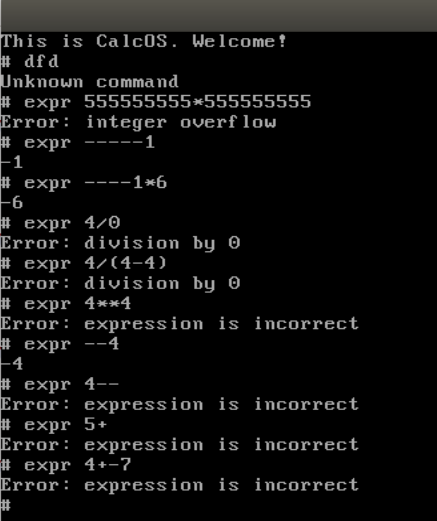


Рисунок 6 – Тесты 10-17.

Наконец, проверим скобок (рисунок 7)

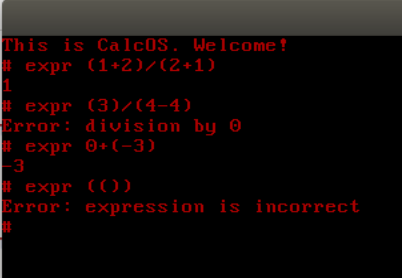


Рисунок 6 – Тесты 17-20.

Как видно, все представленные результаты полностью корректны.

# ВЫВОДЫ ПО РАБОТЕ

В результате данной работы были освоены базовые принципы разработки операционных систем. Было получено понимание механизма работы прерываний, защищенного режима процессора, передачи управления к ядру, работы с портами, создания интерфейса и функционала операционной системы. Также были получены практические навыки организации взаимодействия ядра и загрузчика. Был сделан вывод, что самый простой в реализации алгоритм преобразования строки в выражение без использования библиотек и динамических структур – это метод рекурсивного спуска.